Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«Владимирский государственный университет
имени Александра Григорьевича и Николая Григорьевича Столетовых»

Д. В. ВИНОГРАДОВ

РАЗРАБОТКА МОБИЛЬНЫХ ПРИЛОЖЕНИЙ И ОБЛАЧНЫЕ СЕРВИСЫ

Учебное пособие

Владимир 2022

УДК 33+004(075ю8)
ББК 65:32.973я73
В21

Рецензенты:
Кандидат экономических наук, доцент
зав. кафедрой экономики и финансов Финансового университета
при Правительстве Российской Федерации (Владимирский филиал)
Д. В. Кузнецов

Доктор экономических наук, доцент
профессор кафедры бизнес-информатики и экономики
Владимирского государственного университета
имени Александра Григорьевича и Николая Григорьевича Столетовых
А. М. Губернаторов

Виноградов, Д. В. Разработка мобильных приложений и
В21 облачные сервисы : учеб. пособие / Д. В. Виноградов ; Владим. гос. ун-т им. А. Г. и Н. Г. Столетовых. – Владимир : Изд-во ВлГУ, 2022. – 235 с. – ISBN 978-5-9984-1677-4.

Рассмотрена совокупность эффективных подходов, инструментов и методов, направленных на разработку приложений для мобильных устройств. Изложены методы управления связанными с мобильными приложениями процессами на всех стадиях его жизненного цикла: от формирования замысла до прекращения функционирования. Рассмотрены особенности разработки и обеспечения информационной безопасности мобильных приложений для устройств под управлением операционной системы Android, а также примеры проектирования, разработки интерфейса пользователя и баз данных мобильного приложения.

Предназначено для студентов направления подготовки 38.03.05 – Бизнес-информатика по профилю «Информационно-аналитическое обеспечение предпринимательской деятельности» всех форм обучения, а также будет полезно руководителям организаций и аналитических служб, специалистам, занимающимся вопросами использования мобильных приложений как элемента информационной архитектуры предприятия.

Рекомендовано для формирования профессиональных компетенций в соответствии с ФГОС ВО.

Ил. 56. Табл. 5. Библиогр.: 126 назв.

УДК 33+004(075ю8)
ББК 65:32.973я73

ISBN 978-5-9984-1677-4

ОГЛАВЛЕНИЕ

ВВЕДЕНИЕ

В учебном пособии рассмотрена совокупность эффективных подходов, инструментов и методов, направленных на разработку приложений для мобильных устройств.

В результате изучения материалов пособия у студента должны быть сформированы следующие профессиональные компетенции:

1) способность разрабатывать бизнес-планы, ценовую политику и стратегии развития серии ИТ-продуктов;

2) способность проводить идентификацию конфигурации информационной системы.

В результате освоения дисциплины студент должен демонстрировать следующие результаты образования:

знать:

1) основные подходы к разработке ценовой политики и теорию стратегического управления

2) основы функционирования информационных систем;

уметь:

1) формулировать стратегию развития;

2) определять элементы конфигурации информационных систем;

владеть:

1) способами прогнозирования и разработки ценовой политики;

2) навыками использования платформ информационных технологий.

Теоретической основой при работе над учебным пособием послужили современные концепции, категории и понятия, ведущие мировые практики и стандарты, используемые в области разработки мобильных приложений и использования областных сервисов.

Учебное пособие выступает как основа воспитания экономического мышления, понимания современных задач и методов в области разработки мобильных приложений и использования областных сервисов.

Глава 1. ОБЛАЧНЫЕ ВЫЧИСЛЕНИЯ, СЕРВИСЫ, МОБИЛЬНЫЕ ПРИЛОЖЕНИЯ

В главе рассматриваются следующие вопросы:

1. Общие сведения об облачных вычислениях, платформах и сервисах.

2. Роль мобильных приложений и сервисов в современном обществе.

3. Классификация мобильных приложений.

1.1. Облачные вычисления, платформы и сервисы

Облачные вычисления – «информационно-технологическая концепция обеспечения удобного сетевого доступа по требованию к некоторому общему фонду конфигурируемых вычислительных ресурсов (например, сетям передачи данных, серверам, устройствам хранения данных, приложениям и сервисам — как вместе, так и по отдельности), которые могут быть оперативно предоставлены и освобождены с минимальными эксплуатационными затратами или обращениями к провайдеру» [1].

Мировой рынок решений в области облачных вычислений интенсивно развивается в последние годы. Так расходы компаний на общедоступные облачные сервисы в 2021 году достиг 451 млрд. долл. Для сравнения затраты на все необлачные ИТ-решения составили 744 млрд. долл. При этом в 2017 году рынок общедоступных облачных сервисов составлял 117 млрд. долл. (рост рынка за пять лет составил почти 400%) [2].

Использование облачных вычислений для бизнеса стало частью общей стратегии цифровой трансформации современных предприятий, которая предполагает существенную трансформацию модели ведения хозяйственной деятельности, в рамках которой формируются инновационные подходы к организации производства, логистики, взаимоотношений с контрагентами и т.д. [3].

Использование облачных вычислений в рамках цифровой трансформации предприятия рассматривается как стратегическая задача по улучшению общей конкурентной позиции бизнеса на рынке, а не просто как поиск и использование резервов для снижения операционных

затрат (например, затрат на содержание и обслуживание собственной ИТ-решений).

Источником стратегических преимуществ использование облачных вычислений выступают:

1) ускорение вывода на рынок новой продукции;

2) снижение стоимости ошибок при реализации ИТ-решений;

3) упрощение организации совместной работы с партнерами и клиентами и др.

К числу обязательных характеристик, которыми должны обладать облачные вычисления относят:

1) самообслуживание по требованию (суть данной характеристики заключается в том, что пользователь сервиса облачных вычислений имеет возможность в одностороннем порядке без привлечения сотрудников провайдера выделять для своей работы все необходимые вычислительные ресурсы, например, мощности процессора серверов, место на жестком диске, объём оперативной памяти, пропускные возможности сетевого канала и т.д.);

2) универсальность сетевого доступа (суть данной характеристики заключается в том, что пользователь сервиса облачных вычислений имеет возможность использовать для подключения любое доступное ему терминальное устройство: персональный компьютер, планшет, смартфон и т.д.);

3) объединение ресурсов с целью оптимизации обслуживания (суть данной характеристики заключается в том, что провайдер сервиса облачных вычислений имеет возможность перераспределять доступные ему физические и виртуальные ресурсы между пользователями в зависимости от их текущих потребностей, обеспечивая при этом максимально возможное их использование);

4) эластичность предоставления услуг (суть данной характеристики заключается в том, что провайдер сервиса облачных вычислений имеет возможность мгновенного масштабирования ресурсов, предоставляемых пользователям, за счет автоматического их распределения и освобождения);

5) изменяемость оказанных услуг (суть данной характеристики заключается в том, что провайдер сервиса облачных вычислений имеет возможность в автоматическом режиме контролировать и исчислять потребляемые пользователем ресурсы).

Существуют следующие основные признаки классификации сервисов облачных вычислений:

1) используемая модель их развертывания;

2) используемая в них модель обслуживания.

Выделяют следующие модели развертывания сервисов облачных вычислений:

1) частное облако (предполагает использование ресурсов сервиса облачных вычислений только для решения задач одного субъекта хозяйствования);

2) коллективное облако (предполагает использование ресурсов сервиса облачных вычислений для решения общих задач связанных между собой организаций);

3) публичное облако (предполагает использование ресурсов сервиса облачных вычислений для решения задач любого количества организаций, как связанных между собой, так и нет);

4) гибридное облако (предполагает комбинацию и использование вышеперечисленных вариантов моделей использования сервисов облачных вычислений).

Заниматься администрированием частного облака может как сам его владелец (предприятие для решения задач, которого оно создается), так и третье лицо, нанятое им для указанных целей. Администрированием публичного облако занимается провайдер облачных услуг.

В соответствии с применяемой в сервисах облачных вычислений моделью обслуживания выделяют следующие их виды:

1) SaaS: сервисы, предоставляемые программное обеспечение в качестве услуги (пользователю сервиса облачных вычислений для решения его задач предоставляется программное обеспечение, которое выполняется на облачной инфраструктуре провайдера и может быть использовано с применением любого тонкого клиента или программного интерфейса);

2) PaaS: сервисы, предоставляемые платформы в качестве услуги (пользователю сервиса облачных вычислений для решения его задач предоставляется возможность размещения самостоятельно разработанных или приобретенных на стороне программных приложений, библиотек и инструментов, реализованных на любом поддерживаемой провайдером языке программирования);

3) IaaS: сервисы, предоставляемые инфраструктуру в качестве услуги (пользователю сервиса облачных вычислений для решения его задач предоставляется возможность использовать физические и виртуальные ресурсы провайдера (мощности процессора серверов, место на жестком диске, объём оперативной памяти, пропускные возможности сетевого канала и т.д.) для решения своих задач).

Основными провайдерами сервисов облачных вычислений являются лидеры ИТ-индустрии (например, IBM, Google, Microsoft, SAP), а также лидеры цифрового бизнеса (например, Amazon, Alibaba).

Среди наиболее популярных облачных платформ: Microsoft Azure, облачная платформа Google, Amazon S3, IBM Bluemix и SAP HANA.

1.2. Мобильные приложения и сервисы

Мобильные приложения – это программные продукты, разрабатываемые для смартфонов, планшетных компьютеров, других мобильных гаджетов, расширяющие функционал устройств [1].

В настоящее время мобильные устройства являются неотъемлемой частью повседневной жизни. Проведенные исследования указывают на широкое распространение мобильных устройств [4, 5]: на начало 2022 года в мире 5,31 миллиарда человек используют мобильные устройства (67.1% населения планеты), в России зарегистрировано 237.6 млн. работающих мобильных устройств (по некоторым данным более половины взрослого населения России имеют хотя бы одно устройство, а почти каждый третий – сразу два.

Мобильные устройства популярны среди различных категорий пользователей и используются для различных целей. Самыми популярными видами использования мобильных устройств являются [5]: общение в социальных сетях, обмен сообщениями с контактами (друзьями, знакомыми, коллегами), проверка (получение, чтение) электронной почты, веб-серфинг, ведение электронного ежедневника и совершение покупок в Интернете (23,4%).

По данным ряда исследований к началу 2021 года суточная аудитория интернета (ответившие, что выходили в Сеть в последние сутки) составила 65.2% от всех россиян, ежемесячная – 70.8%. При этом около

двух третей от всех пользователей российского сегмента интернета использует мобильные устройства наряду с компьютерами, а примерно треть и вовсе выходит в интернет только с них [5, 6].

По данным статистики портала LiveInternet [7] в мае 2022 года на долю пользователей мобильных устройств, пришлось около 78.4% от всех пользователей интернета или 140,5 млн. посетителей российских сайтов. Лидером среди используемых мобильных устройств являются устройства, работающие под управлением ОС Android (96,1 млн. пользователей или 68.4% от общего числа пользователей мобильного интернета). Особо следует отметить стремительный рост количества пользователей мобильных устройств за последние несколько лет: с 77 млн. пользователей в июне 2015 года до 140,5 млн. в мае 2022 года (или в 1,8 раза), в то время как общее количество пользователей выросло с 145 млн. до 185,7 млн. (или 1,3 раза).

По данным отчета GfK Crossmedia Landscape [8] россияне проводят в социальных сетях почти треть всего своего онлайн-времени (32% или почти 20 часов в месяц). Наибольший охват аудитории имеют социальные сети и поисковые системы. Почти 10-ю часть своего времени, проведенного в Интернет, пользователи тратят на покупки товаров (по оценке GfK, сегодня более 24 млн. россиян в возрасте 16-55 лет делают покупки в интернете) [5].

При этом отмечается рост доли заказов с мобильных устройств. Так, по данным совместного исследования GfK и Яндекс.Маркет [9] мобильные устройства приходится более 50 % заказов в российских интернет-магазинах. При этом с помощью мобильных приложений магазинов осуществляется более 60 % сделок.

Наиболее популярные категории для покупки в интернете – одежда и обувь, смартфоны и планшеты, косметика и парфюмерия, игрушки. Покупатели этих категорий – преимущественно женщины. При этом планшетом, как и компьютером, пользуются преимущественно из дома, а смартфоном – везде [5, 10].

Рост применения мобильных устройств обусловлен следующими группами факторов [5]:

1) экономическими (снижение стоимости мобильных устройств, снижение стоимости платы за доступ к чети интернет и т.д.),

2) социальными (мобильные устройства становятся стилем жизни, необходимым условием коммуникации, общения, развлечения и т.д.),

3) технологическими (рост производительности мобильных устройств, увеличение скорости работы в сети интернет, применение технологий распознавания образов, распознавания местоположения абонента и т.д.)

4) сервисными (рост количества мобильных приложений, с помощью которых можно получить услуги в любое время и в любом месте).

Популярность мобильных устройств способствует их более широкому внедрению в бизнес: разрабатываются мобильные приложения, ориентированные как на сегмент b2c так b2b; мобильные устройства становятся частью информационной инфраструктуры предприятия; растет число использования личных мобильных устройств для выполнения рабочих обязанностей, и т. д.

Мобильное приложение – это самостоятельный программный продукт, устанавливаемый под необходимую операционную систему мобильного устройства.

В настоящее время мобильные приложения играют ключевую роль практически во всех сферах, включая розничную торговлю, банковскую сферу, путешествия, рестораны быстрого обслуживания, товары широкого потребления, а также медиаиндустрию и индустрию развлечений [10].

В первой половине 2021 года в iOS App Store и Google Play предлагалось более 2 млн и более 3,5 млн мобильных приложений соответственно.

К началу 2022 года ежеквартальный объем мировой индустрии мобильных приложений достиг 34 млрд долл. По некоторым оценкам в ближайшие годы этот рынок может вырасти в несколько раз (этому будут способствовать инновации и переход от физических операций к мобильным) [11].

Доход разработчиков мобильных приложений обеспечивается с помощью следующих бизнес-моделей: платной установки приложений, покупки в приложении и размещении рекламы в приложении. По мере перехода индустрии приложений в более зрелую стадию предполагается распространение модели подписки, что позволит значительно увеличить доходы разработчиков мобильных приложений [5].

Также отмечается стремительный рост мобильных розничных продаж (ритейла) по всему миру. На мобильную сферу приходилось 44 % интернет-трафика ритейлеров и 31 % их продаж. Мобильный ритейл носит разноплановый характер, тем не менее, приложения мобильного ритейла можно условно отнести к одной из следующих категорий [5]:

1) совмещающие онлайн и офлайн каналы;

2) изначально цифровые.

В первую группу входят приложения компаний со значительным присутствием в офлайн-торговле, таких как М-видео, Эльдорадо, Ситилинк и др., а во вторую – приложения, ориентированные на онлайн-торговлю, такие как Amazon, OZON и т. д.

Несколько лет тому назад отличить интернет-ритейлеров от традиционных розничных продавцов было достаточно легко. Однако граница между ними становится все более размытой и это неминуемо приведут к изменению покупательских привычек. Предполагается, что в ближайшие годы будут расти ожидания людей в отношении безопасности, пользы и комфорта осуществления мобильных розничных покупок, что приведет к ситуации, когда для многих потребителей мобильная платформа станет основным способом совершения покупок, независимо от канала продаж. Более того, указанная тенденция может привести к появлению новой парадигмы розничной торговли.

Мобильные устройства благодаря тому, что они являются полноценными вычислительными устройствами, поддерживающими большую часть функционала традиционных электронных вычислительных машин при значительно меньших размерах, в настоящее время всё более активно используются в бизнесе, становясь одним из ключевых элементов информационной инфраструктуры предприятия [5, 12].

Применение мобильных технологий в бизнесе позволяет ускорить, облегчить и удешевить бизнес-процессы, помогая достигать основных целей бизнеса: получить прирост выручки, снизить издержки, привлечь новых клиентов, повышение статуса и компетентности компании в глазах клиентов и т. д. [5, 13].

Спрос на использование мобильных устройств, как ключевого элемента информационной инфраструктуры предприятия, привел к появлению класса информационных систем, поддерживающих возможность использования мобильных устройств в корпоративных деловых процессах, – систем управления корпоративной мобильной средой

(Enterprise Mobility Management, EMM) [14]. Реализация таких систем основана на интеграции данных аппаратных средств в ИТ-системы и в среды обеспечения безопасности на всех этапах управления жизненным циклом ИТ.

Организации могут использовать EMM-решения для выполнения следующих задач своих пользователей:

1) конфигурирование и настройка мобильных устройств, а также управление жизненным циклом приложений, необходимых для использования в бизнес-процессах предприятия;

2) аудит, отслеживание и подготовка отчетов об использовании мобильных устройств и установленных приложений, их конфигурации, настройках, и т.д.;

3) защита данных;

4) поддержка пользователей, устранение проблем с мобильными устройствами с помощью инвентаризации, аналитики и удаленных действий.

Решение указанных задач основано на следующих технических возможностях EMM [5]:

1) на управлении мобильными устройствами (Mobile device management, MDM) – зависящем от платформы наборе сервисов и технологий управления жизненным циклом, который обеспечивает инвентаризацию, управление конфигурацией ОС, подготовку и удаление устройств, удаленное удаление и удаленный просмотр / контроль для устранения неполадок;

2) на управлении мобильными приложениями (Mobile application management, MAM) – наборе сервисов и технологий управления и контроля для отдельных приложений (которые доставляются либо через хранилище корпоративных приложений, либо с использованием собственных API-интерфейсов ОС), а затем управляются локально на устройствах через консоль EMM;

3) на управлении идентификацией (Mobile identity management, MIM) – наборе сервисов и технологий управления идентификацией и доступом (для оценки решений о доступе может использоваться SIM-карта мобильного устройства, а также контекстуальная информация: местоположение, время и т.д.);

4) на управлении мобильным контентом (Mobile content management, MCM) – наборе сервисов и технологий управления распространением контента с помощью мобильных устройств, в том числе: а) принудительное применение политик безопасности (не зависящих от устройства ключи шифрования, аутентификации, правил обмена файлами, на копирование и вставку) для отдельных файлов; б) соблюдение правил для распространения, замены и удаления на основе push-файлов; в) мобильную совместимость для сторонних систем управления правами, а также инфраструктуры защиты корпоративных данных (DLP) и корпоративных цифровых прав (EDRM);

5) на политике сдерживания (Containment) – наборе сервисов и технологий инкапсуляции MDM, MAM, MIM и / или MCM в специальные изолированные от персональных приложений и других пользователей среды.

При этом отмечается, что функциональные возможности EMM-решений при первичном внедрении востребованы заказчиками в среднем не более чем на 10%, наиболее продвинутые заказчики развертывают EMM-решения, обладающие 30-40% функциональных возможностей. Модули MIM, MCM и Containment внедряет небольшое количество компаний, а все пять модулей – единицы [5, 14].

Использование работниками, деловыми партнерами и другими пользователями собственных мобильных устройств при работе с корпоративными ресурсами (далее – консьюмеризация ИТ, Consumerization of IT) несет дополнительные выгоды для бизнеса: экономию затрат на приобретение и обслуживание мобильных устройств, обеспечение более надежной защиты личной конфиденциальной информации и повышение мобильности работников [15]. Предоставление возможности доступа к актуальной корпоративной информации из любого места обеспечивает высокую эффективность операционной деятельности, удобство работы и как следствие удовлетворенность сотрудников работой [16].

Работники при консьюмеризации ИТ получают возможность работать с привычным мобильным устройством, иметь свободу выбора тех технологий, которые они сами хотят использовать, иметь возможность использовать социальные сети, чаты, блоги и др.

Консьюмеризация ИТ может осуществляться в различных режимах [5]:

1) BYOD («Bring Your Own Device», «принести своё собственное устройство»),

2) BYOT («Bring Your Own Technology», «принесите свою собственную технологию»),

3) BYOPC («Bring Your Own Personal Computer», «принесите свой собственный персональный компьютер»)

4) BYOP («Bring Your Own Phone», «принесите свой собственный телефон»).

Ключевые тенденции, связанные с применением BYOD, были исследованы аналитиками Crowd Research Partners [17]. В качестве причин, побуждающих предприятия использовать BYOD, выступают факторы, связанные с персоналом: повышение мобильности сотрудников (63 % от общего числа респондентов), рост удовлетворения сотрудников (56 %) и рост производительности (55 %) сотрудников. Следует отметить, что данные факторы оказались важнее, чем снижение затрат (47 %), безопасность (39 %) и конфиденциальность информации сотрудников (12%) [5].

Подавляющее большинство организаций предоставляют BYOD работникам (76 % от общего числа респондентов), а также подрядчикам – 23 %, партнёрам – 16 % и клиентам – 14 % [5].

Самым популярным типом мобильных приложений, включенным в BYOD, является управление электронной почтой, календарем и контактами (84 % от общего числа респондентов). Доступ к корпоративным документам предоставляют 45 % респондентов, доступ к корпоративной сети – 43 %, доступ к видеоконференциям – 35 %, доступ к файлам – 33 %, доступ к приложениям SaaS – 28 %, доступ к удаленному рабочему столу – 14 % и т. д. [5].

Уровень организационной поддержки пользователей BYOD варьируется. Специальную поддержку без выделения в отдельный бизнес-процесс предоставляют 32 % организаций, ограниченную поддержку – 27 %, не оказывает никакой поддержки (возлагают ответственность за поддержку на собственников мобильных устройств) – 23%, полную поддержку осуществляет 15 % [5].

Средства управления мобильными устройствами (MDM) использует 43% организаций, средства Endpoint Security Tools – 28 % и Network Access Controls (NAC) – 27 % [5].

1.3. Классификация мобильных приложений

Количество мобильных приложений наклонно растет и их общее количество только в общедоступных репозиториях (iOS App Store и Google Play) достигает более 5,5 млн. единиц. Для классификации всего огромного множества доступных для использования мобильных приложений используются следующие основные признаки:

1) роль мобильного приложения для его владельца;

2) исполняемые мобильным приложением функции;

3) технология разработки мобильного приложения;

4) способ извлечения выгоды от распространения мобильного приложения.

Также классифицировать мобильные приложения можно по целевой аудитории мобильного приложения, его стоимости и другим признакам.

Мобильное приложение может вводится его владельцем в хозяйственный оборот либо как экономический ресурс, либо как товар. В первом случае оно выступает как корпоративное мобильное приложение, во втором – как коммерческое мобильное приложение.

Корпоративные мобильные приложения являются неотъемлемой частью системы автоматизации бизнес-процессов предприятия, обеспечивая для его владельцев, сотрудников и партнеров решение следующих задач:

1) доступ к оперативной информации (например, к данным о продукции, клиентах, заказах и сделках);

2) осуществление хозяйственных операций (например, выписка счета, электронная подпись накладных);

3) осуществление управления предприятием и партнерскими взаимоотношениями (например, обеспечение оперативных коммуникаций, финансовое планирование).

Коммерческие мобильные приложения являются товаром и предназначены для реализации на рынке тем или и иным способом. Способы извлечения выгоды от распространения мобильного приложения как товара называются моделями монетизации и будет рассмотрены далее).

С точки зрения исполняемых функций мобильные приложения могут быть отнесены к одной из следующих категорий [1]:

1) развлечения (игровые приложения, мультимедиа, музыка, заказ билетов в театр, кино и т.п.);

2) путешествия (заказ отеля, аренда авто, услуги гида, сервис онлайн-переводчика и т.п.);

3) бизнес (финансовые приложения, планирование, торговля, приложения для города, поиск работы и т.п.);

4) социальные приложения (социальные сети, глобальные брендированные сети, специализированные (клубные) сети и т.п.);

5) еда (заказ и доставка еды, геолокация заведения питания, рецепты);

6) спорт (спортивные новости, покупка билетов на спортивные мероприятия, игровые симуляторы);

7) образование (обучающие программы, интерактивные курсы и т.п.);

8) новости (дайджесты, ленты, рейтинги).

В зависимости от технологии разработки различают следующие виды мобильных приложений:

1) нативные (особенностью данного вида мобильных приложений является то, что создаются они под конкретную операционную систему и устанавливаются непосредственно на мобильное устройство пользователя);

2) кроссплатформенные (особенностью данного вида мобильных приложений является то, что создаются они как веб-приложение и установке их мобильное устройство пользователя не требуется);

3) гибридные (особенностью данного вида мобильных приложений является то, что создаются как комбинация нативных и кроссплатформенные приложений).

Нативные мобильные приложения имеют полный доступ к функционалу мобильного устройства (например, геолокации, звонкам, SMS, камере, микрофону). Скорость работы данного вида мобильных приложений высокая. Распространение осуществляется, как правило, через магазины приложений.

Кроссплатформенные мобильные приложения не имеют доступа к функционалу мобильного устройства и скорость их работы определяется параметрами сетевого подключения.

Способ извлечения выгоды от распространения мобильного приложения зависит от того какую роль мобильное приложение выполняет для его владельца.

Если речь идет о корпоративном мобильном приложении, то выгода от его использования заключается в снижении операционных и транзакционных затрат, связанных с доступом к оперативной информации, работы с первичной документацией, а также с управлением бизнес-процессами.

Для коммерческих мобильных приложений существует несколько ключевых способов извлечения выгоды от их распространения (моделей монетизации) [18]:

1) платные приложения (суть данной модели монетизации заключается в том, что перед тем, как загрузить мобильное приложение пользователь должен его оплатить);

2) условно-бесплатные приложения (суть данной модели монетизации заключается в том, что пользователю изначально бесплатно доступен базовый функционал мобильного приложения, а дополнительные возможности доступны только после оплаты);

3) модель подписки (суть данной модели монетизации заключается в том, что пользователь получает новый доступный в мобильном приложении информационный контент на основе его периодической оплаты);

4) рекламная модель (суть данной модели монетизации заключается в том, что внутри мобильного приложения продаются рекламные места или рекламируется собственная продукция разработчика);

5) модель выполнения рекламных действий (суть данной модели монетизации заключается в том, что за совершения определенных действий пользователю мобильного приложения предоставляются дополнительные функциональные возможности или информационный контент).

Вопросы для обсуждения

1. Понятие облачных вычислений

2. Использование облачных вычислений в рамках цифровой трансформации бизнеса

3. Источники стратегических и оперативных преимуществ использование облачных вычислений

4. Формы и модели реализации обязательных характеристик облачных вычислений

5. Модели развертывания сервисов облачных вычислений

6. Анализ возможностей наиболее популярных облачных платформ: Microsoft Azure, облачная платформа Google, Amazon S3, IBM Bluemix и SAP HANA

7. Идентификация и анализ факторов рост применения мобильных устройств

8. EMM-системы: решаемые задачи и технические возможности

9. Основные признаки классификации мобильных приложений

10. Основные способы извлечения выгоды от распространения мобильных приложений

11. Практическое использование SaaS-, PaaS- и IaaS-сервисов

Практические задания

Задание 1.1.

Сформулируйте задачи, возникающие при ведении заданной хозяйственной деятельности (по вариантам) коммерческим предприятием, которые можно было бы решить с использованием SaaS-сервисов. Оцените приблизительные сроки и стоимость реализации каждой из них с использование и без использования сервисов облачных вычислений.

Варианты видов деятельности:

1) банковская деятельность;

2) электронная коммерция;

3) риэлтерская деятельность;

4) оказание услуг общественного питания;

5) розничная торговля;

6) сельское хозяйство;

7) транспортные услуги;

8) туристические услуги;

9) услуги страхования;

10) бытовые услуги.

Задание 1.2.

Сформулируйте задачи, возникающие при ведении заданной хозяйственной деятельности коммерческим предприятием (вариант принять по заданию 1.1), которые можно было бы решить с использованием PaaS-сервисов. Оцените приблизительные сроки и стоимость реализации каждой из них с использование и без использования сервисов облачных вычислений.

Задание 1.3.

Сформулируйте задачи, возникающие при ведении заданной хозяйственной деятельности коммерческим предприятием (вариант принять по заданию 1.1), которые можно было бы решить с использованием IaaS-сервисов. Оцените приблизительные сроки и стоимость реализации каждой из них с использование и без использования сервисов облачных вычислений.

Задание 1.4.

Осуществите классификацию любого популярного мобильного приложения, которое соответствует следующему виду деятельности (по вариантам):

1) банковская деятельность;
2) рекламная деятельность;
3) риэлтерская деятельность;
4) оказание услуг общественного питания;
5) розничная торговля;
6) сельское хозяйство;
7) транспортные услуги;
8) туристические услуги;
9) услуги страхования;
10) бытовые услуги;
11) страховая деятельность;
12) доставка продуктов;
13) образовательная деятельность.

В качестве классификационных признаков используете:

1) роль мобильного приложения для его владельца;
2) исполняемые мобильным приложением функции;
3) технология разработки мобильного приложения;

4) способ извлечения выгоды от распространения мобильного приложения.

Библиографический список

1. Цифровой бизнес : учебник / под науч. ред. О. В. Китовой. — Москва : ИНФРА-М, 2021. — 418 с. — (Высшее образование: Магистратура). - ISBN 978-5-16-013017-0. - Текст : электронный. - URL: https://znanium.com/catalog/product/1659834 (дата обращения: 04.05.2022)

2. Облачные вычисления (мировой рынок) // TAdviser: портал: [Электронный ресурс]. – Режим доступа: https://www.tadviser.ru/index.php/Статья:Облачные_вычисления_(мировой_рынок)

3. Виноградов Д.В. Проблемы экономического обоснования выбора организационно-технических решений при реализации инновационных проектов в рамках цифровой трансформации бизнеса // Наука Красноярья, Том 11, № 1-2, 2022

4. Digital 2022: Global overview report. [Электронный ресурс] // datareportal.com: портал. URL: https://datareportal.com/reports/digital-2022-global-overview-report (дата обращения: 02.05.2022)

5. Цифровая экономика / Н. В. Абдуллаев, Н. Л. Аванесян, Д. В. Виноградов [и др.]. – Москва : Компания КноРус, 2018. – 286 с. – ISBN 978-5-4365-3040-6. – EDN YPRLET.

6. Развитие интернета в регионах России. [Электронный ресурс] // mediascope.net: портал. URL: https://mediascope.net/news/1250827/ (дата обращения: 02.05.2022)

7. Статистика российского сегмента интернета. [Электронный ресурс] // LiveInternet: портал. URL: https://www.liveinternet.ru/stat/ru/oses.html?period=month (дата обращения: 02.05.2022).

8. Тренды поведения россиян в интернете в 2017 году. [Электронный ресурс] // GfK: официальный сайт. URL: http://www.gfk.com/ru/insaity/press-release/issledovanie-gfk-trendy-

povedenija-rossijan-v-internete-v-2017-godu/ (дата обращения: 02.05.2022).

9. Развитие розничной онлайн-торговли в России. [Электронный ресурс] // Яндекс: портал. URL: https://yandex.ru/company/researches/2020/ecomdash (дата обращения: 02.05.2022).

10. Развитие онлайн-торговли в России: покупки со смартфонов [Электронный ресурс] // Яндекс: портал.URL: https://yandex.ru/company/researches/2017/mobile_retail (дата обращения: 02.05.2022).

11. App Annie Forecasts 36 Billion Downloads and $34 Billion Consumer Spend Globally in Q3 2021, Marking the Biggest Quarter Yet. [Электронный ресурс] // App Annie: официальный сайт. URL: https://www.data.ai/ru/insights/market-data/q3-2021-biggest-quarter-yet/ (дата обращения: 02.05.2022).

12. Сидак А.А., Ильин А.В., Кубарев А.В. Мобильные устройства в информационных системах и угрозы безопасности информации. Взаимосвязи. // Вопросы кибербезопасности. №4. 2014.

13. Иван Семчук, Илья Каштанкин, Максим Болышев. Облака и мобильность ломают старые процессы. [Электронный ресурс] // CNews: портал. 2016. URL:http://www.cnews.ru/reviews/mobilnost_v_ biznese_2016/ arti-cles/oblaka_i_mobilnost_lomayut_ starye_protsessy (дата обращения: 13.01.2018).

14. Rob Smith, Bryan Taylor, Manjunath Bhat, Chris Silva, Terrence Cosgrove. Magic Quad-rant for Enterprise Mobility Management Suites. [Электронный ресурс]. // Gartner: официальный сайт. 03.06.2017. URL: https://www.gartner.com/doc/reprints?id=1-42A6Q84&ct=170607 (дата обращения: 13.01.2018).

15. Johnson, K. & Filkins, B. L., 2012. SANS Mobility/BYOD Security Survey, s.l.: SANS Institute

16. E. T. Tchao, Richard Y. Ansah, Seth D. Kotey. Barrier Free Internet Access: Evaluating the Cyber Security Risk Posed by the Adoption of

Bring Your Own Devices to e-Learning Net-work Infrastructure. // International Journal of Computer Applications (0975 – 8887). Volume 176 – No.3, October 2017

17. Holger Schulze. BYOD & Mobile security: spotlight report. // Crowd Research Partners: официальный сайт. 2017. URL: http://crowdresearchpartners.com/wp-

18. Семенчук, В. Мобильное приложение как инструмент бизнеса: Справочное пособие / Семенчук В. - М. : АЛЬПИНА, 2017. - 240 с. ISBN 978-5-9614-6334-7. - Текст : электронный. - URL: https://znanium.com/catalog/product/1002640 (дата обращения: 04.05.2022

Глава 2. ПРИМЕНЕНИЕ МОБИЛЬНЫХ ПРИЛОЖЕНИЙ В ОТДЕЛЬНЫХ ОТРАСЛЯХ ЭКОНОМИКИ

В главе рассматриваются следующие вопросы:

1. Функции корпоративных мобильных приложений.

2. Использование мобильных приложений в промышленности и строительстве.

3. Использование мобильных приложений в торговле и сфере услуг.

4. Использование мобильных приложений в финансовой сфере.

2.1. Функции корпоративных мобильных приложений

Современное предприятие, ориентированное на работу с клиентами, а также эффективное взаимодействие сотрудников между собой и с деловыми партнерами, невозможно представить без мобильного приложения [1, 2, 3].

Корпоративное мобильное решение выполняет следующие функции [4]:

1) маркетинговую функцию, которая заключается в распространении фирмой информации о себе во внешнем мире с целью формирования положительного образа (имиджа) бизнеса и увеличения сбыта продукции;

2) коммуникационную функцию, которая заключается в формировании и поддержке коммуникаций между предприятием и его потребителями и партнерами;

3) организационно-управленческую функцию, которая заключается в упорядочении и приспособлении организации, процессов производства и реализации продукции предприятия к осуществлению бизнеса в условиях цифровизации.

Маркетинговая функция реализуется посредством:

1) продвижения торговой марки, принадлежащей фирме;

2) адресной рекламы продукции;

3) предоставления информации о фирме потенциальным контрагентам;

4) создания обратной связи с клиентами;

- предоставления подробной информации о продуктах и услугах предприятия;

- предоставления прочей информации клиентам, формирующей у них положительный образ предприятия.

Коммуникационная функция реализуется посредством:

- обеспечения командной работы сотрудников и партнеров предприятия над задачами и проектами;

- обеспечения ведения CRM-баз данных и документооборота, для эффективных внутренних коммуникаций и PR-кампаний и т.д.

Организационно-управленческая функция реализуется посредством:

- обеспечения развития организационной структуры управления в условиях цифровизации;

- обеспечения интеграции технологических и бизнес-процессов предприятия в партнерские сети;

- обеспечения эффективного управления персоналом;

- обеспечения эффективного управления качеством продукции и т.д.

Экономическая эффективность создания корпоративного мобильного приложения достигается как за счет роста продаж, так и за счет [2]:

1) снижения стоимости ввода информации, благодаря использованию централизованных баз данных;

2) снижения затрат на бумагу вследствие использования электронного документооборота;

3) снижения затрат рабочего времени сотрудников;

4) снижения транзакционных затрат (затрат на коммуникацию).

Всё вышеизложенное обуславливает рост активного применения мобильных приложений в различных отраслях экономики.

2.2. Использование мобильных приложений в промышленности и строительстве

Промышленность является одним из ключевых сфер реального сектора экономике, где использование мобильных технологий имеет огромный потенциал.

Основной задачей, которую решают в настоящий момент на промышленных предприятиях, является повышение эффективности отслеживания производственных ресурсов. Мобильные решения разрабатываются в индивидуальном порядке с учетом специфики и уровня развития производственных возможностей предприятия, а также применяемых технологий (например, наличия приборов для сканирования штрихкодов, наличия специализированных устройств ввода-вывода (I/O) и радиочастотной идентификации (RFID) и систем определения местоположения в режиме реального времени (RTLS)).

Мобильные приложения находят применение для:

1) мониторинга и оценки состояния производственного оборудования (например, для паспортизации, расчета индексов технического состояния, оптимизации планов технического обслуживания, автоматизации обходов и осмотров);

2) контроля персонала (например, для контроля за исполнением сотрудником своих должностных обязанностей, доступом в определенные помещения, а также осуществления учета рабочего времени сотрудников и их действий);

3) инспектирования производственных помещений;

4) отслеживания заявок и претензий потребителей и контрааген-тов;

5) управления перевозками;

6) и многих других задач.

В связке с другими технологиями мобильные приложения дают синергетический эффект, который определяет их эффективное использование для решения следующих задач [5]:

1) формирование IoT-окружения и управление им (мобильные приложения позволяют взаимодействовать с современным производственным оборудованием на основе пула стандартных протоколов передачи и обмена данными, определяя их текущее состояние в режиме

реального времени и выбирая наиболее оптимальный режим их нагрузки);

2) формирование среды, обучающей персонал профессиональным навыкам (мобильные приложения позволяют проходить будущим специалистам обучение в виртуальной среде (например, приобретать навыки работы на сложном высокотехнологическом оборудовании), значительно снижая затраты на подготовку и сокращая время обучения);

3) формирование корпоративного искусственного интеллекта (мобильные приложения позволяют обеспечить каждого работника предприятия возможностью доступа к корпоративной базе знаний в форме виртуального ассистента с элементами AI, способного помочь в решении большинства рутинных задач, что значительно снижает коммуникационные издержки и позволят персоналу сосредоточится на устранении нестандартных или стратегических проблем);

4) формирование эффективных производственных цепочек (мобильные приложения позволяют обеспечить мониторинг и настройку параметров автономно работающих групп оборудования, которые функционирует на основе взаимообмена информацией и самоорганизации).

Практика совместного применения информационного моделирования строительных объектов и мобильных решений продемонстрировала высокую эффективность при реализации инвестиционно-строительных проектов различного масштабы и уровня сложности.

В качестве эффекта совместного применения BIM-технологий и мобильных решений выступают:

1) прирост показателей экономической эффективности проекта (NPV, PI сроков окупаемости, IRR);

2) сокращение сроков строительства;

3) повышение точности проекта;

4) повышение эффективности взаимодействия участников проекта.

Основными причинами указанных эффектов называют:

1) сокращение ошибок при проектировании;

2) сокращение длительности этапа проектирования и процесса формирования проектной документации;

3) уменьшение количества коллизий в проектной документации;

4) уменьшение количества запросов на дополнительную информацию и запросов на изменения по проекту;

5) сокращение сроков подсчета объемов строительных работ и последующей корректировки сметных расчетов;

6) повышение точности сметных расчетов;

7) снижение затрат на этапе строительства и эксплуатации;

8) рост производительности труда;

9) сокращение продолжительности процедуры экспертизы;

10) снижение административных расходов;

11) повышение эффективности взаимодействия участников;

12) организация эффективного мониторинга и контроля работ по проекту.

2.3. Использование мобильных приложений в торговле и сфере услуг

Предприятия розничной и оптовой торговли в настоящий момент активно внедряют в свою деятельность мобильные технологии. Данная тенденция обусловлена нарастающими темпами развития онлайн-шопинга, а также продолжением развития интеграции онлайн и офлайн среды торговых предприятий.

Применение мобильных приложений при осуществлении дистанционной торговли посредством Интернета развивается благодаря следующим факторам:

1) совершенствованию правового регулирования заключения и исполнения сделок дистанционной купли-продажи товаров;

2) увеличению охвата мобильными технологиями населения, а также рост технических навыков использования мобильных устройств;

3) росту охвата и качества интернет-связи;

4) развитию услуг быстрой доставки товаров, а также логистических служб и пунктов самовывоза заказанных товаров;

5) совершенствованию электронных средств презентации качественных характеристик товаров;

6) развитию безналичных платежных систем и сервисов, позволяющих производить оплату заказанных товаров онлайн, в том числе с использованием мобильных приложений.

Установка мобильного приложения позволяет наладить между продавцом и покупателем коммуникации, каждому предоставляя новые ранее недоступные возможности.

Для продавца такая возможность заключается в повышении лояльности покупателя за счет оперативной реакции на изменение его потребительских предпочтений, для покупателя – в упрощении скорости и удобства совершении сделки.

Современные мобильные приложения розничной Интернет-торговли, как правило, обладают следующими стандартными функциональными возможностями:

1) просмотр каталога с возможностью последующего оформлением заказа;

2) просмотр информации о текущих акциях;

3) ведение списка желаний и планируемых покупок;

4) обращение за консультацией относительно планируемой покупки или в службу поддержки относительно урегулирования возникающих проблем при исполнении сделки купли-продажи и гарантийного обслуживания;

5) просмотр истории взаимодействия с торговой площадкой (например, ознакомление с историей ранее сделанных заказов и их статусами, с историей обращений в службу поддержки, накопленных бонусах и т.д.);

6) привязка платежных данных к учетной записи;

7) участие в доступных только пользователям мобильных приложений закрытых распродажах и акциях.

Развитие и диффузия в общество технологий дополнительной реальности позволят наделить мобильное приложение следующим функциональными возможностями:

1) онлайн-примерка товаров (например, примерка одежды и аксессуаров на себе, оценка соответствия приобретаемой мебели существующему окружающему интерьеру и т.д. – такой подход дает возможность покупателю быстрее принять решение о покупке);

2) подбор аналогов по изображению (например, пользователь может сфотографировать нужный ему предмет интерьера, аксессуар или элемент одежды, а мобильное предложение предложить ему доступный для покупки аналогичный товар);

3) отслеживать в режиме реального времени нахождение требуемых товаров в офлайн-магазинах или заказанных товаров в пути и точках выдачи онлайн-магазинов.

Приведем в качестве примера для демонстрации функциональных возможностей мобильное приложение старейшего российского универсального интернет-магазина OZON. Оформление страниц товарного каталога и корзины приведено на рис. 2.1.

Вкладки приложения расположены внизу экрана и предполагают возможность перехода на одну из пяти страниц приложения:

1) главная страница;

2) товарный каталог;

3) отобранные для заказа товары («Корзина»);

4) отобранные для последующих покупок товары («Избранное»);

5) личный кабинет пользователя («Мой Ozon»).

На главной странице мобильного приложения у пользователя есть возможность ознакомиться с проводимыми интернет-магазином актуальными акциями и предоставляемыми скидками, познакомиться с условиями продажи и доставки товаров, осуществить быстрый заказ наиболее востребованных из них.

На странице товарного каталога пользователь мобильного приложения может воспользоваться детально выполненным классификатором товаров, а также полнотекстовым поиском по каталогу, для подбора наиболее подходящего предложения.

Отобранные для заказа товары помещаются в «корзину», содержание которой в любой момент пользователь мобильного приложения может просмотреть и при желании изменить, дополнив её новыми товарами или удалив ранее в неё включенные. Функционал «корзины» также предусматривает возможность перехода к оформлению заказа.

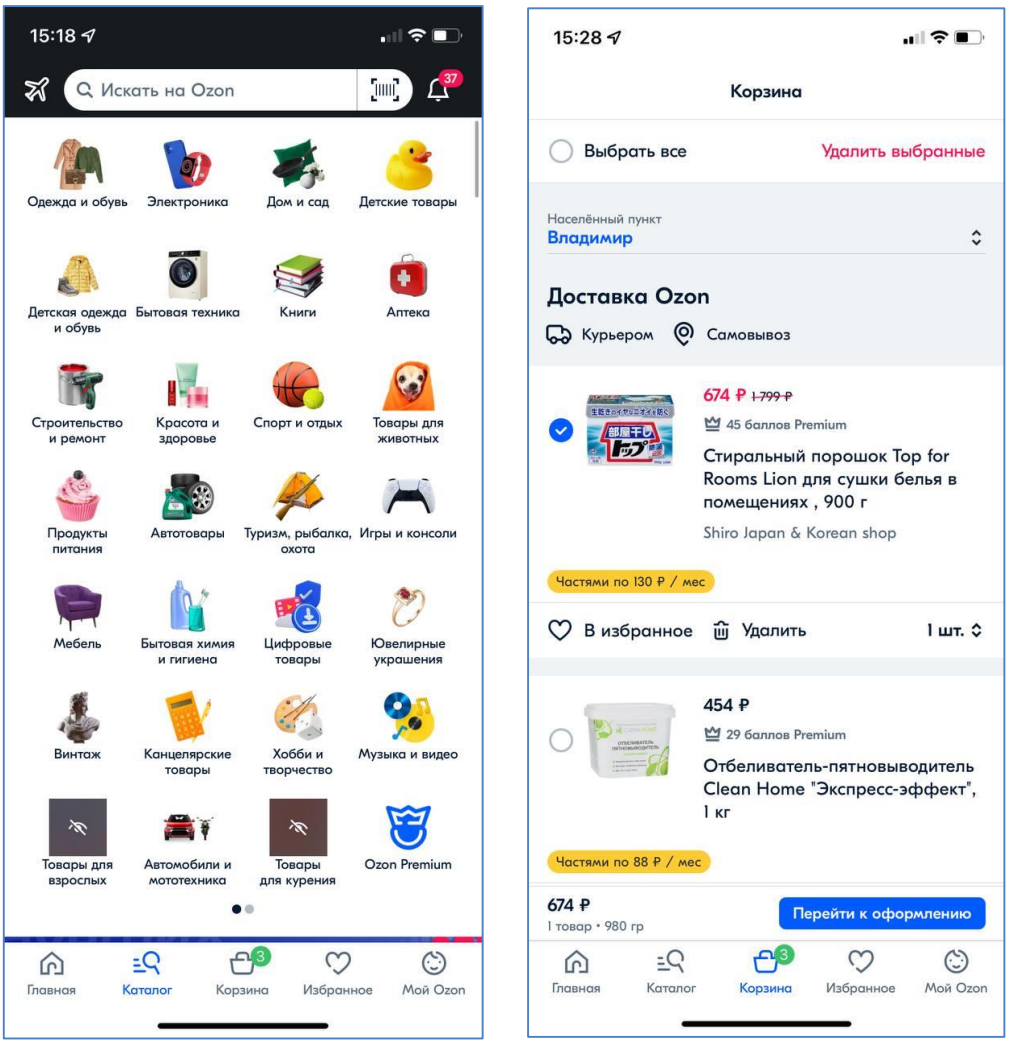


Рис. 2.1. Страницы товарного каталога и корзины в мобильном приложении интернет-магазина OZON

Личный кабинет позволяет пользователю работать с историей заказов, возвратов и оплат, производить сравнение товаров между собой, хранить личную и платежную информацию, копить и тратить бонусные баллы и многие другие полезные функции.

Кроме использования мобильных приложений для работы с потребителями с целью формирования дополнительного канала сбыта и формирования продуктивных коммуникаций с конечным потребителем, они находят широкое применение для обеспечения внутрифирменных коммуникаций и оперативного управления. В этом случае функциональные возможности приложения могут включать:

1) контроль за работой торговых представителей, пунктов выдачи и транспортных средств;

2) контроль текущих и перспективных остатков на складах;

3) контроль за расходами на осуществление отдельных операций и себестоимостью выполняемых заказов;

4) планирование коммуникаций с потребителем;

5) доступ к корпоративной базе знаний;

6) сканирование штрих-кодов.

Предприятия общественного питания используют мобильные приложения, главным образом, для сбыта своей продукции. Мобильные решения активно используются кафе и ресторанами для решения следующих задач:

1) предоставление гостям удобного сервиса по заказу посадочного места до прихода в заведение (электронное бронирование позволяет снизить нагрузку на администратора);

2) предоставление гостям удобного сервиса по выбору блюд (электронное меню позволяет полностью заменить бумажное, с одной стороны разгрузив официанта и сократив время на обслуживание, а с другой стороны – предложив гостю возможность получить больше информации о блюдах и их оценках другими гостями заведения);

3) информирование пользователей о проводимых заведением мероприятиях и акциях;

4) информирование гостей о скидках, бонусах и привилегиях, в т.ч. персональных;

5) интеграция с мобильными сервисами партнеров (например, со службами доставки еды и оказания услуг такси);

6) обеспечение обратной связи с гостями (например, оценка качества отдельных блюд и уровня обслуживания конкретным официантом);

7) обеспечение досуга гостей (например, использование технологии дополнительной реальности в мобильном приложении может позволит гостю участвовать в увлекательном квесте).

На фоне развития пандемии коронавируса толчок к развитию получил бизнес по доставке продуктов и готовой еды. Согласно прогнозам исследователей уже к 2023 году оборот дохода от приложений по доставке еды и продуктов может достигнуть отметки в 21 млрд долларов.

Главными факторами роста рынка доставки продуктов и готовой еды являются:

1) возрастающий уровень мобильности людей;

2) делегирование рутинных задач и тем самым освобождение собственного времени.

Мобильные приложения по доставке продуктов и готовой еды напоминают своими функциональными возможностями программные решения интернет-магазинов. В качестве ключевого отличия можно ответить возможность заказа одного и того же продукта или еды у разных поставщиков.

Приведем в качестве примера для демонстрации функциональных возможностей мобильное приложение доставки продуктов SBERMARKET. Оформление страниц выбора поставщика и товарного каталога приведено на рис. 2.2.

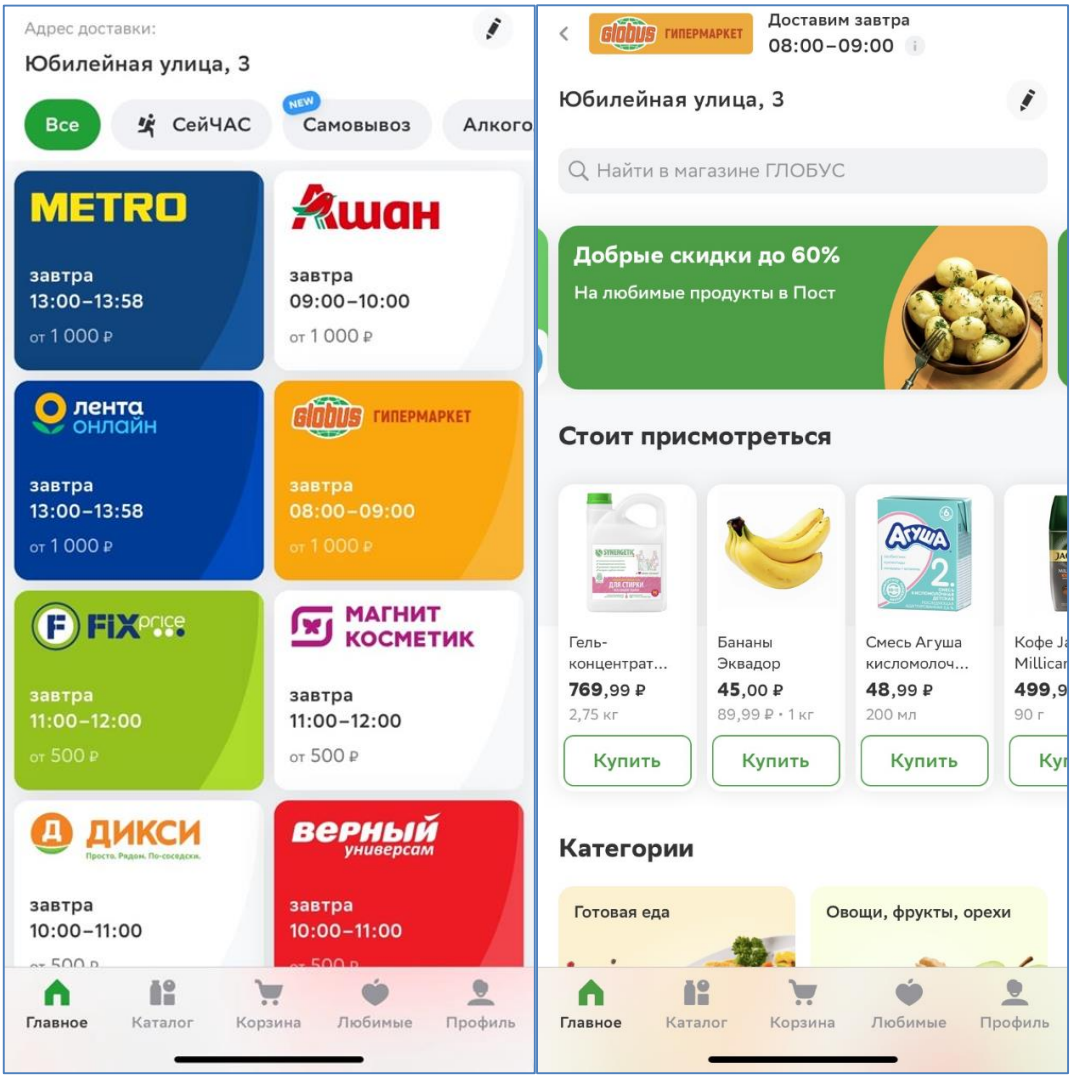


Рис. 2.2. Страницы выбора поставщика и товарного каталога в мобильном приложении сервиса SBERMARKET

Вкладки приложения расположены внизу экрана и предполагают возможность перехода на одну из пяти страниц приложения:

1) главная страница («Главное»);

2) товарный каталог («Каталог»);

3) отобранные для заказа товары («Корзина»);

4) отобранные для последующих покупок товары («Любимые»);

5) личный кабинет пользователя («Профиль»).

На главной странице мобильного приложения у пользователя есть возможность указать адрес доставки продуктов (без выполнения данного действия все остальные функции приложения будут не доступны), а также выбрать с использованием детального классификатора поставщика (магазина), из которого будет осуществляться доставка (без выполнения данного действия не будет доступен товарный каталог поставщика для заказа). В дальнейшем на главной странице будет отображаться:

1) информация с актуальными акциями и предоставляемыми скидками как поставщика, так и сервиса,

2) условия продажи и доставки товаров,

3) каталог доступных к заказу товарных позиций поставщика с возможностью быстрого заказа наиболее популярных из них.

На странице каталога пользователь мобильного приложения может воспользоваться детально выполненным классификатором товаров, а также полнотекстовым поиском по товарным предложениям для отбора для приобретения наиболее подходящих из них.

Отобранные для заказа товары помещаются в «корзину», содержание которой в любой момент пользователь мобильного приложения может просмотреть и при желании изменить, дополнив её новыми товарами или удалив ранее в неё включенные. Функционал «корзины» также предусматривает возможность перехода к оформлению заказа.

Личный кабинет позволяет пользователю работать с историей заказов, возвратов и оплат, производить сравнение товаров между собой, хранить личную и платежную информацию, копить и тратить бонусные баллы и многие другие полезные функции.

Личный кабинет позволяет пользователю работать с историей заказов, хранить личную и платежную информацию, копить и тратить бонусные баллы, воспользоваться чатом службы поддержки.

Функциональные возможности корпоративной части приложений по доставке товаров и еды включают следующий основной перечень вариантов использования применительно к рабочим местам работников [6]:

1) мобильное решения для курьера (расчет расстояния до клиента, расчет времени в пути, построение оптимального маршрута и т.д.);

2) при доставке еды мобильное решение для повара (мониторинг поступающих заказов на кухню и их исполнения, рецептура входящих в заказ отдельных блюд, калькуляция стоимости отдельных заказов и при необходимости фото и видео фиксация факта приготовления и/или готового блюда и т.д.);

3) мобильное решение для менеджера по продажам (введение клиентской базы, анализ криминальных действий пользователя, контроль оплаты и доставки заказа, составление отчетов и т.д.);

4) мобильное решение для менеджера склада (контроль прихода и расхода товара, контроль складских остатков, комплектация заказа и т.д.).

В сфере оказания туристических услуг также наблюдается рост использования мобильных предложений, что обусловлено следующими предпосылками [7]:

1) социальными (общим ростом потребления мобильных приложений в мире людьми разного возраста; популяризацией социальных приложений в мире; необходимостью в прямой и быстрой коммуникации между производителем и потребителями);

2) экономическими (ростом внутреннего и внешнего турпотока; ростом числа самодеятельных туристов; ростом прибытий зарубежных туристов; высокой конкуренцией между предприятиями туристской отрасли; поиск дополнительных каналов сбыта и продажи; популяризацией мобильной рекламы; мировыми тенденциями рынка мобильных приложений);

3) управленческими (внедрением и активным использованием CRM-систем; переориентацией на автоматизацию бизнес-процессов; необходимостью в большем делегировании полномочий с сохранением возможности дистанционного контроля у руководителя).

В сфере туризма широкое распространение получили мобильные решения в рамках следующих направлений:

1) поиск и приобретение авиабилетов;

2) поиск и бронирование мест размещения;

3) поиск и приобретение туристических путевок;

4) поиск информации об туристической инфраструктуре места прибывания (путеводители).

В качестве примера комплексного мобильного решения, в котором реализованы функциональные возможности по поиску и приобретению авиабилетов, поиску и бронированию отеля, а также поиску информации об туристической инфраструктуре приведем приложение AVIASALES. Оформление страниц выбора поставщика и товарного каталога приведено на рис. 3.3.

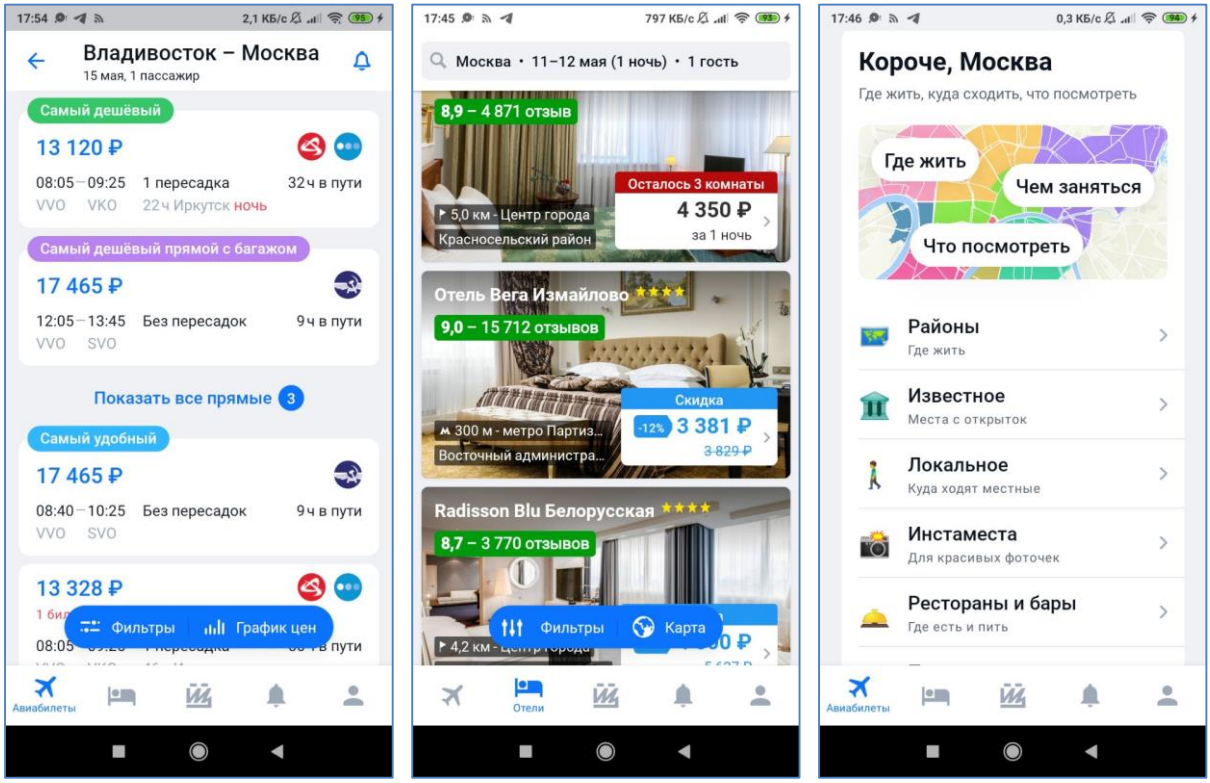


Рис. 2.3. Страницы поиска и заказа авиабилета, поиска и бронирования отеля, и страница путеводителя в приложении AVIASALES

Вкладки приложения расположены также внизу экрана и предполагают возможность перехода на одну из пяти страниц приложения:

1) страница поиска и заказа авиабилета («Авиабилеты»);

2) страница поиска и бронирования мест размещения в отеле («Каталог»);

3) подписка на дополнительный контент («Ещё»);

4) отслеживание цен на авиабилеты («Подписки»);

5) личный кабинет пользователя («Профиль»).

На странице поиска и заказа авиабилета пользователь мобильного приложения может воспользоваться поиском авиабилетов с учетом широкого набора параметров, в т.ч. если маршрут перелета будет сложным, а также осуществить заказ авиабилета на выбранный рейс.

На странице поиска и бронирования места в отеле пользователь мобильного приложения может воспользоваться поиском подходящего отеля и осуществить бронирование в нём места.

Отметим, что мобильные версии путеводителей по сравнению с традиционными бумажными, имеют ряд преимуществ:

1) более высокий уровень информативности;

2) наличия удобного поиска достопримечательностей;

3) более высокий уровень актуальности информации;

Мобильные приложения находят активное применение с сфере грузовых и пассажирских перевозок. Их широко применяют службы такси, грузоперевозок и логистических компаний.

Мобильные приложения для транспорта в первую очередь предназначены для решения следующих задач:

1) для заказчика: подбор подрядчика, способного оказать транспортные услуги по приемлемой стоимости и надлежащего качества;

2) для потребителя: подбор заказов на оказание транспортных услуг;

3) для владельца мобильного сервиса: извлечение экономической выгоды за счет установления коммуникаций между участниками сделки.

На функциональные возможности мобильного приложения влияет используемая схема взаимодействия между участниками, но в целом можно выделить следующие варианты использования:

1) для клиентов сервиса: а) поиск доступных для оказания транспортных услуг машин на определенной территории (районе, улице, регионе, государстве) по определённым критериям (возможность перевозки хрупкого груза, наличие кондиционера); б) ознакомление с ценами на услуги перевозчиков и доступными способами оплаты; в) расчет стоимости оказания транспортных услуг; г) доступ к информации о перевозчиках и водителях, в т.ч. к отзывам других заказчиков об их

работе; д) возможность оставлять отзыв о перевозчике и водителях относительно качества предоставляемых транспортных услуг;

2) для перевозчиков (водителей): а) получение уведомления о новых заявках; б) поиск и построение маршрутов с помощью картографических сервисов; в) получение статистики о доходах и расходах; г) установка тарифов на транспортные услуги;

3) для посредников: а) обеспечение коммуникаций между заказчиками и перевозчиками; б) контроль исполнения заказов; в) введение черных списков заказчиков и перевозчиков; г) получение статистики о доходах и расходах.

2.4. Использование мобильных приложений в финансовой сфере

Финансовые институты занимают второе место после ИТ-компаний по внедрению информационных технологий. При этом для повышения эффективности дистанционного обслуживания клиентов широко применяются мобильные решения [8].

Мобильный банкинг – управление банковским счетом с помощью, установленной на мобильное устройство клиента специального приложения.

По состоянию на середину 2020 года мобильными приложениями банков пользуются 51% россиян. Цифровой банкинг более востребован среди жителей российских столиц: его используют 57% жителей Москвы и Санкт-Петербурга. В сельской местности цифровые банковские сервисы менее распространены (49%).

Среди стандартных задач, которые решают пользователи приложения мобильного банкинга:

1) просмотр остатка денежных средств на банковских счетах;

2) просмотр движения по банковским счетам;

3) оповещение о поступлении и расходовании денежных средств на банковских счетах;

4) осуществление переводов между банковскими счетами;

5) оплата товаров, работ и услуг;

6) управление личными финансами;

7) обращение в техническую поддержку.

Возможности мобильных приложений большинства банков ограничены указанными стандартными функциями. Однако на рынке есть ряд банков-лидеров, которые предлагают пользователям своих мобильных решений расширенные функции [9]:

1) наличие текстового или голосового чата, с помощью которого можно проконсультироваться относительно использования приложения, а также получить разъяснения относительно предлагаемых банком услуг;

2) наличие расширенного набора способов реализации стандартных задач пользователя: оплата товаров и услуг по QR или штрихкоду, настройка шаблонов автоплатежей, подписок, безопасная привязка платежных реквизитов банковских карт и т.д.;

3) наличие информативной истории операций и лента предстоящих расходов;

4) наличие информации о лимитах банка с возможностью задать нужные пользователю значения (например, суточный персональный лимит на платежи и переводы, лимит с дополнительным подтверждением – по биометрии или звонком в банк, устанавливать лимит и ограничения по картам на покупки в интернете или снятие наличных);

5) возможность передачи показаний счетчиков и ежемесячного получения квитанции ЖКХ;

6) наличие интеграции с порталом государственных услуг с возможностью получения государственных уведомлений, выписок из трудовой книжки и об остатки материнского капитала и т.д.

7) возможность открытия, пополнения и закрытие банковского вклада, счетов эскроу, аккредитивов, металлически счета и т.д.;

8) возможность получить информацию о состоянии номинального счета;

9) осуществлять бесконтактную оплату.

В качестве примера банковского мобильного решения приведем приложение «Сбербанк онлайн». Оформление главной страницы, страница с информацией о дебетовых картах и кредитах пользователя приведено на рис. 2.4.

Для первого входа в приложение потребуется информация о карте Сбербанка. При технической возможности используемого мобильного устройства вход в приложение можно осуществлять по отпечатку пальца или изображению.

Вкладки приложения расположены внизу экрана и предполагают возможность перехода на одну из пяти страниц приложения:

1) главная страница («Главный»);

2) осуществление переводов и платежей («Платежи»);

3) общение со службой поддержки и клиентами банка («Диалоги»);

4) отслеживание операций, уведомлений и сообщения от банка («История»);

5) информация о всех продуктах банка («Каталог»).

На главной странице мобильного приложения у пользователя есть возможность получить информацию обо всех продуктах банка, познакомиться с новостями, советами и рекомендациями от банка, а также посмотреть общую статистику. На странице есть кнопки быстрого доступа к способам оплаты и входа в другие сервисы банка. Содержание данной страницы может быть настроено пользователем самостоятельно.

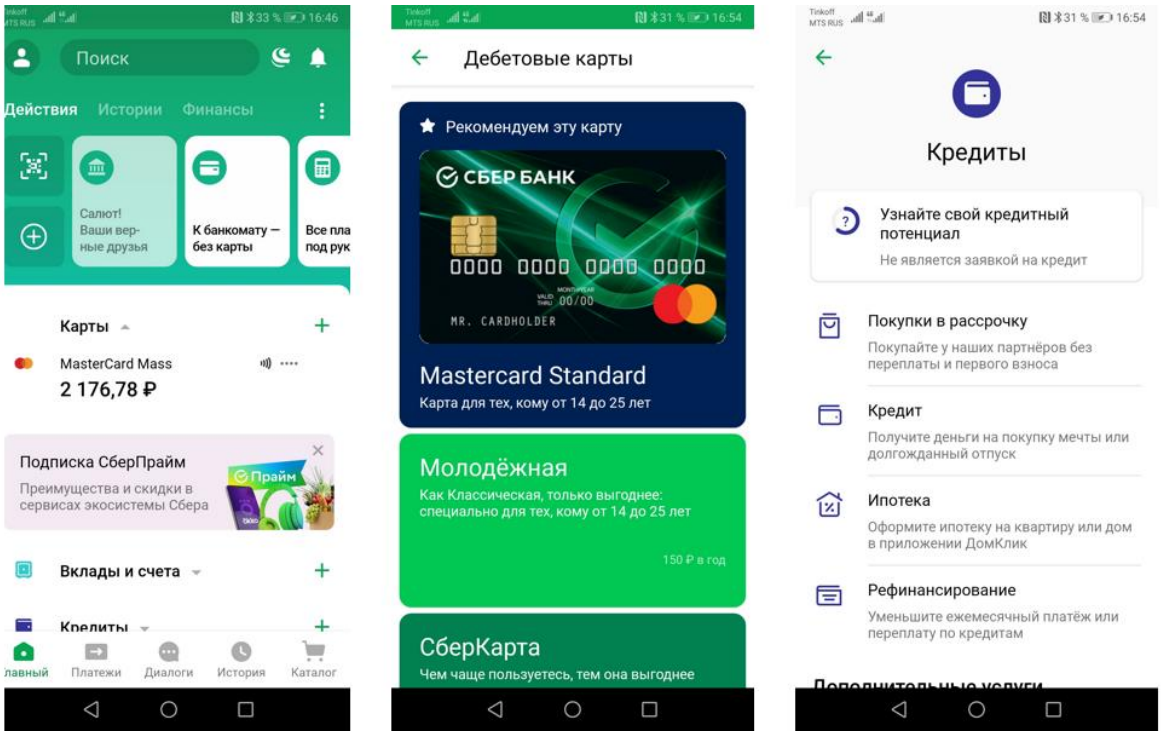


Рис. 2.4. Главная страница, страница с информацией о дебетовых картах и кредитах пользователя в приложении Сбербанк Онлайн

На странице осуществления переводов и платежей пользователь мобильного приложения банка может отправить денежные средства

деньги между своими счетами, а также другим людям по номеру телефона, номеру карты или по банковским реквизитам, в т.ч. с использованием системы быстрых платежей. На данной странице также расположена информация о всех выставленных счетах на оплату и настройки бесконтактной оплаты.

На странице «Диалоги» у пользователя есть возможность связаться со службой поддержки, подписаться на популярные тематические информационные каналы, общаться с другими клиентами банка в рамках индивидуальных или групповых чатов, совершать с ними простые сделки по переводу денежные средств или предоставления их в долг.

На странице «История» пользователь имеет возможность отслеживать совершенные банковские операции по картам и счетам, просматривать уведомления и сообщения от банка, заказывать банковские выписки. Содержание и оформление страницы «История» и связанных с ней страниц, приведено на рис. 2.5.

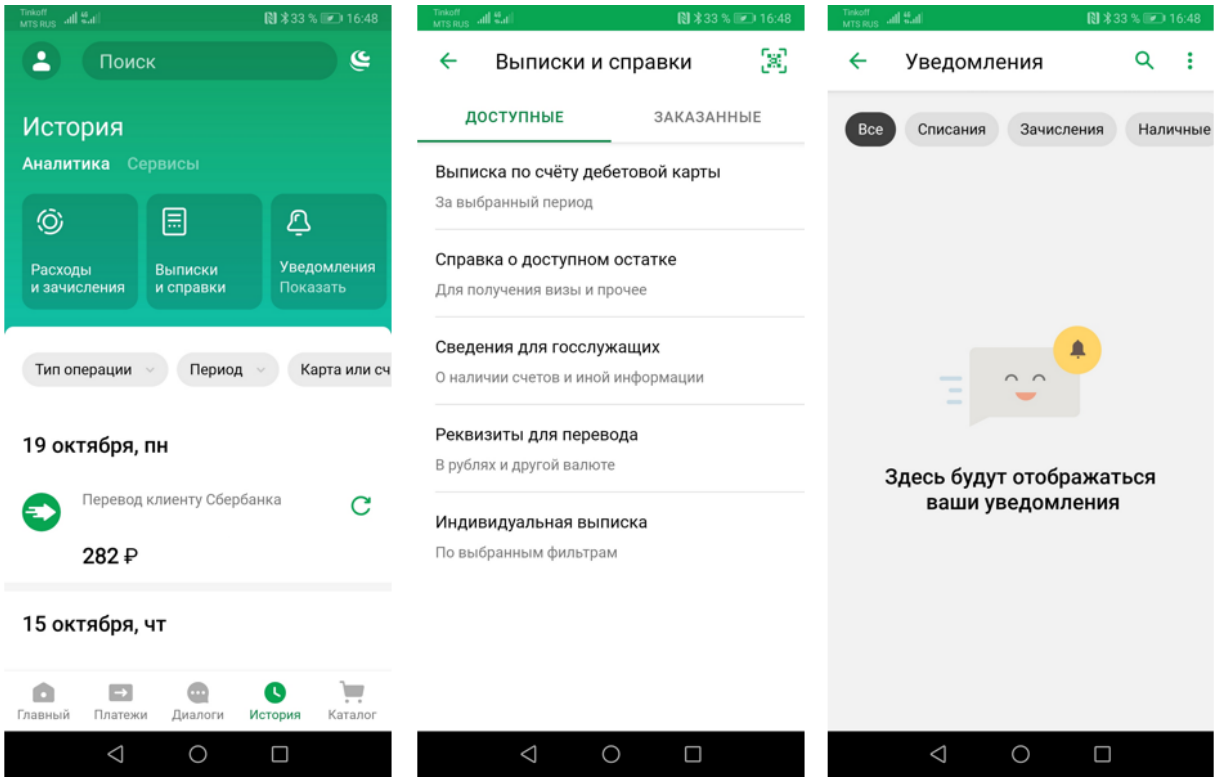


Рис. 2.5. Страница «История» и связанные с ней страницы в приложении Сбербанк Онлайн

По мере цифровизации экономики на рынке страхования наметилась тенденция к индивидуализации взаимоотношений между его участниками, которая обусловлена следующими факторами [10]:

1) значительный рост информации о клиентах и объектах страхования;

2) рост заявок на проведение андеррайтинговых процедур по заявке клиента;

3) активное распространение устройств для дистанционного сбора показателей жизни и здоровья.

Современные страховые мобильные приложения, как правило, предлагают своим потенциальным клиентам следующими функциональными возможностями:

1) предоставление информации о страховой компании, расположении её офисов и предлагаемых страховых продуктах;

2) предоставление информации о текущих акциях, скидках и индивидуальных предложениях;

3) обращение за консультацией относительно различных аспектов заключения договора страхования;

4) выбор страхового продукта, самостоятельное осуществление предварительного расчета его стоимости и оформление страхового полиса (в том случае если законодательством не установлено обязанность очного заключения договора страхования и/или осмотра объекта страхования).

Для клиентов с действующим страховым полисом в качестве дополнения к вышеперечисленным доступны следующие стандартные функциональные возможности мобильного решения [11]:

1) отслеживание состояния приобретённых полисов (в т.ч. настройка параметров напоминания о приближении срока окончания его действия);

2) продление или расторжение договора страхования;

3) дистанционное уведомление страховщика о страховом случае, его оформление и отслеживание статуса;

4) ознакомление с пошаговой инструкцией относительно порядка действий в конкретной ситуации;

5) обращение в службу поддержки относительно урегулирования возникающих проблем при исполнении договора страхования.

Пример страхового мобильного решения для клиента компании РЕСО (главная страница, страница личного кабинета клиента, страница с информацией о страховых полисах) приведен на рис. 2.6.

Мобильные решения позволяют оптимизировать ряд бизнес-процессов, в которых участвуют страховые агенты и работа которых связана с постоянными разъездами и встречами с клиентами, а именно:

1) проведение презентаций продуктов страховых компаний;

2) оценка имущества клиента;

3) расчет стоимости страхового полиса;

4) заполнение подробных отчётов о своей деятельности.

Соответственно, мобильное решение для страховых агентов должно обеспечить оперативный доступ к рекламным материалам по страховым продуктам компании, а также инструментам заключения и сопровождения сделок.

К необходимым для страхового агента функциональным возможностям специализированного мобильного приложения можно отнести:

1) удаленный доступ к корпоративным ресурсам компании (например, к аккумулированным в CRM системам данным, корпоративной электронной почте и другим внутренним каналам коммуникаций в компании);

2) демонстрация презентаций страховых услуг с учетом индивидуальных приоритетов потенциального клиента;

3) составления расписания встреч с клиентами и контроль за их осуществлением;

4) оперативный расчет стоимости страховых услуг на основе индивидуальных пожеланий клиента и порядка формирования стоимости услуг в компании;

5) заключение сделки на месте встречи;

6) работа с уведомлениями о наступлении страхового случая;

7) анализ предлагаемых на рассмотрение рисков с целью принятия решения о заключении договора страхования и разработка его условия (андеррайтинг).

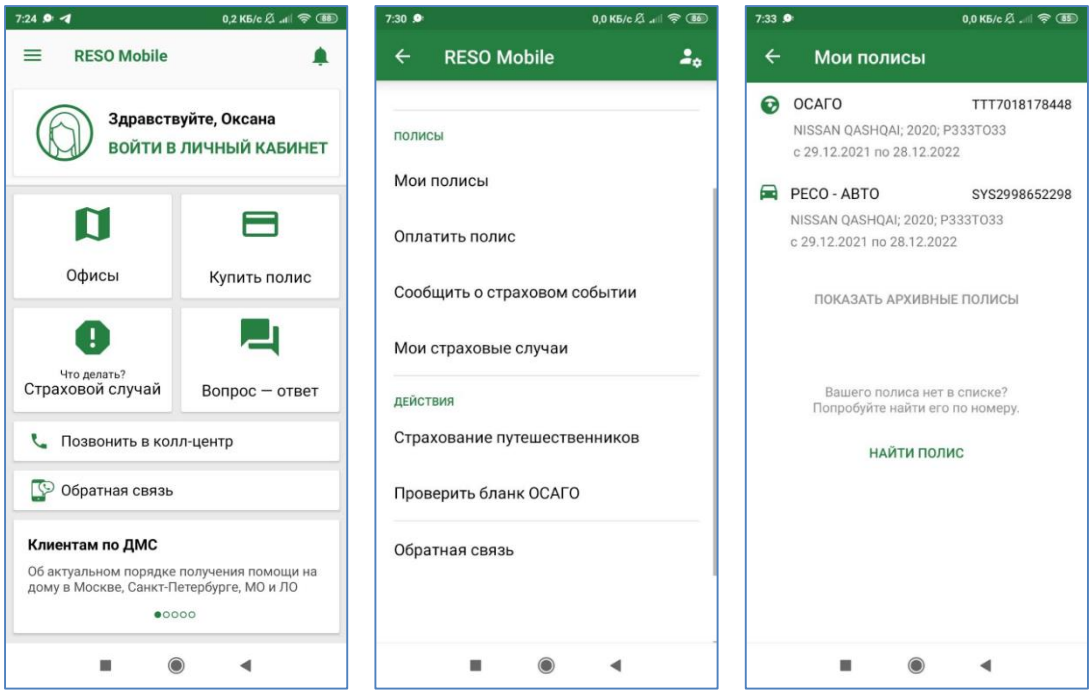


Рис. 2.6. Главная страница, страница личного кабинета, страница с информацией о страховых полисах в приложении RESO Mobile

В качестве примера мобильного решения для страховых агентов приведем приложение «PESO office». Оформление главной страницы, страницы личного кабинета страхового агента, страница с информацией о страховых полисах в данном приложении приведено на рис. 2.7.

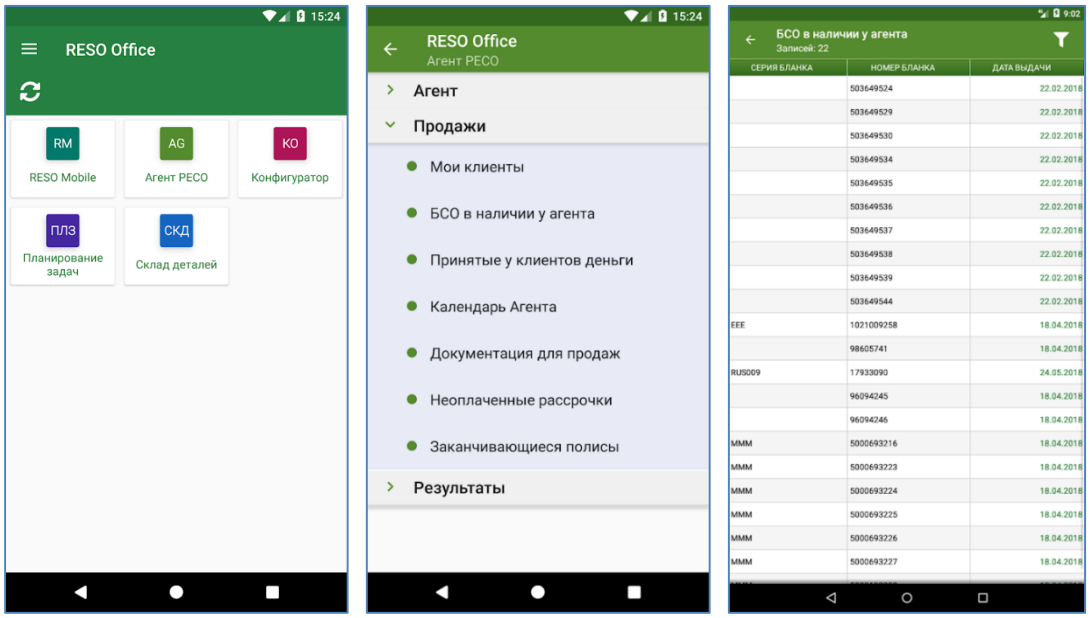


Рис. 2.7. Главная страница, страница личного кабинета страхового агента, страница с информацией о страховых полисах в приложении RESO Office

В данном приложении доступны различные модули корпоративной информационной системы, набор которых для каждого страхового агента зависит от его специализации:

1) модуль отслеживания пролонгации страховых полисов клиентами конкретного страхового агента с функцией напоминания;

2) модуль контроля оплаты страховых полисов клиентами конкретного страхового агента с функцией напоминания;

3) модуль корпоративного документооборота;

4) модуль начисленных вознаграждений;

5) модуль планирования и контроля встреч с функцией напоминания;

6) модуль актуальных маркетинговых материалов;

7) модуль доступных к заполнению бланков строгой отчетности (бумажных страховых полисов);

8) модуль формирования отчетов.

Мобильные решения находят также широкое применение и в других видах деятельности сферы услуг: в работе фитнес-центров и частных медицинских клиниках, в индустрии развлечений, при проведении виртуальных медицинских консультаций и т.д., а также в социальной сфере.

Вопросы для обсуждения

1. Задачи в промышленности, решаемые с помощью мобильных приложений.

2. Синергетический эффект использование мобильных приложений в связки с другими технологиями

3. Задачи в строительстве, решаемые с помощью мобильных приложений.

4. Выгоды использования мобильных технологий участниками торговых сделок.

5. Текущие и перспективные функциональные возможности мобильных приложений розничной Интернет-торговли.

6. Области применения технологий дополнительной реальности в мобильных приложениях розничной Интернет-торговли

7. Текущие и перспективные функциональные возможности мобильных приложений сервисов по доставке товаров и еды.

8. Текущие и перспективные функциональные возможности мобильных приложений предприятий общественного питания.

9. Текущие и перспективные функциональные возможности мобильных приложений в сфере туризма.

10. Текущие и перспективные функциональные возможности мобильных приложений сервисов по приобретению авиабилетов?

11. Текущие и перспективные функциональные возможности мобильных приложений сервисов по бронированию мест размещения

12. Текущие и перспективные функциональные возможности мобильных приложений сервисов по продажи туристических путевок

13. Текущие и перспективные функциональные возможности мобильных приложений сервисов поиска информации об туристической инфраструктуре места прибывания

14. Задачи на транспорте, решаемые с помощью мобильных приложений.

15. Текущие и перспективные функциональные возможности мобильных приложений в области оказания банковских услуг.

16. Текущие и перспективные функциональные возможности мобильных приложений в области страхования

Для решения каких задач предназначены приложения мобильного банкинга?

16. Функциональными возможностями обладают мобильные приложения страховых компаний?

Практические задания

Задание 2.1.

Определите несколько (не менее трех) популярных приложений, которые используются в заданных видах хозяйственной деятельности (по вариантам).

Определите задачи, которые данные мобильные приложения решают, и сравните их функциональные возможности.

Варианты видов деятельности:

1) банковская деятельность;

2) рекламная деятельность;

3) риэлтерская деятельность;

4) оказание услуг общественного питания;

5) розничная торговля;

6) сельское хозяйство;

7) транспортные услуги;

8) туристические услуги;

9) услуги страхования;

10) бытовые услуги;

11) страховая деятельность;

12) доставка продуктов;

13) образовательная деятельность.

Задание 2.2.

Сформулируйте задачи, которое должно решать мобильное приложение предприятия, которое ведет заданный вид хозяйственной деятельности (по вариантам). Предприятие выбрать самостоятельно.

Варианты видов деятельности:

1) банковская деятельность;

2) рекламная деятельность;

3) риэлтерская деятельность;

4) оказание услуг общественного питания;

5) розничная торговля;

6) сельское хозяйство;

7) транспортные услуги;

8) туристические услуги;

9) услуги страхования;

10) бытовые услуги;

11) страховая деятельность;

12) доставка продуктов;

13) образовательная деятельность.

Библиографический список

1. Мартиросян, К. В. Интернет-технологии : учеб. пособие / К. В. Мартиросян, В. В. Мишин. – Ставрополь : СКФУ, 2015. - 106 с.

2. Цифровой бизнес : учебник / под науч. ред. О. В. Китовой. – М. : ИНФРА-М, 2021. – 418 с. – (Высшее образование: Магистратура). – ISBN 978-5-16-013017-0. – Текст : электронный. – URL: https://znanium.com/catalog/product/1659834 (дата обращения: 04.05.2022)

3. Цифровая экономика / Н. В. Абдуллаев, Н. Л. Аванесян, Д. В. Виноградов [и др.]. – Москва : Компания КноРус, 2018. – 286 с. – ISBN 978-5-4365-3040-6. – EDN YPRLET.

4. Курчеева, Г. И. Информационное и программное обеспечение электронного бизнеса : учеб. пособие / Г. И. Курчеева, М. А. Бакаев, В. А. Хворостов. - Новосибирск : Изд-во НГТУ, 2018. - 107 с. - ISBN 978-5-7782-3500-7. - Текст : электронный. - URL: https://znanium.com/catalog/product/1866896 (дата обращения: 05.05.2022).

5. Мобильные технологии в промышленности // НИИ СОКБ: портал [Электронный ресурс]. – Режим доступа: https://www.niisokb.ru/mobilnye-tehnologii-v-promyshlennosti

6. Партолина П. С., Широбокова С. Н. Анализ функциональной полноты программных систем для автоматизации служб доставки еды // Молодой исследователь Дона. 2021. №4 (31). URL: https://cyberleninka.ru/article/n/analiz-funktsionalnoy-polnoty-programmnyh-sistem-dlya-avtomatizatsii-sluzhb-dostavki-edy (дата обращения: 10.05.2022).

7. Скоробогатова Т. Н., Павленко И. Г., Килина А. С., Кендзерская Н. В. Предпосылки формирования мобильных приложений как средства продаж туристских услуг // Геополитика и экогео-динамика регионов. 2021. №2. URL: https://cyberleninka.ru/article/n/predposylki-formirovaniya-mobilnyh-prilozheniy-kak-sredstva-prodazh-turistskih-uslug (дата обращения: 10.05.2022).

8. Батаев А. В. Внедрение мобильного банкинга в финансовых институтах России // Наука, техника и образование. 2016. № 8 (26). URL: https://cyberleninka.ru/article/n/vnedrenie-mobilnogo-bankinga-v-finansovyh-institutah-rossii (дата обращения: 10.05.2022).

9. Барыло Е. В., Шакирова К. В., Зяблицкая Н. В. Развитие дистанционного банковского обслуживания. сравнительная характеристика интернет-банкинга и мобильного банкинга // РППЭ. 2020. №6 (116). URL: https://cyberleninka.ru/article/n/razvitie-distantsionnogo-bankovskogo-obsluzhivaniya-sravnitelnaya-harakteristika-internet-bankinga-i-mobilnogo-bankinga (дата обращения: 10.05.2022).

10. Магзумова Н. В., Кузнецов С. М. Цифровое страхование: современное состояние и перспективы развития // Инновационная экономика: перспективы развития и совершенствования. 2021. №3 (53). URL: https://cyberleninka.ru/article/n/tsifrovoe-strahovanie-sovremennoe-sostoyanie-i-perspektivy-razvitiya (дата обращения: 11.05.2022).

11. Бондаренко В. В. Цифровые каналы распространения страховых полисов // Хроноэкономика. 2020. №4 (25). URL: https://cyberleninka.ru/article/n/tsifrovye-kanaly-rasprostraneniya-strahovyh-polisov (дата обращения: 11.05.2022).

Глава 3. ЖИЗНЕННЫЙ ЦИКЛ МОБИЛЬНЫХ ПРИЛОЖЕНИЙ

В главе рассматриваются следующие вопросы:

1. Понятие жизненного цикла мобильного приложения;
2. Формирование замысла мобильного приложения;
3. Разработка мобильного приложения;
4. Внедрение, применение и списание мобильного приложения.

3.1. Понятие жизненного цикла мобильного приложения

Мобильное приложение является, как правило, одним из элементов информационной системы предприятия, т.е. системы сбора, обработки, поиска, хранения, передачи и предоставления данных, которая представляет собой совокупность программного, аппаратного и организационного обеспечения, а также методологических и информационных ресурсов.

Такие информационные системы имеют, как правило, трехуровневую архитектуру, в рамках которой мобильное приложение используется как клиент, вся логика работы которого реализуется на сервере приложений, а данные хранятся на сервере баз данных. Шаблон архитектуры простой трехуровневой информационной системы с использованием мобильного приложения в качестве клиента на рис. 3.1.

Рис. 3.1. Шаблон архитектуры простой трехуровневой информационной системы с использованием мобильного решения

ГОСТ Р ИСО/МЭК 12207-2010 устанавливает строгую связь между информационной системой и применяемыми в ней программными средствами: «программное средство трактуется как единая часть

общей системы, выполняющая определенные функции в данной системе, что осуществляется посредством выделения требований к программным средствам из требований к системе, проектирования, производства программных средств и объединения их в систему» [1].

В процессе своего развития от момента осознания потребности и до момента прекращения существования как единого целого любой создаваемый человеком объект (предприятие, направление хозяйственной деятельности (бизнес), технология, система, проект, продукция и т.д.) проходит ряд состояний:

1) осознание потребности в объекте;

2) наличие жизнеспособного решения (замысла, концепции) объекта;

3) наличие проекта объекта;

4) наличие созданного объекта;

5) наличие исчерпавшего свой потенциал объекта;

6) прекращение существования объекта как единого целого.

Интервал времени между осознанием потребности в объекте и моментом прекращения его существования как единого целого называют жизненным циклом объекта. Соответственно, интервал времени, относящийся к определенному состоянию реализации объекта принято называть стадией его жизненного цикла.

Применительно к мобильному приложению жизненный цикл – непрерывный процесс, началом которого становится момент принятия решения о необходимости мобильного решения, а завершением –изъятие его из эксплуатации [2].

При этом жизненный цикл мобильного приложения не существует изолировано и является составляющей жизненного цикла информационной системы, в рамках которой оно используется. В свою очередь жизненный цикл информационной системы определяется жизненным циклом используемых в ней информационных технологий, а тот в свою очередь жизненным циклом бизнеса, в котором данные технологии применяются. Общее представление концепция уровней жизненного цикла мобильного приложения приведена на рис. 3.2.

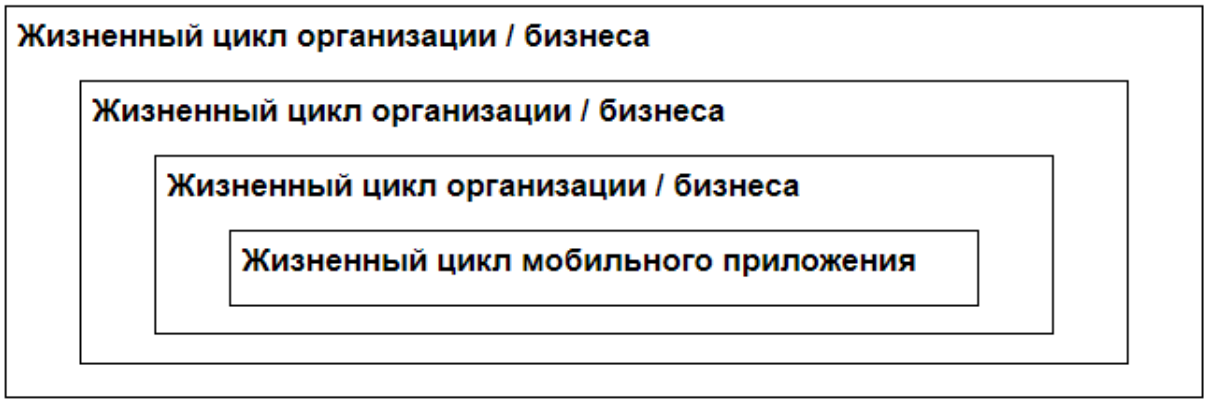


Рис. 3.2. Концепция уровней жизненного цикла мобильного приложения

В ГОСТ Р ИСО/МЭК 12207-2010 [1] совокупность выполняемых в рамках жизненного цикла программного обеспечения взаимосвязанных или взаимодействующих видов деятельности, преобразующих входы в выходы, называют процессами.

Выделяют следующие категории процессов жизненного цикла программного обеспечения:

1) процессы в контексте информационной системы (соглашения, менеджмента и поддержки проекта, технические, организационного обеспечения проекта);

2) специальные процессы программных средств (реализации, поддержки, повторного применения программных средств).

Группа процессов соглашения (процессы в контексте информационной системы) включают процессы приобретения и поставки продуктов, используемых в информационной системе. Процессы данной группа определяют действия, необходимые для выработки соглашений между участниками проекта разработки и внедрения информационной системы.

Процессы группы организационного обеспечения проекта (процессы в контексте информационной системы) обеспечивают ресурсы и инфраструктуру, необходимые для поддержки проекта разработки и внедрения информационной системы. Данная группа процессов включает следующие процессы:

1) процесс менеджмента модели жизненного цикла;

2) процесс менеджмента инфраструктуры;

3) процесс менеджмента портфеля проектов;

4) процесс менеджмента людских ресурсов;

5) процесс менеджмента качества.

Процессы менеджмента и поддержки проекта (процессы в контексте информационной системы) используются для планирования, выполнения, оценки и управления продвижением проекта, а также формируют специфическую совокупность задач, ориентированных на выполнение специальных целей менеджмента. Данная группа включает следующие процессы:

1) процесс планирования проекта;
2) процесс управления и оценки проекта;
3) процесс менеджмента решений;
4) процесс менеджмента рисков;
5) процесс менеджмента конфигурации;
6) процесс менеджмента информации;
7) процесс измерений.

Технические процессы (процессы в контексте информационной системы) предоставляют возможность продуктам и услугам соответствовать ожиданиям или требованиям потребителя. Данная группа включает следующие процессы:

1) определение требований правообладателей;
2) анализ системных требований;
3) проектирование архитектуры системы;
4) процесс реализации;
5) процесс комплексирования системы;
6) процесс квалификационного тестирования системы;
7) процесс инсталляции программных средств;
8) процесс поддержки приемки программных средств;
9 процесс функционирования программных средств;
10) процесс сопровождения программных средств;
11) процесс изъятия из обращения программных средств.

Следует отметить, что указанные технические процессы, несмотря на схожесть с процессами реализации программных средств, являются специальными случаями одноименных процессов жизненного цикла информационных систем и рассматриваются как их компонент.

Процессы реализации программных средств (специальные процессы программных средств) используются для создания в виде программного средства конкретного элемента информационной системы.

Данная группа включает следующие процессы:

1) процесс анализа требований к программным средствам;

2) процесс проектирования архитектуры программных средств;

3) процесс детального проектирования программных средств;

4) процесс конструирования программных средств;

5) процесс комплексирования программных средств;

6) процесс квалификационного тестирования программных средств.

Группа процессов поддержки программных средств (специальные процессы программных средств) включает:

1) процесс менеджмента документации программных средств;

2) процесс менеджмента конфигурации программных средств;

3) процесс обеспечения гарантии качества программных средств;

4) процесс верификации программных средств;

5) процесс валидации программных средств;

6) процесс ревизии программных средств;

7) процесс аудита программных средств;

8) процесс решения проблем в программных средствах.

Группа процессов повторного применения программных средств (специальные процессы программных средств) обеспечивают возможность повторно использования компонентов программных средств в других проектах и включает следующие процессы:

1) процесс проектирования доменов;

2) процесс менеджмента повторного применения активов;

3) процесс менеджмента повторного применения программ.

Лицо ответственное за управление жизненным циклом мобильного приложения может использовать для каждой стадии уникальный набор процессов, который наиболее подходящим образом отвечает его специфики.

В [3] типичные стадии жизненного цикла системы включают: замысел, разработку, производство, применение, поддержку и выведение из эксплуатации.

Применительно к приложениям для мобильных устройств данные стадии могут быть уточнены следующим образом:

1) формирование замысла мобильного решения в контексте его использования в рамках информационной системы;

2) разработка (в т.ч. анализ и постановка задачи, проектирование, конструирование, комплексирование и тестирование) мобильного решения;

3) развертывание и внедрение мобильного решения (в т.ч. при необходимости публикация приложения на электронных площадках агрегаторов);

4) применение мобильного решения и его поддержка;

5) прекращения применения мобильного решения и его списание.

Рассмотрим данные стадии более подробно.

3.2. Формирование замысла мобильного приложения

С момента осознания потребности в создании нового или модификации существующего мобильного приложения, начинается первая стадия его жизненного цикла – стадия замысла.

На данной стадии формируется одно или несколько альтернативных концептуальных решений по созданию нового или модификации существующего мобильного решения и осуществляется оценка их экономических, технических, стратегических и рыночных характеристик.

Основная задача данного этапа – выбор одного наиболее жизнеспособного варианта концептуального решения для его дальнейшей практической реализации. Следует отметить, что на данном этапе формируется нисколько замысел непосредственно мобильного приложения сколько замысел всей информационной системы, в которую мобильное приложение входит в качестве компонента.

Формирование замысла мобильного решения предполагает выполнение нескольких этапов:

1) сбор и анализ бизнес-требований;

2) создание образа решения;

3) определение содержания проекта.

Цель этой работы – определить основные требования бизнеса (исходные данные, истинные цели, которым должно служить информационная система в целом и мобильное приложение в частности, а также проблемы, которые нужно преодолеть) [4].

К основным задачам, выполняемых на этапе сбора требований бизнеса относят:

1) определение исходных стимулов правообладателей;

2) определение целей мобильного приложения и критериев успеха.

Как правило, стимулом к созданию или модернизации мобильного приложения может служить одна или несколько из нижеперечисленных групп проблем и благоприятных возможностей будущих правообладателей:

1) потребность рынка (например, рост количества заказывающих через мобильное приложение продукцию потребителей);

2) производственная необходимость (например, обеспечение интеграции технологических и бизнес-процессов предприятия в партнерские сети);

3) потребности менеджмента (например, обеспечение высокого уровня координации работников при выполнении производственных задач);

4) технический прогресс (например, выход новой версии операционной системы мобильного устройства);

5) юридические ограничения или нормы (например, требования государства к обеспечению высокого уровня информационной безопасности).

При разработке мобильного приложения необходимо идентифицировать конкретные проблемы и благоприятные возможности, связанные с созданием или модернизацией мобильного приложения, а также дать их развернутую характеристику.

Так как единовременное решение всех проблем и реализация всех возможностей в силу ограниченности доступных экономических ресурсов может оказаться невозможным, то с целью определения приоритетов следует оценить уровень значимости каждой из них.

На основе определенного списка приоритетных к решению задач осуществляют определение целей, которые ставятся перед мобильным приложением со стороны правообладателей.

Определяемые цели должны быть измеряемыми и давать полную характеристику тех преимуществ, которые дает мобильное приложение правообладателям, а также должны содержать точные и однозначные критерии её достижения (успеха). Например, цели мобильного приложения и критериев успеха могут быть сформулированы следующим образом: «проектируемое мобильное приложение на фоне сниже-

ния объёмов продаж через интернет-магазин (проблема) позволит увеличить количество продаж (цель) с 1500 до 3000 заказов в месяц через полгода после внедрения (критерий успеха)».

Образ мобильного приложения – принимаемое всеми участниками жизненного цикла видение создаваемого результата (программного решения).

Формулировка образа мобильного решения должна быть лаконичной и при этом содержать все основные сведения о нём. Например, в работе [5] предлагается для описания образа любого выводимого на рынок продукта использовать специальный шаблон, которые применительно к мобильным приложениям может выглядеть следующим образом:

1) целевая аудитория пользователей мобильного приложения или сервиса – «для»;

2) решаемые проблемы пользователя или предоставляемые возможности – «который»;

3) имя мобильного приложения или сервиса – «который»;

4) категория мобильного приложения или сервиса – «является»;

5) ключевое преимущество мобильного приложения или сервиса – «позволяет»;

6) основное конкурирующее мобильное приложение или сервис – «в отличие от»;

7) основное отличие мобильного приложения или сервиса – «за счет».

Формулировка образа мобильного решения по данному шаблону может выглядеть следующим образом: «Для домохозяек со среднем достатком, которые испытывают проблемы с оперативным приобретением ингредиентов для домашней выпечки, предлагаемое «Наше решение», являясь мобильным приложением сервиса по доставке продуктов питания, позволяет недорого и быстро получить нужные компоненты в отличии от своего основного конкурента службы «Яндекс. Еда» за счет простоты и удобства оформления заказа, а также системы индивидуального поощрения».

После определения образа мобильного приложения осуществляется определение содержания проекта по его разработке, внедрению и использованию, а именно:

1) осуществляется предварительная оценка необходимых для осуществления проекта ресурсов;

2) осуществляется предварительная оценка необходимого для осуществления проекта времени;

3) осуществляется предварительная оценка экономической эффективности альтернативных вариантов осуществления проекта (например, основанных на разных образах мобильного приложения) и выбор наилучшего из них с финансовой точки зрения.

Основным используемым на данной стадии инструментарием являются: экспресс-обследование; стратегический анализ замысла мобильного решения (SWOT и PEST-анализы, TELOS-анализ); анализ технической реализуемости и инновационного потенциала проекта по разработке и внедрению мобильного решения; правовой анализ проекта; организационный анализ проекта; коммерческий анализ проекта; финансово-экономический анализ проекта (в т.ч. анализ финансовых рисков).

Обследование проводится с целью сбора информации о целях и задачах компании в целом и внедряемого мобильного решения в частности, а также с целью получения данных об основных бизнес-процессах и покрывающем их ландшафте информационных систем (для определения в нем места внедряемого решения) [2].

Различают две категории обследования: экспресс-обследование и полное информационное обследование. На стадии замысла проводится экспресс-обследование (в виду своей низкой стоимости и трудоёмкости), а полное информационное обследование проводится на стадии разработки (после принятия решения о разработке и внедрении конкретного мобильного решения).

Цели экспресс-обследования:

1) выявление новых возможностей, повышающих эффективность бизнеса правообладателя;

2) выявление базовой линии требований правообладателя и предварительных требований к информационной системе.

Результаты экспресс-обследования:

1) рекомендации по повышению эффективности бизнеса;

2) рекомендации по выбору программного обеспечения для создания мобильного решения с учетом его стоимости и условий приобретения;

3) рекомендации и технические требования к серверам и мобильным устройствам пользователей;

4) этапы и сроки выполняемых работ в рамках проекта (создание аппаратной инфраструктуры, установка программного обеспечения, настройка программы, создание пользователей, настройка интерфейсов и прав доступа, перенос данных из других систем, разработка дополнительного функционала, обучение сотрудников, ввод в промышленную эксплуатацию и т.д.);

5) объем и стоимость выполняемых работ.

Основными методами экспресс-обследования являются анкетирование и интервьюирование.

Метод анкетирования – вербально-коммуникативный метод, в котором в качестве средства для сбора сведений от респондента используется специально оформленный список вопросов – анкета. Анкетирование – опрос при помощи анкеты.

Различают очное (проводится в присутствии лица, которое проводит экспресс-обследование) и заочное анкетирование.

Метод интервью – вербально-коммуникативный метод, в котором в качестве средства для сбора сведений от респондента используется разговор по заранее разработанному плану.

При экспресс-обследовании используется предварительное интервью. Основное интервью используется при полном информационном обследовании, контрольное интервью – для проверки спорных результатов.

По результатам экспресс-обследования составляется отчет, который описывает следующие направления [2]:

1) краткая характеристика объекта исследования, основные виды деятельности;

2) схема и краткое описание организационной структуры предприятия;

3) основные функции обследуемых структурных подразделений с группировкой по бизнес-процессам;

4) краткое описание существующих на предприятии бизнес-процессов, принципов взаимодействия между его подразделениями;

5) описание информационного взаимодействия подразделениями;

6) описание основных проблем и слабых мест на обследуемых участках бизнес-процессов;

7) пожелания заказчика по совершенствованию системы управления предприятием;

8) предложения о дальнейшем сотрудничестве, рекомендации по решению выявленных проблем и ориентировочная стоимость работ.

Цель стратегического анализа проекта заключается в анализе соответствия целей проекта стратегии развития компании. В ходе стратегического анализа выявляются стратегические риски проекта, которые могут быть связаны с изменениями внешней среды проекта (макроокружения и микроокружения), изменениями внутренней среды.

Классификация основных методов стратегического анализа приведена на рис. 3.3.

Рис. 3.3. Основные методы стратегического анализа проектов

К ситуационному анализу также относят оценку целесообразности проекта (TELOS). Акроним TELOS содержит основные аспекты, которые требуют рассмотрения для того, чтобы удостоверится в его реалистичности и достаточном для реализации потенциале:

1) технические (Technological);

2) экономические (Economical);

3) юридические (Legal);

4) операционные (Operatoinal);

5) сроки реализации (Scheduling).

Цель технического анализа проекта заключается в оценке технической реализуемости проекта и его инновационном потенциале. Оценка реализуемости проекта проводится с целью определения осуществимости заложенных в проект научных, конструкторско-технологических решений, наличия соответствующих зарубежных или отечественных аналогов.

Цель коммерческого (маркетингового) анализа проекта заключается в определении общей коммерческой результативности проекта.

Основные методы коммерческого анализа:

1) оценка конкурентоспособности продукции проекта;

2) прогноз объемов продаж выпускаемой продукции (услуг);

3) разработка политика ценообразования;

4) разработка мероприятий по продвижению продукции и т.д.

Цель организационного анализа проекта заключается в оценке руководящего состава проекта, его потенциала и способность успешно реализовать проект.

Проводится анализ организационной структуры компании, уровень централизации (децентрализации) управления, системы менеджмента. Также анализируется способность управленческого персонала осуществлять эффективное руководство предприятием в целом.

Цель финансово-экономического анализа проекта: оценить экономическую эффективность проекта и выбрать из набора альтернативных вариантов наиболее эффективный проект.

Основные методы:

1) анализ точки безубыточности проекта;

2) анализ запас финансовой прочности предприятия;

3) определение простых и дисконтированных показателей экономической эффективности (абсолютных, относительных и временных);

4) определение внутренней нормы доходности проекта;

5) анализ финансовых рисков, включая анализ чувствительности и стресс-тестирование, моделирование воздействия рисков на операционные потоки проекта с учетом волатильности воздействующих условий и факторов и др.

Таким образом, стадия формирования замысла мобильного приложения заключается в выполнении оценки новых возможностей в деловой сфере, а также в разработке предварительных системных требований и осуществимых мобильных решений.

Так как возможностей в деловой сфере может быть несколько, то зачастую реализовать все замыслы в рамках одной информационной системы не предоставляется возможным. Поэтому формируется несколько разумных, возможных и законных альтернативных вариантов замысла, из которых выбирается наиболее эффективный проект со стратегической, технической, коммерческой, организационной, финансово-экономической точки зрения.

3.3. Разработка мобильного приложения

Процесс разработки (реализации) предполагает создание программной части информационной системы в виде мобильного приложения, которая с одной стороны должна удовлетворять требованиям к архитектурным решениям информационной системы, а с другой стороны соответствовать требованиям заинтересованных в её создании лиц.

В первую очередь в рамках данного процесса должна быть определена стратегия разработки мобильного приложения, выбор которой определяется сложностью и масштабами проекта.

В настоящее время широко используют следующие базовые стратегии разработки мобильных приложений [6]: каскадная, инкрементная и эволюционная, которые отличаются между собой способом прохода по основным этапам создания программной части информационной системы (каскадная стратегия предполагает однократный проход без уточнения первоначально установленных требований, инкрементная – многократный проход без уточнения первоначально установленных требований с запланированным улучшением результата, эволюционная – многократный проход с уточнением требований).

Каскадная стратегия разработки мобильных приложений обладает следующими достоинствами:

1) простота применения (стабильность требований к программному решению в процессе всех этапов его разработки позволяет добиться невысокой сложности планирования, контроля и управления проектом);

2) высокое качество результата при отсутствии жестких временных и бюджетных ограничений;

3) доступность для понимания всем участникам проекта.

К недостаткам каскадной стратегии разработки мобильных приложений относят:

1) высокая сложность разработки требований (требования формулируются на начальных стадиях и в дальнейшем не подлежат корректировке, что предполагает высокий уровень их проработки);

2) невозможность использования промежуточных состояний программного решения (мобильное приложение полностью не работоспособно до тех пор, пока оно не пройдет все этапы жизненного цикла разработки);

3) низкий уровень вовлечения заинтересованных лиц к процессу разработки программного решения (интересы стейкхолдеров учитываются только при разработке требований и не уточняются в процессе реализации мобильного приложения).

Инкрементная стратегия разработки мобильных приложений обладает следующими достоинствами:

1) возможность использования промежуточных состояний программного решения снижает риск неудачи (после каждого цикла (инкремента) разработки с разным уровнем детализации учтенных в решении требований мобильное приложение является частично или полностью работоспособным);

2) вовлечение заинтересованных лиц к процессу разработки программного решения (стейкхолдерам предоставляется возможность апробировать прототип мобильного решения).

К недостаткам инкрементной стратегии разработки мобильных приложений относят:

1) высокая сложность разработки требований (требования формулируются на начальных стадиях и в дальнейшем не подлежат уточнению, что предполагает высокий уровень их проработки);

2) сложность применения (стабильность требований к программному решению при наличии разбиения жизненного цикла на итерации

требует высокого уровня планирования, контроля и управления проектом).

Эволюционная стратегия разработки мобильных приложений обладает следующими достоинствами:

1) возможность уточнения требований на каждом цикле разработки (первоначально требования устанавливаются только частично, а затем уточняются по мере реализации каждого цикла разработки);

2) возможность использования промежуточных состояний программного решения (после каждого цикла разработки мобильное приложение представляет собой полностью работоспособную свою версию);

3) возможность осуществления гибкой разработки;

4) высокий уровень вовлечение заинтересованных лиц к процессу разработки программного решения (стейкхолдерам предоставляется возможность не только апробировать промежуточную версию мобильного решения, но и участвовать в корректировке требований к его новой версии).

К недостаткам эволюционной стратегии разработки мобильных приложений относят:

1) сложность применения (нестабильность требований к программному решению при наличии разбиения жизненного цикла на итерации, а также высокую вовлеченность в процесс разработки стейкхолдеров, требует высокого уровня планирования, контроля и управления проектом);

2) необходимость активного участия стейкхолдеров в проекте (требуются значительные усилия для их привлечения, а также специализированные инструментальные средства поддержки совместной работы).

К основным этапам разработки мобильного приложения соответственно относят:

1) анализ требований к мобильному приложению;

2) проектирование архитектуры мобильного приложения;

3) детальное проектирование мобильного приложения;

4) конструирование мобильного приложения;

5) комплексирование мобильного приложения;

6) квалификационное тестирование мобильного приложения.

Этап анализа требований к мобильному приложению начинается с выявления (сбора) требований к программному решению, а затем проводится их анализ и уточнение. Сбор предварительных требований может осуществляться на стадии замысла, однако их анализ и уточнение является основным процессом стадии разработки.

Сбор требований подразумевает несколько аспектов:

1) анализ специфики деятельности правообладателя;

2) и анализ подходов к решению стоящих перед ним задач в области применения разрабатываемого мобильного приложения;

3) анализ требований непосредственно к самому программному решению.

Специфика деятельности может быть выявлена в результате проведения информационного обследования деятельности (бизнеса) правообладателя, анализ подходов к решению стоящих перед его предприятием задач – в результате описания бизнес-процессов, анализ требований непосредственно к самому мобильному решению – в результате процедуры сбора и анализа требований.

Ранее было отмечено, что различают две категории обследования: экспресс-обследование и полное информационное обследование. На стадии реализации мобильного приложения с целью изучения специфики деятельности правообладателя проводится полное информационное обследование, основными задачами которого является:

1) выявления требований со стороны правообладателя;

2) выявление инструктивно-методических и директивных материалов, на основании которых определяются состав подсистем и перечень задач проектируемого мобильного приложения;

3) выявление возможности применения новых методов решения задач.

В полном информационном обследовании принимают участие: владельцы, высшее руководство, руководители бизнес-подразделений и сотрудники (специалисты).

Привлечение к обследованию высшего руководства проводится с целью определения проблемных зон в функциональном взаимодействии подразделений и информационном обмене, затрудняющих процесс принятия управленческих решений для руководства предприятия [2].

Привлечение к обследованию руководителей бизнес-подразделений проводится с целью определения проблемных зон в информационном обмене и функциональном взаимодействии подразделений.

Привлечение к обследованию специалистов проводится с целью выявления зоны ответственности конкретных сотрудников, основные «узкие места» процессов, с которыми им приходится сталкиваться при работе.

Основные этапы полного информационного обследования [2]:

1) сбор первичной информации;

2) систематизация и анализ информации;

3) представление собранных данных (в формате отчета) для согласования результатов обследования с заказчиком проекта.

Сбор первичной информации предполагает:

1) анкетирование и интервьюирование;

2) запрос первичных документов, регистров, отчетных, аналитических, плановых и т.д. документов;

3) запрос информации об организационной структуре, положений о структурных подразделениях и должностных обязанностях сотрудников;

4) запрос документации по используемым информационным системам.

Систематизация и анализ информации осуществляется в двух взаимосвязанных формах:

1) функции (информация о событиях и процессах, которые происходят в бизнесе);

2) сущности (информация об объектах, имеющих значение для организации).

Выявленные функции можно классифицировать следующим образом:

1) необходимые функции (события и процессы, без которых бизнес правообладателя не может успешно функционировать);

2) желательные функции (не критичные для бизнеса правообладателя события и процессы, реализация которых желательна в ближайшей перспективе, но ограничена временными и финансовыми рамками на момент обследования);

3) возможные функции (не критичные для бизнеса правообладателя события и процессы, реализация которых возможна в далекой перспективе, но ограничена временными и финансовыми рамками на момент обследования);

4) отсутствующие функции (не критичные для бизнеса правообладателя события и процессы, реализация которых нецелесообразна, но описание которых необходимо для идентификации границ проекта).

На основании выявленных функций и оценки их важности для реализации в рамках создаваемой информационной системы формируется их упорядоченный по приоритету востребованности список. Часть из них, которая подлежит реализации, станет основой для формулировки функциональных требований к информационной системе и её программной части в виде мобильного приложения.

Описание бизнес-процессов осуществляется с целью анализа подходов к решению стоящих перед предприятием задач. Задачей описания бизнес-процессов является формирование модели группы бизнес-процессов, участие в которых могут принимать одно или несколько подразделений.

Бизнес-процесс – это регулярно повторяющаяся последовательность взаимосвязанных мероприятий (операций, процедур, действий), при выполнении которых используются ресурсы внешней среды, создается ценность для потребителя и выдается ему результат.

Моделирование бизнес-процессов осуществляется в двух видах [7]:

1) «как есть» (отражает существующие в организации бизнес-процессы);

2) «как должно быть» (отражает необходимые изменения бизнес-процессов с учетом внедрения проектируемой информационной системы).

Основная цель моделирования процессов – их документирование и последующее осуществление функционального анализа на предмет поиска «узких» мест процессов и возможностей для их совершенствования.

К основным нотациям / методологиям моделирования бизнес-процессов относят BPMN, UML, IDEF, EEPC, BPMN (англ. Business

Process Model and Notation, нотация и модель бизнес-процессов) – система условных обозначений (нотация) для моделирования бизнес-процессов.

Основная цель BPMN – создание стандартного набора условных обозначений, понятных всем бизнес-пользователям. Бизнес-пользователи включают в себя бизнес-аналитиков, создающих и улучшающих процессы, технических разработчиков, ответственных за реализацию процессов и менеджеров, следящих за процессами и управляющих ими. Следовательно, BPMN призвана служить связующим звеном между фазой дизайна бизнес-процесса и фазой его реализации.

Моделирование в BPMN осуществляется посредством диаграмм с небольшим числом графических элементов. Это помогает пользователям быстро понимать логику процесса. Выделяют четыре основные категории элементов:

1) объекты потока управления: события, действия и логические операторы;

2) соединяющие объекты: поток управления, поток сообщений и ассоциации; роли:

3) пулы и дорожки;

4) артефакты: данные, группы и текстовые аннотации.

Элементы этих четырёх категорий позволяют строить простейшие диаграммы бизнес-процессов. Для повышения выразительности модели спецификация разрешает создавать новые типы объектов потока управления и артефактов.

Недостаток BPMN заключается в том, что моделирование иных аспектов, помимо бизнес-процессов, находится вне зоны внимания BPMN.

UML (Unified modeling language, универсальный язык моделирования) является методологией объектно-ориентированного подхода, представляющей набор диаграмм для проектирования информационных систем. UML версии 2.4.1 был также принят в качестве международного стандарта моделирования [8].

IDEF (Icam (Integrated computeraided manufacturing) DEFinition for functional modeling) представляет собой семейство стандартов описания и отображения бизнес-процессов, имеющее следующую структуру:

1) IDEF0, отображающая процесс на уровне функций;

2) IDEF1, фокусирующая на информационных потоках; IDEF1X для разработки реляционных баз данных;

3) IDEF2 для динамического моделирования систем;

4) IDEF3, моделирующая технологические процессы как следующий уровень после IDEF0;

5) IDEF4 для построения объектно-ориентированных систем;

6) IDEF5 для онтологического исследования сложных систем;

7) IDEF6, акцентирующая внимание на процессе создания модели (обстоятельствах и причинах выбора того или иного метода моделирования);

8) IDEF7 для аудита информационных систем;

9) IDEF8 для разработки пользовательских интерфейсов;

10) IDEF9 для определения бизнес-ограничений при сценарном проектировании информационных систем;

11) IDEF10 – IDEF14 – методы в области проектирования компьютерных сетей, архитектуры внедрения и прочих предметных областей.

Сбор и анализ требований к мобильному приложению предполагает преобразовании требований правообладателя, выраженных в виде его представлений о желаемых функциональных возможностях, в техническое видение требуемого мобильного приложения, способного предоставить такие функциональные возможности.

В широком смысле требование – это условие или возможность, которой должна соответствовать программный продукт [9].

Различают функциональные и нефункциональные требования к проектируемому мобильному приложению.

Функциональные требования определяют поведение мобильного приложения и отвечают на вопрос «что должно делать мобильное приложение» в рамках реализации различных сценариев своего функционирования.

Нефункциональные требования определяют характеристики, которые должно демонстрировать мобильное приложение и которые не относятся к его поведению (например, внешние интерфейсы программного решения, его атрибуты качества и ограничения с ним связанные).

К внешнем интерфейсам мобильного приложения относят интерфейс пользователя и программные интерфейсы, к основным атрибутам

качества мобильного приложения – удобство использования, производительность, гибкость и т.д.

Ограничения представляют собой условия, которые сужают выбор возможных решений по реализации функциональных требований (например, выбор среды разработки, протоколов передачи данных, языка программирования, способов хранения данных и т.д.)

Все требования можно разделить на следующие группы:

1) маркетинговые требования;

2) требования к мобильному приложению;

3) функциональные спецификации.

Маркетинговые требования включают:

а) бизнес требования (стимулы, целевой рынок, сценарии использования и проблемы, обзор конкурентов);

б) решения (образ мобильного приложения, список его возможностей);

в) мероприятия по созданию следующей версии мобильного приложения (суть проекта, возможности продукта, вехи);

Требования к мобильному приложению включают:

а) профили стейкхолдеров (пользователей);

б) пользовательские истории (сценарии работы стейкхолдеров);

в) варианты использования;

г) пользовательские требования.

Функциональные спецификации включают:

а) системные требования;

б) требования проектирования;

в) ограничения интерфейсов;

г) модель взаимодействия (варианты использования, диаграмма активности, макеты графического пользовательского интерфейса);

д) техническая информация.

В качестве источников требований к мобильному приложению выступают:

1) соображения, высказанные или приписываемые стейкхолдерам и их группам;

2) артефакты, описывающие предметную область;

3) «лучшие практики» (описание моделей деятельности успешных компаний отрасли, используемые длительное время в сотнях и тысячах компаний по всему миру).

К моменту сбора требований исходные стимулы правообладателя, а также цели создания и критерии успеха мобильного приложения являются определенными. Поэтому следующим шагом является определение того, кто, какие проблемы и в каких условиях будет решать при помощи создаваемого мобильного приложения.

Наиболее эффективным способом получения ответа на эти вопросы является определение сценариев работы всех заинтересованных лиц (стейкхолдеров) с будущим мобильным приложением.

Стейкхолдер (также заинтересованная сторона, причастная сторона, участник работ, роль в проекте) – лицо или организация, имеющая права, долю, требования или интересы относительно программной системы или её свойств, удовлетворяющих их потребностям и ожиданиям [10, 11].

Иными словами, стейкхолдер – индивидуум, команда, организация или их группы, имеющие интерес в реализации мобильного программного решения.

Исчерпывающего списка типов (групп) стейкхолдеров не существует, так как для различных целевых мобильных приложений они могут значительно отличаться. Можно привести примеры наиболее распространённых типов (групп) стейкхолдеров, которые упоминаются в стандартах [2] и Своде знаний по системной инженерии (SEBoK) [12]: потребитель продукции (группы покупателей, заказчиков и клиентов и ключевые субъекты); разработчик; поставщик; пользователь; производитель; сопровождающая сторона; ликвидатор; регулятор; персонал (руководители отдельных служб, группы сотрудников, отдельные работники); операторы и другие.

Пример идентификации стейкхолдеров (профиля пользователя) приведен на рис. 3.4.

Сценарий – это совокупность всех процессов, в которых будет участвовать мобильное приложение, а также описание окружения, в котором его планируется использовать.

Сценарий не должен являться описанием работы отдельного стейкхолдера для достижения конкретной цели. Его ценность состоит в том, что он описывает способы взаимодействия с мобильным приложением всех его пользователей одновременно на протяжении всего цикла его эксплуатации.

Группа стейкхолдеров: пользователи мобильного приложения	
Наименование стейкхолдера:	***Гость (незарегистрированный пользователь)***
Цели стейкхолдера	*Просмотр товарного каталога* *Регистрация в приложении* *Авторизация в приложении*
Наименование стейкхолдера:	***Посетитель (неаутентифицированный пользователь)***
	Просмотр товарного каталога *Авторизация в приложении*
Наименование стейкхолдера:	***Покупатель (аутентифицированный пользователь)***
Цели стейкхолдера	*Просмотр товарного каталога* *Покупка товара* *Окончание работы*

Рис. 3.4. Пример идентификации стейкхолдеров мобильного приложения

Каждый сценарий должен содержать как минимум следующую информацию: краткое наименование, группа участников (или их роль), описание целей участников, описание необходимых шагов и вариантов развития событий (реализация), а также приоритет.

Для приоритезации сценариев может быть использован метод MoSCoW [13], в котором используются четыре оценки:

1) M - MUST (описываемое требование должно быть выполнено обязательно при выпуске текущей версии мобильного приложения);

2) S – SHOULD (описываемое требование должно быть выполнено при наличии возможности, в случае отсутствия таковой – должны быть выполнены при выпуске ближайшей версии мобильного приложения);

3) C - COULD (описываемое требование желательно выполнить при наличии возможности);

4) W - WON'T (описываемое требование в настоящий момент не подлежит выполнению в виду низкого уровня важности для достижения целей мобильного приложения или отсутствия экономической эффективности реализации).

Пример описания сценариев работы стейкхолдеров (пользовательских историй) приведен на рис. 3.5.

Наименование сценария: регистрация гостя	
Стейкхолдеры (группа пользователей, выполняемая роль)	*Гость*
Цели стейкхолдера	*Регистрация в приложении*
Реализация	*Заполнение и оправка на сервер формы регистрации с обязательным указанием своего имени, адреса электронной почты и желаемого пароля*
Приоритет	*S - SHOULD (Необходимо)*
Наименование сценария: авторизация гостя	
Участники (группа пользователей, выполняемая роль)	*Гость*
Цели участников	*Авторизация в приложении*
Реализация	*Заполнение и оправка на сервер формы авторизации с обязательным указанием адреса электронной почты и пароля*
Приоритет	*S - SHOULD (Необходимо)*
Наименование сценария: выход из системы пользователя	
Участники (группа пользователей, выполняемая роль)	*Аутентифицированный пользователь*
Цели участников	*Окончание работы (например, чтобы исключить осуществление несанкционированные действий в приложении)*
Реализация	*Нажатие кнопки «Выход» или бездействие в течении 10 минут*
Приоритет	*W - WON'T (Не делать)*

Рис. 3.5. Пример описания сценария работы стейкхолдеров с мобильным приложением

Разработанные сценарии позволяют провести идентификацию и описание уникальных требований к мобильному приложению. Так как разработанные сценарии работы стейкхолдеров с создаваемым мобильным приложением за счет дублирующих друг друга элементов являются избыточными, то требуется проведения их систематизации.

В процессе такой систематизации выделяются основные требования и требования, которые их дополняют, что позволяет представить их в виде дерева, в котором элементами первого уровня являются независимые с точки зрения возможности реализации требования, а дочерними элементами – дополнительные требования.

Необходимость такого разделения заключается в возможности анализа каждого требования по отдельности с целью рассмотрения возможности реализации их в текущей или ближайшей версии мобильного приложения (для этого отдельным требованиям назначаются

определенные приоритеты, которые комбинируются с унаследованным от родительских сценария приоритетом).

Пример описания требований к мобильному приложению, сформулированных на основание сценариев работы стейкхолдеров приведен на рис. 3.6.

(ФТ) Функциональные требования	
(ФТ.1) Наименование группы требований: общие требования	
(ФТ.1.1) Наименование требования: обеспечение регистрации гостя	
Родительские сценарии	*Регистрация гостя*
Максимальный приоритет родительского сценария	*S - SHOULD (Необходимо)*
Описание требования	*Предусмотреть возможность регистрации гостя в мобильном приложении посредством заполнения и оправки на сервер формы регистрации с обязательным указанием своего имени, адреса электронной почты и желаемого пароля* *В форме должна быть* *- галочка согласия на обработку персональных данных со ссылкой на страницу об использовании - пока она не установлена, продолжить регистрацию нельзя* *- галочка согласия получать пуш-уведомлений об акциях - по умолчанию стоит.*
Приоритет требования	*C – COULD (желательно)*
Общий приоритет требования	*S - SHOULD (необходимо)*
(НФТ) Нефункциональные требования	
(ФТ.1) Наименование группы требований: обеспечение высокого уровня юзабилити	
(ФТ.1.1) Наименование требования: обеспечение корректного отображения на экранах мобильных устройств с разной ориентацией экрана	
Родительские сценарии	*Удобство использования*
Максимальный приоритет родительского сценария	*S - SHOULD (необходимо)*
Описание требования	*При работе на мобильных телефонах ориентацией экрана является портретной.* *Верстка должна учитывать смену ориентации экрана: размещать блок информации по центру, шапку растягивать.*
Приоритет требования	*C – COULD (желательно)*
Общий приоритет требования	*S - SHOULD (необходимо)*

Рис. 3.6. Пример описания требований к мобильному приложению

Все сформулированные требования должны быть согласованы между собой, а также с требованиями к информационной системе, ча-

стью которой является мобильное приложение. Кроме того, все разработанные требования должны обладать свойствами осуществимости, тестируемости и возможности сопровождения

Для уточнения сформулированных требований и придания создаваемому мобильному приложению конкурентных преимуществ проводят его сравнение с мобильными решениями конкурентов.

От результатов данного сравнения ожидают реализацию в требованиях к создаваемому мобильному приложению наиболее удачных решений, которые нашли воплощение в продуктах конкурентов. При этом решения, которые оказались неудачными, игнорируются или перерабатываются. Идеальным является обоснование и включение новых уникальных решений, которые дадут возможность владельцу мобильного решения получить преимущества в конкурентной борьбе.

Структура обзора конкурентов, обычно следующая:

1) идентификация лидеров отрасли с разработкой резюме по каждому из них;

2) идентификация и анализ решаемых с помощью мобильного приложения проблем лидеров отрасли;

3) идентификация и анализ возможностей мобильных приложений лидеров отрасли с разработкой их реестра.

Следует отметить, что обзор конкурентов проводится только на основе лидеров отрасли и их мобильных приложений (при этом не имеет значение то, каким образом они коммерциализируется).

Определить лидеров отрасли можно, используя различные обзоры, результаты опросов или продаж. Также можно использовать маркетинговые и рекламные материалы конкурентов, а также руководства пользователя (при их наличии).

Если предметная область продуктов достаточно популярна, то в сети интернет можно найти уже готовый обзор, который будет содержать необходимую информацию.

Анализ бизнеса конкурентов и место, которое в этом бизнесе занимает их мобильные приложения, можно с использованием различных подходов (например, с использованием инструмента стратегического управления, используемого для описания бизнес-моделей предприятий – бизнес-модели Остервальдера (Business Model Canvas) [14].

Также следует описать преимущества и недостатки бизнес-модели конкурента, выделить используемые конкурентом в его бизнесе

интересных идей, в т.ч. при использовании возможностей его мобильного приложения.

Идентификация решаемых с помощью мобильного приложения проблем лидеров отрасли может быть проведена на основе документации для пользователей, в которых описаны сценарии использования продукта в тех или иных ситуациях (это касается приложений и сервисов, работающих на обширную аудиторию в рамках их информационных систем).

Также идентификация может быть определена на основе различных маркетинговых материалов, в которых описаны выгоды, которые сулит мобильное приложении при его использовании.

По завершении исследования проблем, которые решают мобильные приложения конкурентов, следует провести повторный анализ бизнес требований к своему решению. Полученная информация о конкурентах поможет улучшить ранее разработанные бизнес требования предприятия.

Идентификация функциональных и нефункциональных возможностей мобильных приложений лидеров отрасли проводится с целью определения подходов к решению проблем конкурентов с их помощью, а также определения сильных и слабых сторон таких решений.

В заключении обзора конкурентной среды необходимо выявить наиболее перспективные пути развития создаваемого мобильного приложения и его шансы на успех.

Этап анализа требований к мобильному приложению считается выполненным если оформлены следующие документы:

1) спецификации функциональных требований;

2) спецификации нефункциональных требований;

3) описание внешних интерфейсов (например, в форме вайрфреймов – набросков всех основных компонентов пользовательского интерфейса, которые предполагается разместить на экранах мобильного приложения);

4) описание данных и требования к базам данных (например, в форме инфологической модели предметной области);

5) описание методов зашиты конфиденциальной информации;

6) другие определяемые предметной областью документы.

Все данные документы могут быть объединены в рамках технического задания на разработку мобильного приложения, которое либо

является неотъемлемой частью договора между заказчиком и подрядчиком (подрядный способ), либо передается внутреннему подразделению организации для выполнения работ по созданию программного решения.

В соответствии с ГОСТ 34.602-89 [15] техническое задание на разработку информационной системы в себя следующие разделы:

1) общие сведения;

2) назначение и цели создания системы;

3) характеристика объектов автоматизации;

4) требования к системе;

5) состав и содержание работ по созданию системы;

6) порядок контроля и приёмки системы;

7) требования к составу и содержанию работ по подготовке объекта автоматизации к вводу системы в действие;

8) требования к документированию;

9) источники разработки;

В соответствии с п. 2.2 указанного выше стандарта в зависимости от вида, назначения, специфических особенностей объекта автоматизации и условий функционирования системы допускается оформлять разделы техническое задания в виде приложений, вводить дополнительные, исключать или объединять подразделы.

Следующим этапом разработки мобильного решения является проектирование архитектуры мобильного приложения, которая представляет собой совокупность важнейших решений относительно организации создаваемой программной системы для мобильных устройств (программных компонентов, модулей, размещений, их свойств и отношений между ними). Архитектура мобильного приложения также определяет используемый набор технологий для разработки программного решения и руководящие правила проектирования и развития программной системы [16].

Целью проектирования архитектуры мобильного приложения является создание такого программного решения, которое с одной стороны может легко быть адоптировано под изменяющийся нужды правообладателя, а с другой стороны позволит реализовать не только функциональные требования, но и удовлетворить атрибуты качества программного решения.

К архитектуре мобильного приложения предъявляются следующие требования:

1) эффективность выполнения функций при любой интенсивности использования;

2) гибкость и возможность масштабирования;

3) эффективность тестирования;

4) защищенность и безопасность;

5) надежность;

4) простота и структурированность.

Архитектура мобильного приложения, как и любой другой программной системы, в зависимости от её уровня рассмотрения может быть разделена на логическую и физическую составляющие [17].

Понятие логического уровня подразумевает мышление в понятиях реального мира (например, с точки зрения различных пользователей), и соответственно, взятие непосредственно из реального мира объектов для разработки мобильного приложения. Понятие физический уровень подразумевает мышление в понятиях исполнителя программного кода (то есть зависит от конкретной среды исполнения кода мобильного приложения, т.е. понятен проектировщикам и конструкторам).

Логическая архитектура описывает функции, поведение, использование мобильного приложения с точки зрения правообладателя и состоит из набора связанных концепций и принципов.

Проектирование логической архитектуры мобильного приложения предполагает описание [18]:

1) внутренних и внешних пользователей программной системы;

2) основных функций программной системы;

3) основных видов поведения системы программной системы;

4) определяющих границу между программной системой и её окружением компонентов;

5) наборов данных;

6) потоков данных;

7) функциональных интерфейсов.

В качестве выходных документов при проектировании логической архитектуры могут выступать: диаграммы процессов на уровне функций (IDEF0), контекстные диаграммы, схемы функциональных

потоков (FFBD), диаграммы потоков данных, диаграммы последовательности форм (например, в форме карты переходов между экранами) и концептуальное решение дизайна внешнего интерфейса мобильного приложения.

Пример карты перехода между экранами, как выходного документа при проектировании логической архитектуры мобильного приложения, приведен на рис. 3.7.

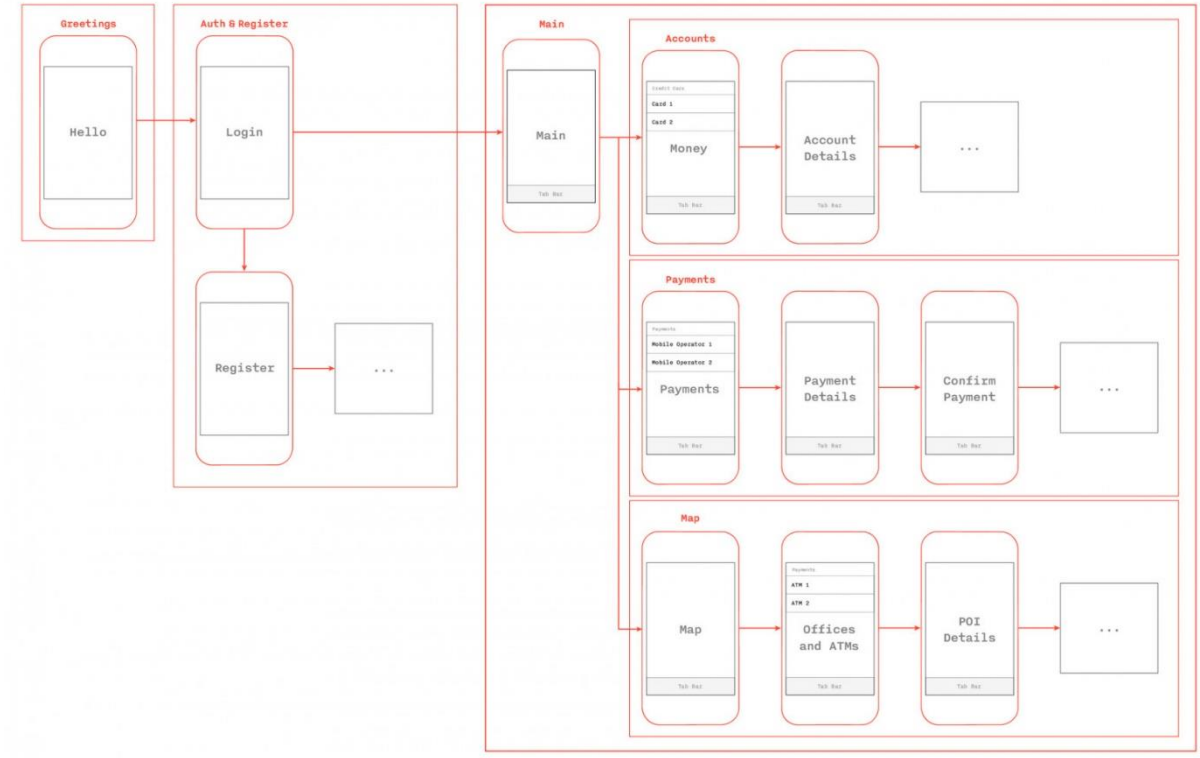


Рис. 3.7. Логическая карта переходов между экранами

Завершение проектирования логической архитектуры мобильного приложения предполагает идентификацию конкретных физических элементов, которые поддерживают функциональные, поведенческие и временные свойства программной системы [19].

В качестве указанных физических элементов выступают:

1) программные компоненты (библиотеки, модули, классы и т.д.);
2) интерфейсы программных компонентов;
3) потоки взаимодействия между программными компонентами;
4) структуры данных.

В качестве выходных документов при проектировании физической архитектуры могут выступать: диаграммы архитектуры программной системы, детализированные физические блок-схемы, диаграммы взаимодействия компонентов, топологии баз данных, физическая карта переходов между экранами.

Пример физической карты переходов между экранами мобильного приложения, приведен на рис. 3.8.

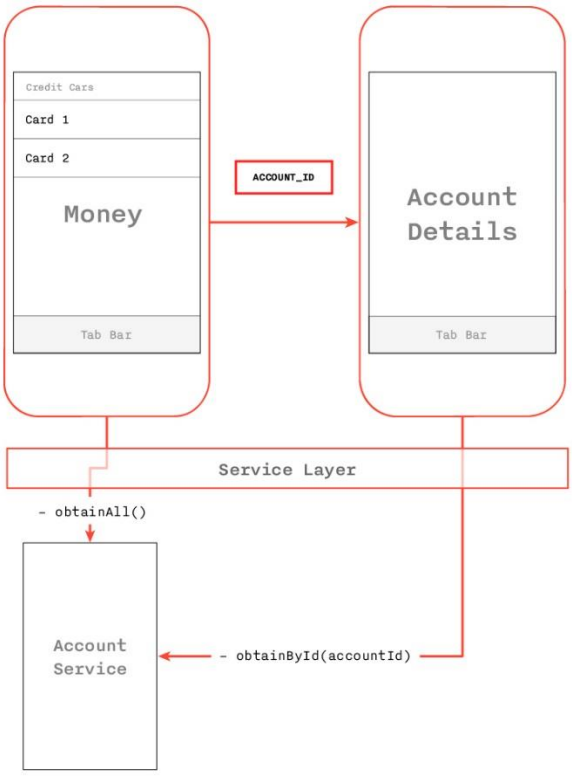


Рис. 3.8. Физическая карта переходов между экранами

Таким образом, при проектировании архитектуры мобильного приложения необходимо преобразовать требования к программному решению в его логическую и физическую архитектуру.

Все компоненты мобильного приложения должны быть согласованы как между собой, так и с требованиями, установленными к программному решению в целом. Кроме того, разработанный проект архитектуры мобильного приложения, его интерфейсов и баз данных должны обладать свойствами приспособленности методов проектирования, осуществимости детального проектирования, функционирования и сопровождения программного решения.

На основании проекта архитектуры осуществляется детальное проектирование мобильного приложения, которое заключается в подробном описании каждого программного компонента и его внешних интерфейсов, а также подходов к его тестированию с целью наиболее полной подготовки проекта к последующему конструированию.

Типичная схема процесса детального проектирования выглядит следующим образом:

1) определяются наиболее сложные и уязвимые программные компоненты, а также взаимосвязи между ними, которые требуют детальной проработки;

2) для выбранных компонентов программной системы уточняются и детализируются архитектурные модели физического уровня (например, разрабатывается подробная диаграмма классов, содержащая пограничные классы, классы-сущности и управляющие классы, разрабатываются подробные диаграммы последовательности и диаграммы потоков данных, уточняется модель базы данных и модель пользовательского интерфейса и т.д.);

3) осуществляется детальная проработка и проектирование каждого класса (уточняются и дополняются атрибуты, методы и их сигнатуры, область видимости и типы, устанавливаются инварианты класса и т.д.);

4) осуществляется детальная проработка и проектирование каждого метода каждого класса (задаются и прорабатываются предусловия и постусловия метода, обеспечивается описание используемого нетривиального алгоритма с помощью блок-схем, диаграмм деятельности и т.д.);

5) для программной системы и для каждого программного компонента разрабатывается план тестирования;

6) осуществляется инспектирование выполненного детального проекта (в качестве критериев оценки используются: согласованность детального проекта с требованиями к программной системе и с архитектурным проектом, согласованность компонентов между собой, осуществимость тестирования, функционирования и сопровождения как отдельных компонентов, так и программной системы в целом);

7) осуществляется выпуск детального проекта на этап конструирования.

Выполненный детальный проект мобильного приложения позволяет оценить с большой точностью объёмы работ по конструированию, комплексированию и тестированию программной системы, построить календарные графики их выполнения с точностью до отдельных задач, а также закрепить за ними исполнителей.

На основании детального проекта осуществляется конструирование мобильного приложения, которое заключается в создании исполняемых программных компонентов.

Типичная схема процесса конструирования программного компонентов мобильного приложения выглядит следующим образом:

1) определяется стандарт кодирования, инструменты и среда программирования;

2) для каждого класса программного компонента на основе детального проекта кодируются классы и их члены (структура базы данных также кодируется в методах класса, например в виде SQL-предложений);

3) для каждого класса программного компонента выполняется инспектирование кода;

4) осуществляется тестирование каждого программного компонента в соответствии с планом, проработанном при осуществлении детального проектирования.

5) осуществляется инспектирование каждого программного компонента (в качестве критериев оценки используются: согласованность кода и результатов тестирования с требованиями к программной системе и к отдельным её компонентам, согласованность компонентов между собой, осуществимость комплексирования, тестирования, функционирования и сопровождения программной системы, уровень тестового покрытия компонентов и др.);

6) осуществляется выпуск созданного программного компонента на этап комплексирования.

При конструировании мобильных приложений необходимо учитывать особенности операционной системы, на которую оно будет установлено.

Операционная система Android является самой востребованной в мире по состоянию на начало 2022 года: более 70% всех мобильных устройств работают под её управлением. iOS занимает второе место имея чуть более четверти доли рынка.

Исследования показывают, что более 60% пользователей Android используют старые версии операционной системы, у iOS ситуация обратная: 80% пользователей используют последнюю версию. Данный факт надо учитывать разработчикам мобильных приложений для максимального распространения своего решения.

Операционная система Android является открытой операционной системой и поддерживает широкий спектр мобильных устройств. Особенностями данной операционной системы с точки зрения разработки по них мобильных приложений являются: многозадачность, множество устройств и форм-факторов (охватить в проекте все возможные варианты не предоставляется возможным), выполнение приложений в изолированной среде (обмен данными с другими приложениями затруднен). Подробнее операционная система Android и детали разработки мобильных приложений для неё будет рассмотрены в последующих главках.

Операционная система iOS используется только для установки на мобильные устройства, произведенные корпорацией Apple. Особенностями данной операционной системы в свою очередь являются: многозадачность, зависящие от устройств ресурсы, а также выполнение в изолированной среде.

Комплексирование предполагает объединение программных компонентов в единую интегрированную программную систему, которая полностью отвечает всем функциональным и нефункциональные требованиям, которым должно соответствовать мобильное приложение. Сборка мобильного приложения осуществляется в соответствии с архитектурным проектом.

Конструирование и комплексирование мобильного приложения может быть осуществлено либо вручную, либо с помощью графических конструкторов, среди которых наиболее востребованными являются: Android Studio, RAD Studio и Eclipse IDE.

Сравнение указанных программ для разработки мобильных приложений приведено в работе [20]. Результаты данного сравнения представлены на рис. 3.9.

После завершения конструирования и комплексирования мобильного приложения необходимо выполнить квалификационное тестирование, которое проводится для того, чтобы подтвердить соответствие результата установленным требованиям.

Название программы	Платформы разработки	Язык программирования	Наличие бесплатной версии	Возможность работы в облаке	Язык интерфейса
Android Studio	Android	Java, C/C++	Скачивание бесплатно	Есть	Мультиязычный, есть русский
RAD Studio (Berlin)	Android	Object Pascal, C++	248 999₽	Есть	Английский, Немецкий, Французский, Японский
Eclipse IDE	Android, IOS, windows phone	Java, C/C++, PHP, Ruby, Python,	Скачивание бесплатно	нет	Английский

Рис. 3.9. Сравнение программ для разработки мобильных приложений

На данном этапе проводится системное тестирование, верификация и валидация созданного мобильного приложения.

Системное тестирование направлено на проверку корректности программной реализации мобильного приложения в условиях реальной работы. Различают следующие виды системного тестирования: альфа-тестирование (к проверке привлекаются узкий круг представителей разработчика и правообладателя) и бета-тестирование (к проверке привлекаются широкий круг пользователей).

Особенностью тестирования мобильных приложений является многообразие целевых моделей устройств, их технической конфигурации и версий установленных операционных систем.

Тестирование может проводится вручную или автоматизировано. В качестве технических средств тестирования мобильных приложений могут выступать [21]:

1) реальное устройство (предполагает тестирование программной системы в реальных условиях с учетом различных форм-факторов: предоставляет максимально точные результаты, но имеет низкую масштабируемость и высокую стоимость);

2) локальный эмулятор (предполагает тестирование программной системы в программной модели реального устройства с заданными характеристиками форм-фактора на рабочем месте разработчика: предоставляет неполные результаты, но имеет низкую стоимость, высокое быстродействие и простую настройку);

3) облачный эмулятор (предполагает тестирование программной системы в программной модели реального устройства с заданными ха-

рактеристиками форм-фактора в облаке: за счет большого выбора моделей результаты являются более полными чем при использовании локального эмулятора, но быстродействие при этом уменьшается).

Верификация направлена на подтверждение того, что заданные (специфицированные) требования к мобильному приложению полностью реализованы в готовом продукте. Иными словами, верификации должна подтвердить, что разработчик создал программный продукт так как изначально предполагалось.

Верификации подлежат: требования (осуществляется проверка на корректность, согласованность, выполняемость и проверяемость), архитектурный и детальный проект, код, комплексирование и документация. Проблемы и несоответствия, обнаруженные при проведении верификации, должны быть устранены.

Валидация направлена на подтверждение того, что мобильное приложение соответствует своему назначению, в т.ч. заданные (специфицированные) требования к мобильному приложению выполняются для конкретного применения рабочего программного продукта. Иными словами, валидация должна подтвердить, что разработчик сделал «правильный» программный продукт.

Квалификационное тестирование является заключительным этапом разработки мобильного приложения. Следующей задачей является развертывание и внедрение мобильного решения.

3.4. Развертывание, функционирование и сопровождение мобильного приложения

Развертывание и инсталляция мобильного приложения предполагает копирование на мобильное устройство сотрудника или внешнего пользователя инсталляционного файла и установку его в целевую среду применения.

Развертывание мобильного приложения может быть организованно одним из следующих способов:

1) публикация инсталляционного файла в агрегаторах мобильных приложений;

2) размещение инсталляционного файла на корпоративном сервере правообладателя;

3) размещение инсталляционного файла в облаке.

Google Play и App Store в настоящий момент являются основными агрегаторами мобильных приложений, предлагая простой и универсальный способ распространения мобильного приложения: пользователю достаточно сделать описание мобильного приложения и закачать инсталляционный файл на сервер агрегатора.

Агрегаторы мобильных приложений предъявляют к размещаемым приложениям ряд требований, которые, как правило, связаны с:

1) с соблюдением прав на интеллектуальную собственность;
2) с достоверностью описания приложения;
3) с запрещенными методами продвижения;
4) с нарушением информационной безопасности;
5) с использованием некачественного контента;
6) с использованием недопустимых систем оплаты;

Если мобильное приложение отвечает всем требованиям, то после непродолжительной модерации оно размещается на страницах агрегатора и становится доступным для скачивания и установки широкого кругу потенциальных пользователей.

Размещение инсталляционного файла на корпоративном сервере правообладателя или в облаке предполагает, как правило, внедрение MDM-решений, которые нами были рассмотрены в первой главе.

С точки зрения развертывания мобильного приложения MDM-решение кроме размещенного на сервере или в облаке контрольного центра и установленного на мобильном устройстве сотрудника клиентского программного обеспечения предполагает использование специальных протоколов взаимодействия сервера и мобильных устройств.

При внедрении мобильных решений в корпоративную информационную среду необходимо предварительно закупить и настроить требуемую для их функционирования IT-инфраструктуру, а также провести обучение сотрудников. При необходимости должна быть проведены основные виды тестирования и опытно-промышленная эксплуатация.

Процесс функционирования мобильного решения предполагает его применение по назначению и обеспечение поддержки правообладателей программного решения.

В рамках данного процесса осуществляется сбор данных об инцидентах, связанных с применением мобильного приложения. Сбор информации производится через встроенные в приложения механизмы

аудита, а также через механизмы обратной связи с пользователями (например, на странице приложения в агрегаторе мобильных приложений). Полученные данные регистрируются, классифицируются по степени риска и контролируются, а при экономической целесообразности вызвавшие инциденты причины устраняются в ближайших обновлениях программного решения).

Обеспечение поддержки правообладателей программного решения осуществляется по их просьбе в форме содействия по решению связанных с применением мобильного приложения проблем, а также в форме консультирования.

Процесс функционирования мобильного решения предполагает обеспечение его эффективной по затратам поддержки и включает усовершенствование (модификацию) и устранение дефектов (техническую поддержку) программного решения.

Техническая поддержка мобильного приложения осуществляется в следующих формах:

1) в форме коррекции (дефекты идентифицируются и устраняются);

2) в форме адаптации (развитие технологий приводит к модификации мобильных устройств, операционных систем и систем управления базами данных, что требует корректировки программного решения).

Модификация мобильного приложения осуществляется в следующих формах:

1) в форме улучшения (предлагаемые правообладателями изменения учитываются и при экономической эффективности реализуются);

2) в форме упреждения (по мере осуществления корректировок программного решения его структура усложняется, с каждым разом увеличивая затраты на сопровождение, что обуславливает внесение а приложение изменений для обеспечения удобства его сопровождения).

Процесс прекращения применения мобильного приложения завершает деятельность по поддержке его функционирования и сопровождения. При этом основной задачей является обеспечение сохранения целостности тех сохраняемых бизнес-процессов, в рамках которых

программная система используется. Данные, используемые в утилизируемом мобильном приложении, должны быть сохранены и доступны пользователям для получения.

О прекращении функционирования мобильного приложения его пользователи должны быть оповещены заранее. В частности, пользователи должны получить исчерпывающую информацию о причинах прекращения поддержки или вывода из эксплуатации программного продукта, а также информацию с описанием возможных вариантов замены мобильного приложения на другие решения.

В случае замены функционирующего мобильного приложения на новое необходимо обеспечить плавный переход пользователя, который подразумевает в том числе обучение пользователей.

Вопросы для обсуждения

1. Роль мобильных приложений в информационной архитектуре предприятия.
2. Концепция уровней жизненного цикла мобильного приложения.
3. Процессы в контексте информационной системы: понятие, назначение и основные группы процессов.
4. Специальные процессы программных средств: понятие, назначение и основные группы процессов.
5. Процесс менеджмента модели жизненного цикла: понятие, цели, задачи и результат.
6. Процесс менеджмента инфраструктуры: понятие, цели, задачи и результат.
7. Процесс менеджмента портфеля проектов: понятие, цели, задачи и результат.
8. Процесс менеджмента людских ресурсов: понятие, цели, задачи и результат.
9. Процесс менеджмента качества: понятие, цели, задачи и результат.
10. Процесс менеджмента планирования проекта: понятие, цели, задачи и результат.
11. Процесс менеджмента управления и оценки проекта: понятие, цели, задачи и результат.

12. Процесс менеджмента решений: понятие, цели, задачи и результат.

13. Процесс менеджмента рисков: понятие, цели, задачи и результат.

14. Процесс менеджмента конфигурации: понятие, цели, задачи и результат

15. Процесс менеджмента информации: понятие, цели, задачи и результат.

16. Процесс определения требований правообладателей: понятие, цели, задачи и результат.

17. Процесс измерений: понятие, цели, задачи и результат.

16. Процесс анализа системных требований: понятие, цели, задачи и результат.

17. Процесс проектирования архитектуры системы: понятие, цели, задачи и результат.

18. Процесс реализации: понятие, цели, задачи и результат.

19. Процесс менеджмента документации программных средств: понятие, цели, задачи и результат.

20. Процесс менеджмента конфигурации программных средств: понятие, цели, задачи и результат.

21. Процесс обеспечения гарантии качества программных средств: понятие, цели, задачи и результат.

22. Процесс верификации программных средств: понятие, цели, задачи и результат.

23. Процесс валидации программных средств: понятие, цели, задачи и результат.

24. Процесс ревизии программных средств: понятие, цели, задачи и результат.

25. Процесс аудита программных средств: понятие, цели, задачи и результат.

26. Процесс проектирования доменов: понятие, цели, задачи и результат.

27. Процесс менеджмента повторного применения активов: понятие, цели, задачи и результат.

28. Процесс менеджмента повторного применения программ: понятие, цели, задачи и результат.

29. Формирование замысла мобильного решения в контексте его использования в рамках информационной системы: понятие, специфические особенности, цели, задачи и результат.

30. Разработка (в т.ч. анализ и постановка задачи, проектирование, конструирование, комплексирование и тестирование) мобильного решения: понятие, специфические особенности, цели, задачи и результат.

31. Развертывание и внедрение мобильного решения (в т.ч. при необходимости публикация приложения на электронных площадках агрегаторов): понятие, специфические особенности, цели, задачи и результат.

32. Применение мобильного решения и его поддержка: понятие, специфические особенности, цели, задачи и результат.

33. Прекращения применения мобильного решения и его списание: понятие, специфические особенности, цели, задачи и результат.

34. Стратегии управления жизненным циклом мобильных приложение: понятие, специфические особенности, преимущества и недостатки.

35. Стимулы к созданию или модернизации мобильного приложения.

36. Формулировка образа мобильного решения.

37. Экспресс-обследование и полное информационное обследование: понятие, методы и результаты.

38. Основным нотации и методологии моделирования бизнес-процессов, используемые при разработке мобильных приложений.

39. Требования к архитектуре мобильного приложения.

40. Состав проекта логической архитектуры мобильного приложения.

41. Схема процесса детального проектирования мобильного приложения.

42. Схема процесса детального конструирования и комплексирования мобильного приложения.

43. Платформы разработки под основные операционные системы мобильных приложений.

44. Методы развертывание мобильного приложения: понятие, преимущества и недостатки.

45. Требования агрегаторов мобильных приложений к размещаемым приложениям.

46. Методы технической поддержки функционирования мобильного приложения.

47. Методы технической модификации мобильного приложения.

Практические задания

Задание 3.1.

Идентифицировать конкретные проблемы и благоприятные возможности, связанные с созданием или модификацией мобильного приложения предприятия, которое ведет заданный вид хозяйственной деятельности (по вариантам). Дать их развернутую характеристику. Предприятие выбрать самостоятельно.

Оформить результаты выполнения данного задания предлагается с использованием следующей таблицы:

Наименование проблемы, благоприятной возможности	*Краткая характеристика проблемы, благоприятной возможности*	*Уровень значимости решения проблемы (по шкале от 1 до 10)*
1	*2*	*3*

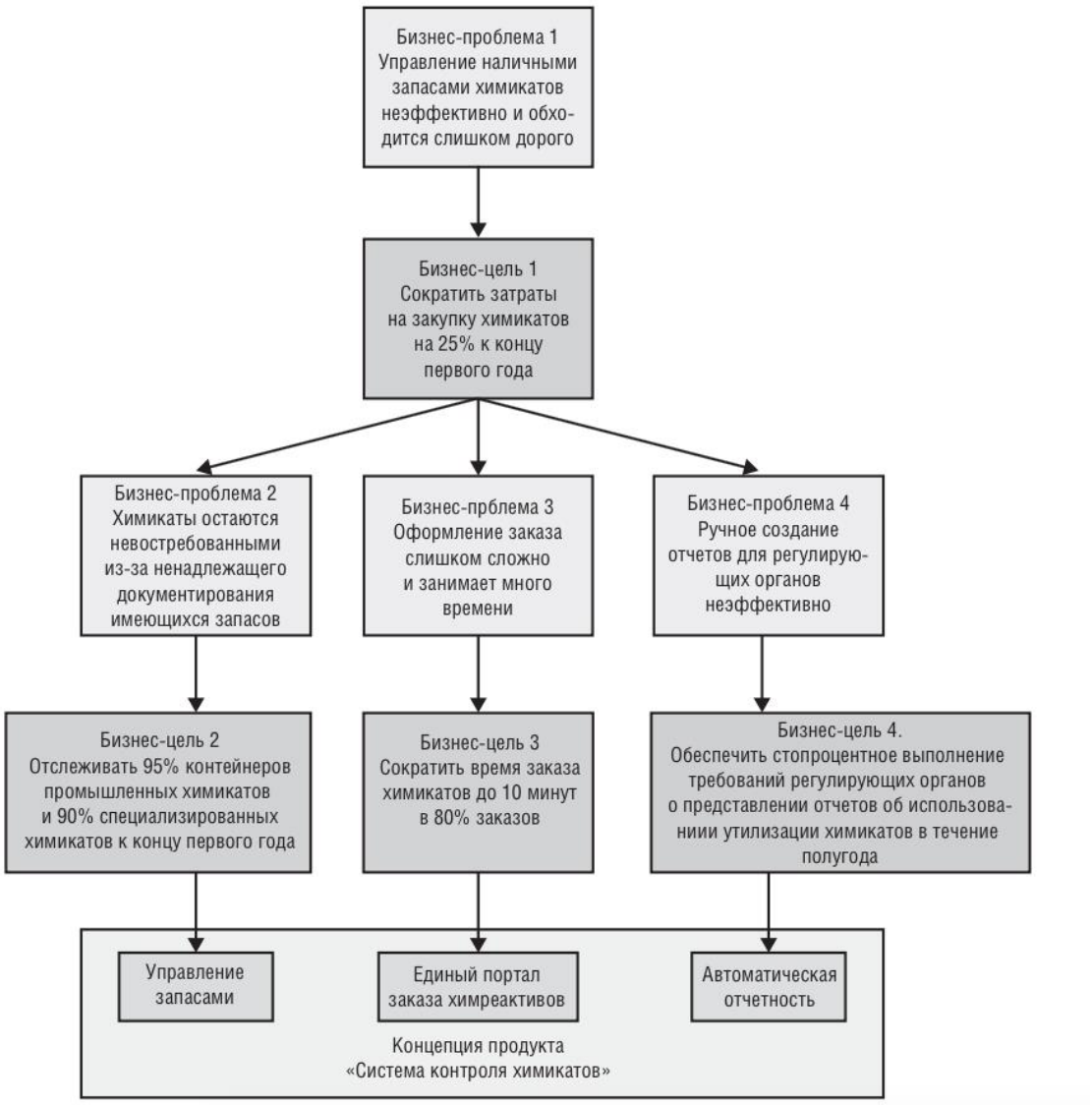
Варианты видов деятельности:

1) банковская деятельность;
2) рекламная деятельность;
3) риэлтерская деятельность;
4) оказание услуг общественного питания;
5) розничная торговля;
6) сельское хозяйство;
7) транспортные услуги;
8) туристические услуги;
9) услуги страхования;
10) бытовые услуги;
11) страховая деятельность;
12) доставка продуктов;
13) образовательная деятельность.

Задание 3.2.

Определить цели мобильного приложения и критериев успеха согласовать с ранее выявленными в задании 3.1 стимулами и оформить в

виде модели бизнес-целей, внешний вид которой приведен на рисунке (пример модели бизнес-целей для мобильного приложения контроля химикатов):



Задание 3.3.

Осуществить идентификацию стейкхолдеров мобильного приложения в соответствии с исходными данными и решением заданий 3.1 и 3.2.

Решение оформить в соответствии с формой, приведённой на рис. 3.4.

Пример идентификации стейкхолдеров мобильного приложения

Задание 3.4.

Выполнить описание сценариев работы стейкхолдеров с мобильным приложением в соответствии с исходными данными и решением заданий 3.1, 3.2 и 3.3.

Решение оформить в соответствии с формой, приведённой на рис. 3.5.

Задание 3.5.

Выполнить описание требований к мобильному приложению в соответствии с исходными данными и решением заданий 3.1, 3.2, 3.3 и 3.4. Решение оформить в соответствии с формой, приведённой на рис. 3.6.

Задание 3.6.

Осуществить идентификацию лидеров отрасли с использованием любого доступного источника информации. Анализ бизнеса конкурентов и места, которое в этом бизнесе занимает его сетевой ресурс, предлагается проводить с использованием инструмента стратегического управления, используемого для описания бизнес-моделей предприятий – бизнес-модели Остервальдера (Business Model Canvas).

Также следует описать преимущества и недостатки бизнес-модели конкурента, выделить используемые конкурентом в его бизнесе интересных идей, в т.ч. при использовании возможностей его сетевого ресурса.

Идентификацию лидеров отрасли и анализ их бизнес-модели осуществить в соответствии с исходными данными задания 3.1.

Задание 3.7.

Произвести идентификацию и определите приоритет возможностей концептуально решения мобильного приложения в соответствии с исходными данными и решением заданий 3.1, 3.2, 3.3, 3.4, 3.5 и 3.6.

Оформить результаты выполнения данного задания предлагается с использованием следующей таблицы:

Наименование возможности	*Краткое описание возможности*	*Приоритет возможности*	*Экспертное суждение о времени и стоимости реализации*		*Наличие функции у каждого конкурента*			
1	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*9*

Задание 3.8.

Произвести описания образа создаваемого или модифицированного мобильно решения по схеме:

1) целевая аудитория пользователей мобильного приложения или сервиса – «для»;

2) решаемые проблемы пользователя или предоставляемые возможности – «который»;

3) имя мобильного приложения или сервиса – «который»;

4) категория мобильного приложения или сервиса – «является»;

5) ключевое преимущество мобильного приложения или сервиса – «позволяет»;

6) основное конкурирующее мобильное приложение или сервис – «в отличие от»;

7) основное отличие мобильного приложения или сервиса – «за счет».

В качестве исходных данных использовать решение заданий 3.1, 3.2, 3.3, 3.4, 3.5, 3.6, 3.7 и 3.8.

Задание 3.9.

Разработайте стратегию управления жизненным циклом разработки мобильного приложения, образ которого был сформулирован при решении задания 3.9.

Задание 3.10.

Разработайте стратегию развертывания мобильного приложения, образ которого был сформулирован при решении задания 3.9.

Задание 3.11.

Разработайте стратегию сопровождения функционирования мобильного приложения, образ которого был сформулирован при решении задания 3.9.

Библиографический список

1. ГОСТ Р ИСО/МЭК 12207-2010. Информационная технология. Системная и программная инженерия. Процессы жизненного цикла программных средств. М., Стандартинформ, 2011

2. Зараменских, Е. П. Управление жизненным циклом информационных систем : учебник и практикум для академического бакалавриата / Е. П. Зараменских. — Москва : Издательство Юрайт, 2018. — 431 с. — (Бакалавр. Академический курс). — ISBN 978-5-9916-9200-7. — Текст : электронный // Образовательная платформа Юрайт [сайт]. — URL: https://urait.ru/bcode/413822 (дата обращения: 05.06.2022)

3. ISO/IEC/IEEE TR 24748-1 <1>, Systems and software engineering - Life cycle management - Part 1: Guide for life cycle management

4. Химонин Ю. И. Сбор и анализ требований к программному продукту [Электронный ресурс] // PMI.RU: портал. URL: https://pmi.ru/index.php/files/8/Documents/2/Software_Requirements_Khimonin.pdf (дата обращения: 02.05.2022)

5. Moore, Geoffrey A. 1991. Crossing the Chasm: Marketing and Selling High-Tech Products to Mainstream Cus-tomers. New York: Harper-Business

6. Технология разработки программного обеспечения: учеб. пособие / 15. В. Бахтизин, Л. А. Глухою. - Минск: БГУИР, 2010. - 267 с – ISBN 978-985-488-512-4

7. Грекул, В. И. Проектирование информационных систем : учебное пособие / В. И. Грекул, Г. Н. Денищен-ко, Н. Л. Коровкина. — 3-е изд. — Москва : Интернет-Университет Информационных Технологий (ИН-ТУИТ), Ай Пи Ар Медиа, 2020. — 299 с. — ISBN 978-5-4497-0689-8.

8. ISO/IEC 19505:2012. Информационные технологии. Унифицированный язык моделирования группы по управлению объектами (OMG UML)

9. Маглинец Ю.А. - Анализ требований к автоматизированным информационным системам - Национальный Открытый Университет "ИНТУИТ" - 2016 - 191с. - ISBN: 978-5-94774-865-9 - Текст электронный // ЭБС ЛАНЬ - URL: https://e.lanbook.com/book/100567

10. ГОСТ Р 57193-2016. Информационная технология. Системная и программная инженерия. Процессы жизненного цикла систем. М., Стандартинформ, 2017

11. ГОСТ Р 58607-2019. Информационная технология. Системная и программная инженерия. Управление жизненным циклом. Планирование системной инженерии. М., Стандартинформ, 2019

12. Guide to the Systems Engineering Body of Knowledge [Электронный ресурс] // sebokwiki.org: портал. URL: https://www.sebokwiki.org/w/images/sebokwiki-farm!w/6/66/SE-BoK_v_2.6_20220520.pdf (дата обращения: 02.05.2022).

13. Литвинов В.В., Богдан И.В., Задорожний А.А., Белоус И.В. Методы приоритизации задач в гибких методологиях разработки программного обеспечения // ММС. 2020. №2. URL: https://cyberleninka.ru/article/n/metody-prioritizatsii-zadach-v-gibkih-metodologiyah-razrabotki-programmnogo-obespecheniya (дата обращения: 27.05.2022).

14. Остервальдер А., Пинье И.- Построение бизнес-моделей: Настольная книга стратега и новатора - Альпина Паблишер - 2013 - ISBN: 9785961423457 - Текст электронный - URL: https://hse.alpinadigital.ru/book/351

15. ГОСТ 34.602-89. Информационная технология. Комплекс стандартов на автоматизированные системы. Техническое задание на создание автоматизированной системы. М., Стандартинформ, 2009

16. ГОСТ 42010-17. Информационная технология. Системная и программная инженерия. Описание архитектуры. М., Стандартинформ, 2017

17. Pyster, A., D. Olwell, N. Hutchison, S. Enck, J. Anthony, D. Henry, and A. Squires (eds). Guide to the Systems Engineering Body of Knowledge (SEBoK) version 1.0. — The Trustees of the Stevens Institute of Technology, 2012.

18. А. Косяков, У. Свит и др. Системная инженерия. Принципы и практика. Пер. с англ. по ред. В.К. Батоврина. ISBN 978-5-97060-464-9 - М.: ДМК Пресс, 2017.

19. Галимянов А.Ф., Галимянов Ф.А. Архитектура информационных систем / А. Ф. Галимянов, Ф. А. Галимя-нов. – Казань: Казан. ун-т, 2019. – 117 с

20. Ямских М. Е. Сравнительный анализ программ для создания мобильных приложений // Актуальные проблемы авиации и космонавтики. 2019. №. URL: https://cyberleninka.ru/article/n/sravnitelnyy-analiz-programm-dlya-sozdaniya-mobilnyh-prilozheniy (дата обращения: 10.05.2022).

21. Шаин Д.А. Тестирование юзабилити пользовательского интерфейса мобильных приложений // Вестник магистратуры. 2018. №5-3 (80). URL: https://cyberleninka.ru/article/n/testirovanie-yuzabiliti-polzovatelskogo-interfeysa-mobilnyh-prilozheniy (дата обращения: 29.05.2022).

Глава 4. МЕТОДИЧЕСКИЕ АСПЕКТЫ РАЗРАБОТКИ МОБИЛЬНЫХ ПРИЛОЖЕНИЙ

В данной главе рассматриваются следующие вопросы:

1. Общие принципы разработки мобильных приложений;

2. Методы анализа и проектирования мобильных приложений;

3. Методы разработки интерфейса мобильного приложения;

4. Методы разработки баз данных мобильных приложений.

4.1. Общие принципы разработки мобильных приложений

На разработку мобильных приложений существенное влияние оказывают следующие факторы:

1) особенности мобильных устройств;

2) особенности использования мобильных приложений пользователями;

3) особенности жизненного цикла мобильных приложений.

Мобильные устройства обладают следующими ключевыми особенностями [1]:

1) миниатюрность (имеют небольшой вес и размеры, а вместо клавиатуры – сенсорный экран с его виртуальным аналогом);

2) мобильность (возможность работы с информационными ресурсами без привязки к конкретной точке пространств);

3) ограниченность вычислительных ресурсов (мощности процессора, размер оперативной и постоянной памяти значительно уступает возможностям персональных компьютеров);

4) мультифункциональность (использование не только функции телефонных переговоров, но и фото- и видеокамеры, навигационного устройства, диктофона, переносной точки доступа, съемного носителя информации и др.);

5) большое по техническим характеристикам многообразие мобильных устройств (смартфоны разных размеров, планшеты с большими экранами, маленькие смарт-часы);

6) большое разнообразие используемых для управления устройствами версий операционных систем.

В качестве основных особенностей использования мобильных приложений пользователями называют [2]:

1) короткие сеансы работы (пользователи открывают мобильные приложения для быстрого просмотра информации: новых сообщений, быстрого поиска в Интернете, просмотра текущего местоположения для расчета маршрута, быстрой игры в общественном транспорте и т.д.);

2) активное использование функций системных приложений (отправка сообщений, фото и видеосъёмка, хранение файлов и т.д.);

3) активное использование интернет-сервисов и социальных сетей (возможность использования идентификации через социальные сети и интернет-сервисы, отправка сообщений и т.д.);

4) хранение информации в памяти устройства (например, фотографий, деловой переписки, технической документации и т.д.).

В качестве основных особенностей разработки мобильных приложений отметим:

1) небольшая продолжительность жизненного цикла (обычно обновления предоставляют новые функции наряду с исправлениями, которые мгновенно получают обратную связь от пользователей и могут быть улучшены в соответствии с требованиями пользователей и исправлены в следующих версиях мобильного приложения);

2) акцент на оптимизации использования ресурсов мобильного устройства (в первую очередь расходу батареи и памяти);

3) учет регулярного появления новых информационно-коммуникационных технологий;

4) использование облачных вычислений.

При разработке мобильных приложений необходимо также учитывать общие особенности проектирования программных систем [3]:

1) стремление к минимизации количества связей между отдельными подсистемами программной системы;

2) обеспечение максимальной связности отдельных частей внутри каждой подсистемы;

3) инкапсулирование содержимого каждой подсистемы;

4) определение для каждой подсистемы интерфейса взаимодействия с другими подсистемами;

5) применение CASE-средств управление жизненным циклом программного обеспечения.

Указанные выше особенности позволяют сформулировать основные правила и принципы разработки мобильных приложений.

Разработка мобильного приложения должна быть обоснована. При их проектировании, конструировании и развертывании следует использовать подход, ориентированный на потребителя, т.е. приложение должно быть: понятным, полезным, простым и соответствовать своему назначению.

Мобильные приложения не должны заменять другие каналы коммуникаций и продаж, а должны дополнять их. Контент мобильных приложений должен соответствовать контенту, доступному через другие каналы, т.е. являться частью многоканальной системы коммуникаций и продаж.

При выборе мобильного приложения в качестве одного из каналов коммуникаций и продаж необходимо учитывать предпочтения потребителей. Мобильное приложение должно создаваться только в том случае, если оно наилучшим или наискорейшим образом помогает людям решать свои проблемы.

При рассмотрении вопроса о создании мобильного приложения необходимо в качестве основной альтернативы рассматривать создание интернет-приложения, оптимизированного для работы с мобильных устройств, которое может выполнять многие из тех же функций, что и мобильное приложение, но при этом охватывать более широкую аудиторию.

В некоторых случаях требовать от потребителя найти приложение, загрузить его и затем работать с ним может быть слишком сложной для него задачей, что может привести к высокому уровню отказов. В этом случае работа с интернет-приложением для потребителя может быть более удобным вариантом, что приводит к более высокому уровню его вовлеченности.

Интернет-приложение может быть более подходящим вариантом в том случае, когда нет необходимости использовать основные аппаратные возможности мобильного устройства (например, акселерометр, сканер отпечатков пальцев и т.д.).

В том случае если есть необходимость использовать аппаратные возможности мобильного устройства, то необходимо рассмотреть вопрос создания гибридного мобильного приложения, которое более предпочтительнее нативного мобильного приложения, поскольку оно обычно обеспечивает повышенную гибкость и масштабируемость и требует более низких затрат на разработку и обслуживание.

Мобильные приложения должны соответствовать своему прямому назначению – обеспечению взаимодействия потребителя с сервисом правообладателя, что подразумевает:

1) наличие подтвержденного маркетинговыми исследованиями потребности потребителей в предоставляемом мобильным приложением сервисе и высокий спрос на него;

2) мобильное приложение повышает качество обслуживания потребителей, упрощая и повышая эффективность доступа к информации и услугам, а также совершения транзакций;

3) мобильное приложение позволяет взаимодействовать потребителю с правообладателем в любом месте и в удобное для него время;

4) предоставляемый мобильным приложением сервис имеет для потребителя добавленную стоимость (например, возможность использовать встроенную камеру телефона или глобальную систему позиционирования);

5) мобильное приложение связывает потребителя с сервисом и может обеспечить более персонализированный и своевременный для него подход;

6) мобильное приложение имеет ценность для потребителя, способствуя повторное частое использование сервиса (например, такие функции, как push-уведомления, отслеживание выполнения, информация в режиме реального времени, контекст на основе местоположения и т. д. могут способствовать повторному и постоянному использованию приложения);

7) поиск, загрузка и установка приложения выполняется легко и просто и не препятствует взаимодействию с сервисом.

При проектировании, разработке и развертывании мобильных приложений следует активно решать проблемы, связанные с безопасность и конфиденциальностью.

Доверие клиентов необходимо для максимального использования цифровых каналов коммуникации и продаж, и любое нарушение информационной безопасности может снизить доверие и поставить под угрозу цифровую трансформацию, инновации и предоставление онлайн-услуг правообладателем.

Для повышения уровня информационной безопасности необходимо разрабатывать мобильные решения с учетом:

1) запроса у целевой операционной системы наименьшего количества привилегий (например, не следует запрашивать доступ для записи к хранилищу данных устройства, за исключением случаев, когда это необходимо для выполнения мобильным приложением своих функций);

2) предоставления пользователям мобильного приложения четкого, конкретного и полного уведомления о том, как правообладатель будет использовать и раскрывать личную информацию, собранную мобильным приложением, включая функции устройства (например, видеокамеру), к которым приложение запрашивает доступ, и причины запроса этих разрешений (уведомление может быть осуществлено в форме кратких уведомлений с размещением важной информации со ссылками на более подробные пояснения или в форме использования графики, цвета или звука для привлечения внимания к уведомлениям);

3) включение в состав мобильного приложения средств контроля информационной безопасности;

4) обеспечение тестирования мобильного приложения на наличие уязвимостей на ключевых этапах его жизненного цикла и перед его развертыванием.

Мобильные приложения должно разрабатываться для широкого спектра платформ с учетом их потенциальной аудитории.

Все мобильные приложения должны как минимум быть развернуты на платформах iOS от Apple и Android от Google. Пользователи, которые не могут воспользоваться мобильным приложением, не должны быть ущемлены и должны иметь возможность получать соответствующую информацию или услуги по другим каналам.

Мобильное приложение должно быть разработано с учетом проблем с сетевым подключением. Потребители должны иметь возможность взаимодействовать с мобильным приложением как в зоне действия сети, так и за ее пределами. Например, если клиент заполняет форму в мобильном приложении без подключения к сети, он сможет отправить форму, как только вернется в зону действия сети. Если услуга основана на их геолокации, а отсутствие покрытия сети не позволяет сделать это автоматически, потребитель должен иметь возможность вручную выбрать свое местоположение и получить контекстную информацию (которая должна быть обновлены оперативными данными, если это возможно, когда они вернутся в зону покрытия сети).

Для безопасного и надежного распространения мобильных приложений должны использоваться их официальные агрегаторы, к которым в настоящий момент относят:

1) Apple: «App Store» (https://developer.apple.com/app-store/);

2) Google: «Google Play» (https://play.google.com/store);

3) Microsoft: «Store» (https://microsoft.com/store).

Иные практики распространения мобильных приложений следует избегать, чтобы предотвратить серьезные нарушения информационной безопасности (например, изменение целостности программной системы, фишинг и др.).

Корпоративные мобильные приложения, разработанные или приобретенные для корпоративного использования, не должны распространяться публично с использованием магазина приложений.

Для внутреннего распространения мобильных приложений необходимо рассмотреть возможность использования «корпоративных магазинов», который предоставляет правообладателю следующие преимущества:

1) конфиденциальность и безопасность (внутренние приложения и связанная с ними информация не находятся в открытом доступе);

2) ускоренный цикл выпуска (приложения можно быстро обновлять и распространять без длительных систем проверки: например, Apple проверяет приложения каждый раз, когда размещается обновление мобильного приложения);

3) консолидация (сторонние агрегаторы в дополнение к системам управления мобильными устройствами (MDM) и системам дополнительного управления мобильными приложениями (MAM) позволяют управлять и развертывать программные решения из корпоративных сетей).

При распространении мобильных приложений разработчику следует использовать одну и ту же учетную запись правообладателя для всех мобильных приложений. Это делается для того, чтобы потребители имели возможность легко находить другие приложения правообладателя.

Имена мобильных приложений в разных агрегаторах должны быть подобными друг другу. Описания приложений должны соответствовать их назначению: у потребителей должно быть очень четкое

представление о том, что делает приложение, какую пользу оно им принесет и почему им следует загрузить и использовать его.

Для улучшения качества мобильного приложения необходимо осуществлять мониторинг его функционирования. Для этого правообладатели должны включать в приложение подходящий код мониторинга и анализа (например, Apple iTunes Connect и Google App Analytics), который соответствует их бизнес-потребностям. Аналитика по мобильным приложениям может быть дополнена дополнительными аналитическими платформами (такими как Google Analytics 360, Adobe Marketing Cloud и т. д.).

Особое внимание следует необходимости развития и постоянного совершенствования мобильных приложений. При этом надо учитывать, что платформы мобильных устройств, операционные системы и браузеры часто обновляются, влияя на функциональность мобильного приложения, однако эти изменения не будут приводить к автоматическому обновлению приложения (пользователь сам должен его инициировать целенаправленно, что требует дополнительных усилий, которые навсегда он желает осуществлять).

Любое улучшение должно учитываться в мобильном приложении и поддерживаться количественными исследованиями и отзывами клиентов. Необходимо обеспечить различные варианты обратной связи с потребителями как в мобильном приложении, так и другими способами (корпоративный сетевой ресурс, службы поддержки по телефону и т.д.).

Одним из важных этапов разработки мобильных приложений является проектирование пользовательского интерфейса. Разработка мобильных приложений с простым в использовании интерфейсом имеет решающее значение для успешного внедрения и использования приложений [4].

Мобильные приложения (в первую очередь из-за непродолжительных сеансов работы пользователей с ними) должны иметь удобный пользовательский интерфейс с не очень глубокой структурой навигации, позволяющей после запуска приложения максимально быстро перейти на нужный экран.

Пользователь должен иметь доступ к общим функциям системных приложений (таким как отправка сообщений, создание заметок

или фотографирование), которые могут быть обеспечены одним из следующих способов:

1) вывод виджетов на домашний экран или быстрое меню с небольшой информацией;

2) текстовые или графические уведомления с необязательными быстрыми действиями, всегда доступные в панели уведомлений или на экране блокировки.

Мобильные приложения должны корректно обрабатывать возможные сворачивания и разворачивания экраны, корректно сохраняя данные пользователя и состояние.

Следует также отметить, что небольшой размер экрана значительно увеличивает нагрузку на дизайнеров и специалистов по эргономике. Каждый визуальный элемент и каждый жест пользователя следует выбирать с осторожностью. Следует учитывать не только размер элементов, но и доступность элементов на экране.

В 2018 году в Российской Федерации был разработан предварительный национальный стандарт [5], который обобщил характеристики качества мобильных приложений по следующим группам:

1) функциональность (соответствие заявленной функциональности, возможность пробного ознакомления, возможности приложения без авторизации, виджет мобильного приложения, поддержка ориентаций, выбор места хранения данных, очистка кеша, поддержка сервисов и расширений платформы, проигрывание аудиоконтента при заблокированном или выключенном дисплее, проигрывание аудиоконтента на домашнем экране, проигрывание аудиоконтента во время пользования другими приложениями, возможность настройки приложения, возможность отмены последнего действия);

2) удобство пользования (простота и качество навигации, демонстрация статус-панели устройства, приветственная инструкция, поддержка пользователей, политика конфиденциальности, соответствие дизайна руководствам платформы, использование стандартных жестов, уведомления приложения, качество русификации, использование единиц измерения соответствующего региона, адаптация приложения для людей с ограниченными возможностями, рекомендации по корректировке поиска, предупреждения о необратимых действиях и др.);

3) производительность и надежность (время запуска приложения, энергопотребление мобильного приложения, устойчивость к внешним

прерываниям, стабильность работы приложения, корректность работы приложения, корректность отображения элементов интерфейса);

4) безопасность (возможность установки пароля и входа по биометрическим данным, возможность удаления учетной записи, запрос только необходимых разрешений, наличие уязвимостей, наличие вредоносного программного обеспечения, безопасность передачи данных, безопасность обработки платежей, хранение персональных данных);

5) сопровождаемость (регулярность обновления, оперативность реагирования на критические программные ошибки и уязвимости, оперативность ответов на комментарии и вопросы пользователей);

6) переносимость (синхронизация данных, наличие адаптированной версии для планшета, наличие версии для умных часов, наличие веб-версии или версии приложения для персонального компьютера, наличие версии приложения для телевизора, наличие версии приложения для нескольких мобильных платформ);

7) информативность (информация о мобильном приложении, название приложения в магазине, название приложения на устройстве, информативность описания, наличие отзывов в описании, снимки экранов, видео в описании, информативность описания обновлений, ссылка на сайт разработчика, ссылка на политику конфиденциальности, регулярность обновлений, ответы разработчика).

4.2. Методы проектирования мобильных приложений

Основной проблемой, которую приходится решать при проектировании мобильных приложений, является уменьшение их сложности, за счет их представления как совокупности небольшого количества частей (подсистем), а их с свою очередь как элементы более меньшего размера и так далее.

Существуют два основных подхода к декомпозиции систем и, соответственно, к проектированию мобильных приложений:

1) функционально-модульный (предполагает описание структуры системы в виде иерархии её функций и передачи информации между ними);

2) объектно-ориентированный (предполагает описание структуры системы в виде объектов и передачи сообщений между ними).

Функционально-модульный подход относится к структурным методам анализа и проектирования, для которых характерно [3]:

1) разбиение системы на уровни абстракции с ограничением числа элементов на каждом из уровней (каждый элемент является «черным ящиком», количество таких элементов обычно от 3 до 7);

2) ограниченный контекст, включающий лишь существенные на каждом уровне детали;

3) использование строгих формальных правил записи;

4) итерационное приближение к конечному результату.

При использовании структурных методов для облегчения восприятия широко используют визуальное моделирование, которое предполагает построение моделей двух видов:

1) модели «AS-IS» («как есть»);

2) модели «AS-TO-BE» («как должно быть»).

Функционально-модульный подход к анализу и проектированию предполагает моделирование функциональной структуры системы, а также потоков информации и протекающих между ними процессов с помощью следующих методов построения моделей:

1) функционального моделирования (SADT, IDEF0) [6, 7, 8];

2) описания происходящих в системе процессов (IDEF3) [8, 9];

3) метода описания потока данных (DFD) [10].

В основе объектно-ориентированного подхода к анализу и проектированию систем лежит её представление как совокупности объектов, которые характеризуются:

1) состоянием (набором уникальных характеристик и их значениями в определенный момент времени);

2) связями между собой (в форме обмена сообщений);

3) поведением (наблюдаемой из вне деятельности в виде динамического изменения значений его характеристик и передаваемого им сообщений).

Основными принципами объектно-ориентированного подхода к анализу и проектированию систем являются: абстрагирование, инкапсуляция, модульность, иерархичность, типизация, параллелизм и сохраняемость.

Основными методами, используемыми в рамках объектно-ориентированного подхода к анализу и проектированию программных систем, являются [11]:

1) метод объектно-ориентированного системного анализа (ObjectOriented analysis, OOAS);

2) метод объектно-ориентированного анализа (ObjectOriented analysis, OOA);

3) метод структурного проектирования SD (Structured Design);

4) методология объектно-ориентированного анализа и проектирования (Object-oriented analysis and design, OOAD);

5) технология объектного моделирования (Object Modeling Technique, OMT);

6) объединенный метод UML;

7) метод определения распределенных объектов на основе объектной модели CORBA.

В настоящий момент широкое применение получил унифицированный язык моделирования (UML), реализующий одноименный метод.

Преимущества UML:

1) UML объектно-ориентирован, в результате чего методы описания результатов анализа и проектирования семантически близки к методам программирования на современных объектно-ориентированных языках;

2) UML позволяет описать систему практически со всех возможных точек зрения и разные аспекты поведения системы;

3) диаграммы UML сравнительно просты для чтения после достаточно быстрого ознакомления с его синтаксисом.

UML содержит стандартный набор диаграмм и нотаций самых разнообразных видов, которые можно объединить в следующие группы [12]:

1) моделирование использования (диаграммы вариантов использования);

2) моделирование структуры (диаграммы классов, диаграммы реализации, диаграммы компонентов, диаграммы размещения);

3) моделирование поведения (диаграммы состояний, диаграммы деятельности, диаграммы последовательности, кооперативные диаграммы).

Для моделирования требований к системе используются диаграммы вариантов использования (use case diagram) или даиграмма прецедентов.

Прецеденты – это технология определения функциональных требований к системе. Прецедент соответствует отдельному сервису системы, определяет один из вариантов её использования и описывает типичный способ взаимодействия пользователя с системой.

Основное назначение данной диаграммы заключается в описании функциональности и поведения мобильного приложения, позволяющее заказчику, конечному пользователю и разработчику совместно обсуждать проектируемую или существующую систему.

При моделировании системы с помощью диаграммы прецедентов системный аналитик стремится:

1) чётко отделить систему от её окружения;

2) определить действующих лиц (актёров), их взаимодействие с системой и ожидаемый функционал системы;

3) определить в глоссарии предметной области понятия, относящиеся к детальному описанию функционала системы (то есть, прецедентов).

Работа над диаграммой может начаться с текстового описания, полученного при работе с заказчиком. При этом нефункциональные требования (например, конкретный язык или система программирования) при составлении модели прецедентов опускаются.

Диаграммы прецедентов включают следующие элементы:

1) прецеденты;

2) действующее лицо;

3) связи между элементами.

Действующее лицо (actor, участники) – это роль, которую пользователь играет по отношению к системе. Действующие лица представляют собой роли, а не конкретных людей или наименования работ.

В UML действующие лица изображаются в виде стилизованных человеческих фигурок. Однако это не означает, что действующие лица обязательно являются людьми – действующие лица могут также быть внешними системами, которые взаимодействуют с данной системой, а также временем. Время становится действующим лицом, если от него зависит запуск каких-либо событий в системе.

Действующие лица находятся вне сферы действия разрабатываемой системы и не подлежат контролю со стороны разработчика.

Прецедент (вариант использования) представляет собой последовательность действий (транзакций), выполняемых системой в ответ на

событие, инициируемое некоторым внешним объектом (действующим лицом).

Вариант использования описывает типичное взаимодействие между пользователем и системой. В простейшем случае вариант использования определяется в процессе обсуждения с пользователем тех функций, которые он хотел бы реализовать.

В UML прецедент обозначается как эллипс с надписью, обозначающий выполняемые системой действия (могут включать возможные варианты), приводящие к наблюдаемым действующими лицами (актёрами) результатам. Надпись может быть именем или описанием (с точки зрения актёров) того, «что» делает система (а не «как»).

Имя прецедента связано с непрерываемым (атомарным) сценарием — конкретной последовательностью действий, иллюстрирующей поведение. В ходе сценария актёры обмениваются с системой сообщениями. Сценарий может быть приведён на диаграмме прецедентов в виде UML-комментария. С одним прецедентом может быть связано несколько различных сценариев.

Связи между элементами бывают следующих видов:

1) ассоциативное отношение (отношение ассоциации, association relationship);

2) отношение включения (include relationship);

3) отношение расширения (extend relationship);

4) отношение обобщения (generalization relationship).

Ассоциативные отношения служат для показа взаимосвязей между прецедентом и действующим лицом. Отношения расширения, включения и обобщения служат для показа взаимосвязей между прецедентами. Для демонстрации отношений между действующими лицами предназначены только обобщенные отношения.

Ассоциативные отношения служат для показа отношений между действующими лицами и прецедентами. Данный тип отношений устанавливает, какую конкретно роль играет действующее лицо при взаимодействии с экземпляром варианта использования.

В UML ассоциативные отношения показывают в виде прямой линии (пример ассоциативного отношения приведен на рис. 4.1). Каждый вариант использования должен быть инициирован действующим лицом; исключения составляют лишь варианты использования в связях использования и расширения.

Интерпретация: кассир и бухгалтер участвуют в оформлении первичных документов

Рис. 4.1. Ассоциативное отношение (association relationship)

Прямая линия может иметь ряд дополнительных обозначений:

1) направление связи (показывается стрелкой в направлении от инициатора связи);

2) кратность связи;

3) наименование связи.

Пример ассоциативного отношения с дополнительными обозначениями приведен на рис. 4.2.

Интерпретация: экономист составляет аналитические отчеты, учредитель является их потребителем

Рис. 4.2. Ассоциативное отношение с дополнительными обозначениями

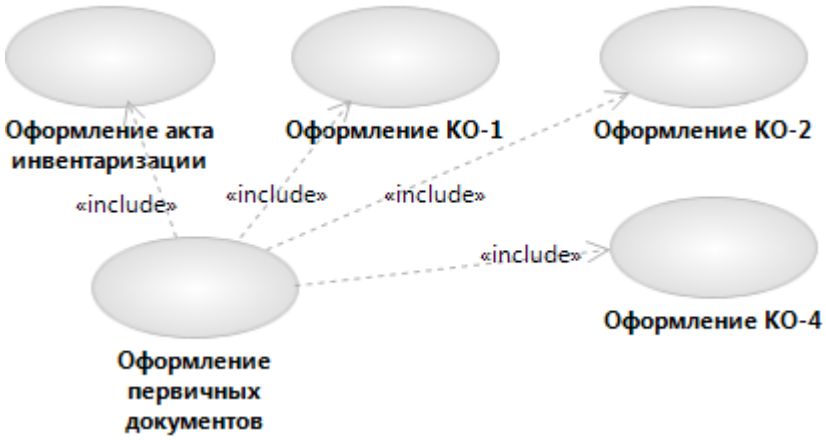
Отношения включения служат для показа взаимосвязей между прецедентами и позволяют одному прецеденту предоставлять функции другим прецедентам (т.е. указывает, что некоторое заданное поведение для одного варианта использования включается в качестве основного компонента в последовательность поведения другого варианта использования).

Такие отношения необходимы в двух случаях:

1) если несколько прецедентов имеют во многом идентичную функциональность, ее можно разделить на отдельные прецеденты, которые включаются (входят в состав) общего прецедента;

2) включающие отношения помогут в ситуации, когда один из прецедентов обладает необычно широкой функциональностью.

В UML отношения включения отмечаются пунктирной стрелкой со словом «include». Стрелка указывает в сторону включаемого варианта использования. Пример отношения включения приведен на рис. 4.3.



Интерпретация: оформление первичных документов предполагает оформление акта инвентаризации, оформление первичных кассовых документов по формам № КО-1, № КО-2 и № КО-4

Рис. 4.3. Отношение включения (include relationship)

Отношения расширения служат для показа взаимосвязей между прецедентами и позволяют при необходимости одному варианту использования дополнить свою функциональность за счет другого (т.е. определяют такую взаимосвязь базового варианта использования с некоторым другим вариантом использования, при которой функциональное поведение последнего задействуется базовым не всегда, а только при выполнении некоторых дополнительных условий).

В UML расширяющие отношения отмечаются пунктирной стрелкой со словом «extend». Стрелка указывает на базовый вариант использования. Пример отношения расширения приведен на рис. 4.4.

Интерпретация: оформление первичных документов предполагает при необходимости оформление первичных кассовых документов по формам № КО-3 и № КО-5

Рис. 4.4. Отношение расширения (extend relationship)

Отношения обобщения предназначены для показа общих черт прецедентов (или действующих лиц) и указывают на то, что некоторый прецедент (или действующее лицо) может быть обобщен до другого прецедента. В UML расширяющие отношения отмечаются стрелкой в направлении родительского прецедента (или действующего) лица.

Формировать обобщенные отношения не всегда необходимо. В общем случае они необходимы только при различиях в поведении разных действующих лиц с точки зрения анализа системы. Это же справедливо для прецедентов. Если есть некоторый набор функций, от которого отталкиваются несколько прецедентов, можно создать обобщенный прецедент и наследовать его свойства в других прецедентах через обобщенные отношения. Пример отношения обобщения приведен на рис. 4.5.

Моделирование функциональности мобильного приложения необходимо проводить, следуя следующим этапам:

1) идентификация участников, окружающих системы; необходимо выделить и провести анализ групп, заинтересованных в выполнении системой своих функций;

2) организация похожих участников с помощью отношений обобщения/специализации;

3) размещение участников на диаграмме прецедентов и определение их связи с прецедентами системы.

Пример диаграмма прецедентов для мобильного приложения по учету кассовых операций приведен на рис. 4.6.

Интерпретация: кассир, бухгалтер, экономист и руководитель – работник предприятия

Рис. 4.5. Отношение обобщения (generalization relationship)

При этом необходимо соблюдать ряд правил:

1) показывать на диаграмме прецедентов действующих лиц следует только в том случае, когда им действительно необходимы некоторые варианты использования;

2) моделировать связи между действующими лицами нельзя (действующие лица находятся вне сферы действия системы и поэтому связи между ними разработчику системы не подконтрольны);

3) соединять стрелкой непосредственно два варианта использования для определения порядка их выполнения нельзя (диаграмм прецедентов для этого не предназначена);

4) каждый вариант использования должен быть инициирован действующим лицом (исключением являются связи использования и расширения).

Диаграммы классов (class diagram) предназначены для моделирования статической структуры классов системы и связей между ними и представляет собой визуальное представление множество классов, интерфейсов, кооперации и отношений между ними.

Основными элементами диаграммы класса являются:

1) классы;

2) статические отношения между классами (связи).

Классы (шаблоны для создания объектов) на диаграмме классов отображаются в виде прямоугольников, которые разделены на три части, в которых соответственно отображается информация об имени класса, атрибутах класса и операциях класса (операции реализации, операции управления, операции доступа, вспомогательные операции).

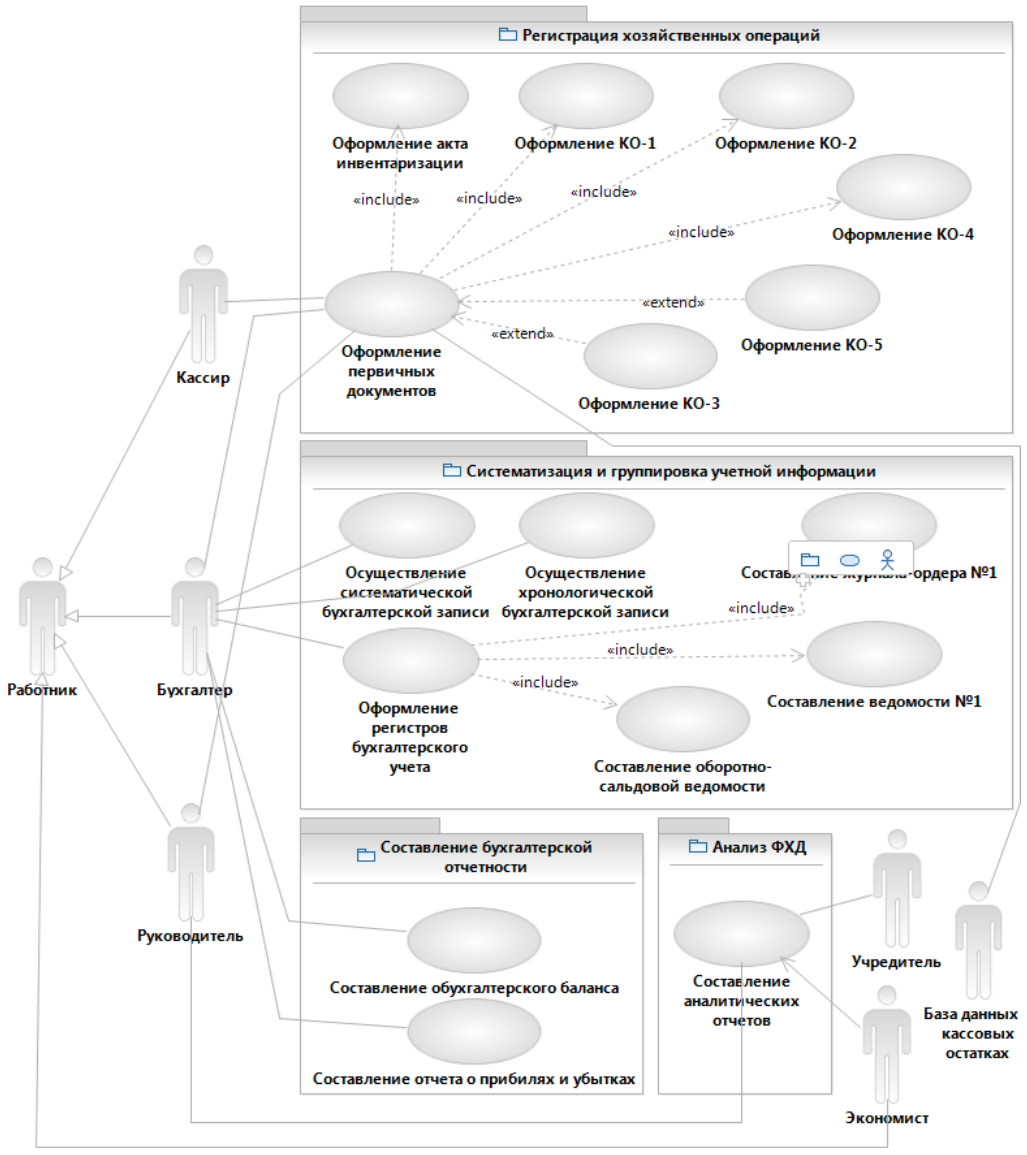


Рис. 4.6. Диаграмма прецедентов: мобильное приложение по учету кассовых операций

Существуют несколько типов связей между классами, которые отображаются с помощью различных вариантов линий: ассоциации, зависимости, агрегации и обобщения. Для связи может быть установлена её кратность (мощность), которая может принимать один из трех вариантов: «один-к-одному»; «один-ко-многим»; «многие-ко-многим».

Моделирование необходимо проводить, следуя следующим этапам:

1) идентификация участников, которые являются внешними по отношению к системе;

2) идентификация классов;

3) идентификация операций;

4) идентификация отношений связи.

Чтобы идентифицировать операции, необходимо выполнить следующие действия:

1) изучить спецификации прецедентов, диаграммы последовательности и кооперативные диаграммы (большая часть сообщений на этих диаграммах является операциями реализации, рефлексивные сообщения будут вспомогательными операциями);

2) рассмотреть управляющие операции (может потребоваться добавить конструкторы и деструкторы);

3) рассмотрите операции доступа (для каждого атрибута класса, с которым должны будут работать другие классы, надо создать операции Get и Set).

Пример диаграммы классов для мобильного приложения страховой компании на рис. 4.7.

Диаграмма последовательности (англ. sequence diagram) — диаграмма, на которой показано взаимодействие объектов (обмен между ними сигналами и сообщениями), упорядоченное по времени, с отражением продолжительности обработки и последовательности их проявления.

Основными элементами диаграммы последовательности являются:

1) обозначения объектов;

2) вертикальные "линии жизни";

3) обозначения фокусов управления;

4) обозначения сигналов и сообщений.

Объекты графически изображаются в форме прямоугольника и располагаются в верхней части своей линии жизни. Внутри прямоугольника записываются собственное имя объекта и имя класса, которые разделяются между собой двоеточием (вся запись подчеркивается).

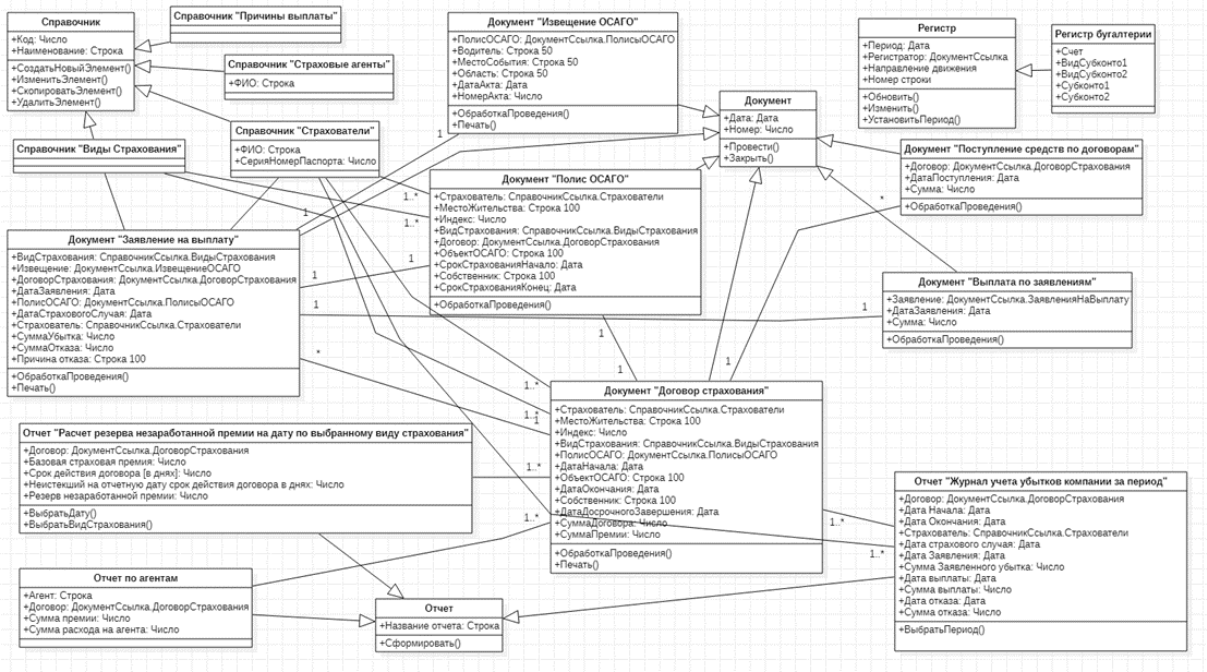


Рис. 4.7. Диаграмма классов мобильного приложения страховой компании

При этом на диаграмме последовательности может:

1) отсутствовать собственное имя объекта, но указывается имя класса (такой объект считается анонимным);

2) отсутствовать имя класса, но указывается собственное имя объекта (такой объект считается сиротой).

На данной диаграмме объекты располагаются слева направо. Порядок расположения объектов на диаграмме последовательности определяется исключительно соображениями удобства визуализации их взаимодействия друг с другом. Так объект, который является инициатором взаимодействия, изображается крайним слева. Другой объект, который непосредственно взаимодействует с ним, отображается чуть правее и т.д.

Линия жизни объекта (object lifeline) изображается пунктирной вертикальной линией, ассоциированной с единственным объектом на диаграмме последовательности. Линия жизни служит для обозначения периода, в течение которого объект существует в системе.

В случае если объект не создаётся в начальный момент времени, то прямоугольник такого объекта изображается не в верхней части диаграммы последовательности, а в той ее части, которая соответствует моменту времени создания объекта. При этом объект обязательно создается со своей линией жизни и возможно, с фокусом управления.

В случае если объект уничтожается, то линия жизни таких объектов обрывается в момент их уничтожения. Для обозначения момента уничтожения объекта используется специальный символ «X». ниже этой линии пунктирная линия не отображается.

Фокус управления (focus of control) – специальный, указывающий период времени, в течение которого объект выполняет некоторое действие, находясь в активном состоянии.

Фокус управления изображается в форме вытянутого по вертикали узкого прямоугольника (который на период активности объекта заменяет его линию жизни), верхняя сторона которого обозначает начало получения фокуса управления объекта (начало активности), а ее нижняя сторона - окончание фокуса управления (окончание активности).

Сообщение (message) – это средство, с помощью которого объект-отправитель запрашивает у объекта получателя выполнение одной из его операций.

Различают следующие виды сообщений: 1) информационное сообщение (informative message) – это сообщение, снабжающее объект-получатель некоторой информацией для обновления его состояния; 2) сообщение-запрос (interrogative message) – это сообщение, запрашивающее выдачу некоторой информации об объекте-получателе; 3) императивное сообщение (imperative message) – это сообщение, запрашивающее у объекта-получателя выполнение некоторых действий.

Сообщения изображаются горизонтальными стрелками, соединяющими линии жизни или фокусы управления двух объектов на диаграмме последовательности.

Сообщения можно разделить на 2 вида: синхронные (synchronous message) – требующие возврата ответа (объект, посылающий такие сообщения, прекращает работу до момента возврата ответа) и асинхронные (asynchronous message) – не требующие возврата ответа (объект, посылающий такие сообщения, не прекращает работу).

На диаграмме синхронные вызовы обозначаются сплошными линиями и закрашенными стрелочками, ответ на такой вызов – пунктирными линиями и закрашенными стрелочками. Асинхронные – не закрашенными или половинными стрелочками.

Сообщения могут иметь собственное имя (например, в качестве которого выступает имя операции, вызов которой инициируют эти сообщения у принимающего объекта). В этом случае рядом со стрелкой записывается имя операции с круглыми скобками, в которых могут указываться параметры или аргументы соответствующей операции. Если параметры отсутствуют, то скобки все равно должны быть изображены после имени операции.

При записи сообщений также могут использоваться стереотипы:

1) «call» (вызвать) — сообщение, требующее вызова операции или процедуры объекта-получателя;

2) «return» (возвратить) – сообщение, возвращающее значение выполненной операции или процедуры вызвавшему ее объекту, значение результата может инициировать ветвление потока управления;

3) «create» (создать) – сообщение, требующее создания другого объекта для выполнения определенных действий, созданный объект может стать активным (ему передается поток управления), а может остаться пассивным;

4) «destroy» (уничтожить) – сообщение с явным требованием уничтожить соответствующий объект (посылается в том случае, когда необходимо прекратить нежелательные действия со стороны существующего в системе объекта либо когда объект больше не нужен и должен освободить задействованные им системные ресурсы);

5) «send» (послать) – обозначает посылку другому объекту некоторого сигнала, который асинхронно инициируется одним объектом и принимается (перехватывается) другим (отличие сигнала от сообщения заключается в том, что сигнал должен быть явно описан в том классе, объект которого инициирует его передачу).

Для изображения ветвления потока управления изображаются две или более стрелки, выходящие из одной точки фокуса управления объекта. При этом рядом с каждой из стрелок должно быть явно указано соответствующее условие ветви в форме булевского выражения.

Алгоритм создания диаграмм последовательности выглядит следующим образом:

1) выявление всех объектов (и их классов) которые участвуют в моделируемом взаимодействии;

2) нанесение объектов на диаграмму с соблюдением порядка инициализации сообщений;

3) нанесения сообщений на диаграмму и их спецификация.

Пример диаграммы последовательности для мобильного приложения страховой компании (экран работы с договором страхования) на рис. 4.8.

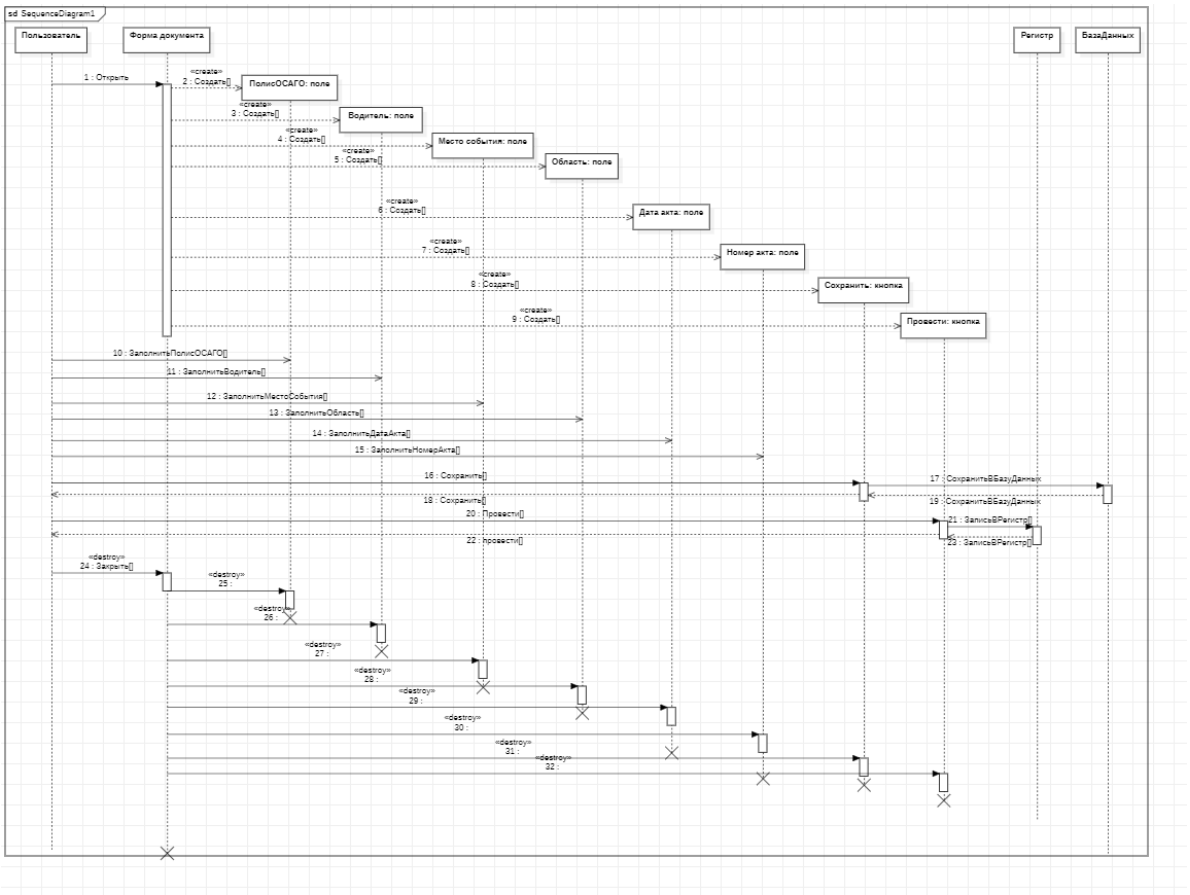


Рис. 4.8. Диаграмма последовательности для мобильного приложения страховой компании (экран работы с договором страхования)

Дадим краткую характеристику других наиболее часто используемых диаграмм UML:

1) диаграммы деятельности (activity diagram) используются для моделирования поведения системы в рамках различны вариантов использования или моделирования деятельности;

2) кооперативные диаграммы (collaboration diagram) используются для моделирования поведения объектов системы при переходе из одного состояния в другое;

3) диаграммы компонентов (component diagram) используются для моделирования иерархии компонентов (подсистем системы);

4) диаграммы размещения (deployment diagram) используются для моделирования физической архитектуры системы.

4.3. Методы разработки интерфейса пользователя мобильных приложений

Интерфейс пользователя мобильного приложения – интерфейс, обеспечивающий передачу информации между пользователем-человеком и программно-аппаратными компонентами мобильного устройства.

Различают следующие наиболее распространённые виды интерфейса пользователя:

1) командный (взаимодействие осуществляется с помощью строковых команд);

2) графический или WIMP-интерфейс (взаимодействие осуществляется в интерактивном режиме с визуальными формами, содержащими различные элементы управления: кнопки, надписи, поля ввода данных, пиктограммы и т.д.);

3) речевой или SILK-интерфейс (взаимодействие осуществляется в интерактивном режиме с помощью голоса).

В целом интерфейс пользователя можно разделить на два уровня: восприятие (зрение, осязание, слух и т. д.) и эмоции.

Дизайн интерфейсов – это работа, требующая больших профессиональных знаний: когнитивная психология, дизайн, лингвистика и так далее играют в ней ключевую роль.

Процесс разработки интерфейса мобильного приложения можно разделить на три составляющих:

1) исследование пользователей;

2) разработка дизайна взаимодействия с пользователем;

3) разработка дизайна интерфейса.

Исследование пользователей включает в себя два аспекта:

1) исследование юзабилити мобильного приложения, которое направлено на изучение методов улучшения удобства использования программного решения с целью более легкого принятия, использования и запоминания пользователем его интерфейса;

2) исследование потенциальных потребностей пользователей в области его взаимодействия с мобильным приложением с целью предложение новых инновационных интерфейсов их элементов.

Разработка дизайна взаимодействия с пользователем (UX-дизайн) предполагает создание привлекательных для пользователя мобильных приложений интерфейсов логичной и продуманной схемой поведения и действий.

Среди основных решаемых задач при разработке дизайна взаимодействия с пользователем:

1) разработка расположения и размера элементов с учетом недопущения перегрузки его внимания, а также положения руки пользователя на экране мобильного устройства;

2) разработка интуитивно понятной навигации (использование понятных пользователю зрительных образов для кнопок и символов, повторное использование элементов навигации на разных экранах мобильного приложения);

3) разработка первичных, вторичных и акцентных цветовых решений, самостоятельная разработка шрифтов или обоснование применение шрифтов сторонних производителей;

4) разработка экранных форм, минимизирующих использование виртуальной клавиатуры.

Создание UX-дизайна направлено улучшение мобильного приложения по одному или одновременно по нескольким основным направлениям:

1) увеличение длительности взаимодействия пользователя с приложением;

2) увеличение интенсивности взаимодействия пользователя с приложением;

3) формирование у пользователя положительных эмоций при работе с приложением;

4) снижение количества ошибок пользователя при работе с приложением;

5) повышение уровня обучаемости пользователя при работе с приложением.

Разработка дизайна интерфейса мобильного приложения (UI-дизайн) предполагает определение окончательных цветовых решений, типов и размеров шрифтов и т.д.

К основным принципам дизайна пользовательского интерфейса относят [13]:

1) принцип согласованности;

2) принцип удобства использования;

3) принцип стандартизации макета интерфейса;

4) принцип использования визуальных эффектов;

5) принцип унификации.

Принцип согласованности предполагает согласование внешнего вида элементов дизайна интерфейса между собой и согласование их поведения.

Внешний вид элемента управления влияет на взаимодействие пользователя с мобильным приложением. Однообразный внешний вид элементов управления не отвлекает внимание пользователи и позволяет им сосредоточиться на основных задачах, решаемых с помощью программного решения.

Поведение элемента управления также влияет на взаимодействие пользователя с мобильным приложением. Одинаковое поведение различных однотипных элементов приложения также не отвлекают пользователя от решения основных задач (например, все диалоги, требующие от пользователя подтверждения действия, должны содержать минимум две кнопки: одна для подтверждения действия, вторая - для отмены).

Принцип удобства использования мобильного приложения предполагает понятность его интерфейса и соответствие элементов управления требуемым пользователями функциям.

Для того, чтобы мобильное решение могло комфортно использоваться, пользователи должны быть в состоянии понять соответствующие функции каждого используемого элемента программного обеспечения. Для этих пользователей при первом использовании приложения может инициироваться подпрограмма обучения, суть которой заключается в демонстрации основных функций приложения в интерактивном режиме. Также необходимо использовать узнаваемые элементы, которые применяются в аналогичных приложениях [14].

Пользователи мобильного приложения должны полностью контролировать свою работу с ним. Для этого необходимо, чтобы все элементы управления соответствовали требуемым пользователями функциям.

Принцип стандартизации макета интерфейса мобильного приложения предполагает при его разработке мобильного использование нескольких стандартных информационных областей, на которые делится каждый экран:

1) область заголовка;

2) функциональная рабочая область;

3) область навигации.

Область заголовка предполагает размещение логотипа программного обеспечения и информации о версии программного обеспечения.

Функциональная рабочая область – самая большая часть макета, представляющая собой основную часть программного обеспечения, в рамках которой реализуются все функциональные требования к мобильному приложению.

Область навигации: это область перехода между отдельными частями приложения, доступная на всех его экранах. Также данная область используется для размещения элементов управления с помощью, которых сохраняются результатов текущей операции, а также завершается работы с приложением.

Принцип использования визуальных эффектов предполагает стимуляцию зрительных ощущений пользователя при работе с мобильным приложением, обеспечивая психологический комфорт такой работы. Достигается это за счет простоты и оригинальности используемых визуальных эффектов, а также создания интереса к ним.

Принцип унификации предполагает использование единого стиля оформления для всех элементов управления в мобильном приложении [15].

Основными методами, которые используются при разработке интерфейса пользователя мобильного приложения являются (в порядке их применения) [16]:

1) моделирование пользовательских сценариев (user flow);

2) создание набросков основных элементов (sketches);

3) создание шаблонов интерфейса экранов (wireframe);

4) создание макета интерфейса;

5) создание интерактивного прототипа.

Пользовательский сценарий (User Flow) представляет собой визуальное представление последовательности действий пользователя

мобильного приложения, которые он должен выполнить для достижения своей цели.

Разработка пользовательского сценария мобильного приложения осуществляется на основе разработанных на этапе анализа функциональных требований к программному решению и включает в себя следующие этапы [17]:

1) идентификация пользователя;

2) идентификация цели пользователя;

3) идентификация последовательности действий пользователя для достижения своей цели.

В зависимости от необходимой степени детализации пользовательский сценарий может быть представлен в одной из следующих форм:

1) поток задач (или task flow): приводится информация об основных действиях пользователя на каждом шаге на пути достижения цели;

2) поток схем (или wire flow): приводится как информация об основных действиях пользователя на каждом шаге на пути достижения цели, так и схематический набросок дизайна каждого экрана (wireframes), демонстрирующий его структуру и расположение на нём основных управляющих элементов (кнопок, надписей, полей и т.д.);

3) поток экранов (или wire flow): приводится как информация об основных действиях пользователя на каждом шаге на пути достижения цели, так и детально проработанные экраны мобильного приложения.

Набор инструментальных средств для разработки пользовательских сценариев в настоящий момент широко представлен. Для совместной работы может использоваться, например, Figma, Overflow, Miro, Flowmapp и другие.

Пример пользовательского сценария мобильного приложения в форме потока задач приведен на рис. 4.9.

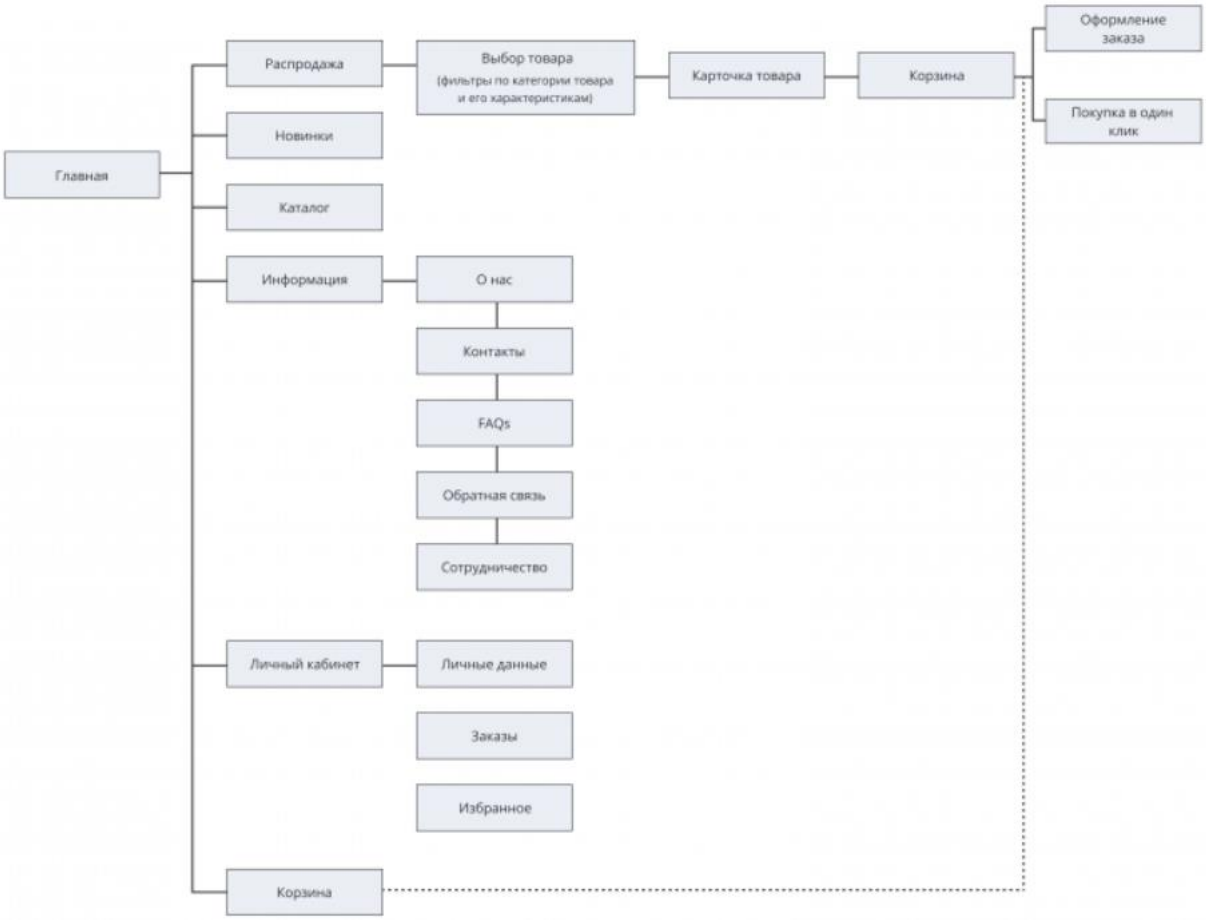


Рис. 4.9. Пример пользовательского сценария мобильного приложения

Полученные пользовательские сценарии в виде поток задач требуется представить в графической форме. Для этого осуществляют прототипирование пользовательского интерфейса, т.е. создание дизайна экранов мобильного приложения разной степени детализации, в частности прорабатываются:

1) наброски «на скорую руку» (скетчинги);

2) схемы экранов (вайрфреймы);

3) интерактивные макеты.

Для предварительной демонстрации идей дизайна интерфейса используют наброски от руки (скетчинги). Их цель дать общее представление о внешнем виде экранов мобильного приложения.

Пример использования скетчинга совместно с пользовательским сценарием приведен на рис. 4.10.

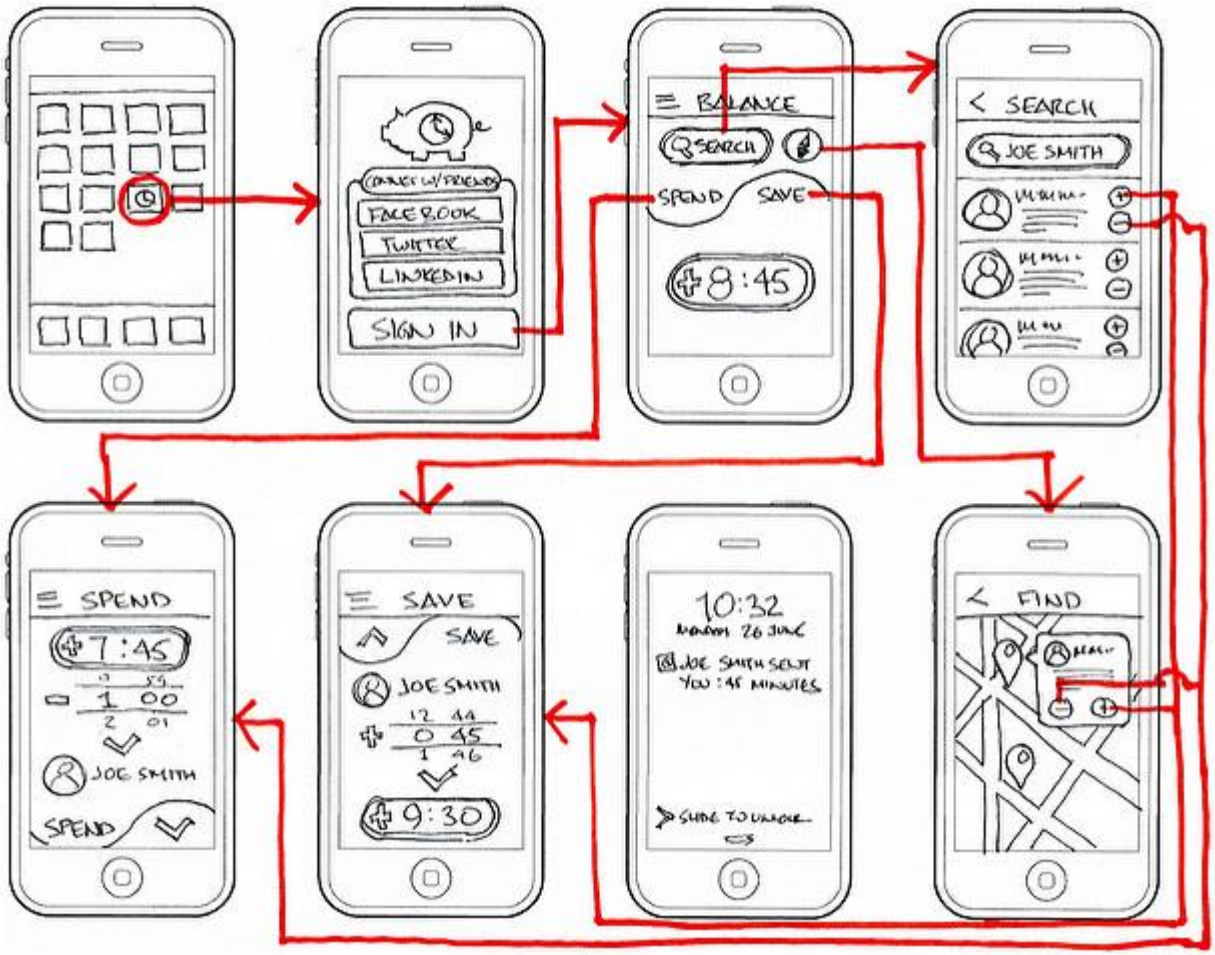


Рис. 4.10. Пример предварительной демонстрации идей дизайна интерфейса в форме скетчинга

Для моделирования структуры экранов и интуитивно понятного перемещения пользователя между ними прорабатываются шаблоны интерфейса экранов мобильного приложения (вайрфреймы). Элементы управления и навигации на них прорабатываются с учетом их ассоциативного восприятия пользователем. Схемы экранов не содержат мелких деталей и цветовых решений. Расстановка элементов на экранах осуществляется для каждого размера экрана в вертикальном и горизонтальном расположении.

Пример вайрфреймов экранов мобильного приложения приведен на рис. 4.11.

Рис. 4.11. Пример моделирования структуры экранов в форме вайрфреймов

После утверждения вайрфреймов осуществляется:

1) разработка концепции дизайна экранов мобильного приложения (т.е. основных цветовых решений, используемых шрифтов, внешнего вида элементов и т.д.);

2) создание готового к верстке макета интерфейса мобильного приложения (осуществляется детальная отрисовка всех экранов и всех расположенных на них элементов до возможности).

На основании макета интерфейса мобильного приложения может быть создан его интерактивный прототип, который предоставляет возможность имитации работы с программным решением (осуществлять нажатие кнопок, выбирать элементы из списка, переходить между экранами и т.д.).

4.4. Методы разработки баз данных мобильного приложения

Корпоративное мобильное приложение является, как правило, одним из элементов информационной системы предприятия и имеет, как правило, трехуровневую архитектуру, в рамках которой мобильное приложение используется как клиент, вся логика работы которого реализуется на сервере приложений, а данные хранятся на сервере баз данных. Такой подход позволяет экономить ограниченные мощности и ресурсы мобильного устройства.

Шаблон архитектуры простой трехуровневой информационной системы с использованием мобильного приложения в качестве клиента был представлен на рис. 3.1.

Данные мобильных приложений конечного пользователя также могут хранить данные на удаленном сервере. Однако для небольших приложений является целесообразным использование локальное размещение необходимых для работы мобильного приложения данных.

Для обеспечения работы мобильного приложения при наличии проблем с сетевым подключением является целесообразным наличие локальной реплики базы данных или её части на устройстве.

В настоящее время широкое применение получило размещение сервера приложений и сервера баз данных в рамках единых облачных решений поставщиков BaaS-услуг (например, Google Firebase, IBM Cloud Functions, Microsoft Azure и др.) [18].

BaaS-сервисы предлагают свои пользователям инструментарий работы для основных мобильных платформ, с помощью которого мобильное приложение может получить доступ к разнообразным функциям хранения и обработки данных, push-уведомлений, геолокации и т.д. Это позволяет значительно ускорить процесс разработки мобильного приложения, но при этом в процессе эксплуатации такой подход может не справляться с ростом нагрузок.

При проектировании взаимодействия мобильного приложения и сервера приложения (или BaaS-сервиса) для эффективного обмена данными широко используется архитектурный стиль REST [19].

База данных – организованная в соответствии с определёнными правилами и поддерживаемая в памяти компьютера совокупность данных, характеризующая актуальное состояние некоторой предметной

области и используемая для удовлетворения информационных потребностей пользователей.

К организации базы данных предъявляются определённые требования: неизбыточность, непротиворечивость, независимость от приложений (правило «Три НЕ»).

База данных неизбыточна, если удаление какого-либо элемента данных ведет к потере информации о предметной области; база данных непротиворечива, если все хранящиеся в ней данные удовлетворяют определенным условиям; база данных независима от приложений, если существует возможность осуществлять реструктуризацию базы данных или ее реорганизацию, не меняя при этом приложения.

Для обеспечения независимости базы данных от приложений используют системы управления базами данных. Система управления базами данных (СУБД) – совокупность языковых и программных средств, предназначенных для создания, ведения и совместного использования базы данных многими пользователями (ГОСТ Р ИСО МЭК ТО 10032-2007).

СУБД обеспечивает выполнение двух групп функций:

1) предоставление среды взаимодействия пользовательского приложения и базы данных;

2) поддержка базы данных на всех этапах ее жизненного цикла.

СУБД должна предоставлять возможность доступа к данным любым пользователям, включая и тех, которые практически не имеют и (или) не хотят иметь представления о:

1) физическом размещении в памяти данных и их описаний;

2) механизмах поиска запрашиваемых данных;

3) проблемах, возникающих при одновременном запросе одних и тех же данных многими пользователями (прикладными программами);

4) способах обеспечения защиты данных от некорректных обновлений и (или) несанкционированного доступа;

5) поддержании баз данных в актуальном состоянии и множестве других функций СУБД.

В общем случае с одной базой данных могут работать множество различных приложений (именно СУБД призвана обеспечить работу множества приложений с единой базой данных таким образом, чтобы каждое из них выполнялось корректно, но учитывало все изменения в базе данных, вносимые другими приложениями).

Жизненный цикл базы данных – непрерывный процесс, началом которого становится момент принятия решения о необходимости базы данных, а завершением – ее изъятие из эксплуатации. Поддержка базы данных на всех этапах ее жизненного цикла предполагает возможность проектирования, создания и эксплуатации (ведения, использования, модернизации и развития).

Таким образом, СУБД имеет два режима работы:

1) проектировочный – предназначен для создания или изменения структуры базы и создания её объектов;

2) пользовательский – использование ранее подготовленных объектов для наполнения базы или получения данных из нее.

Соответственно, процесс проектирования баз данных представляет собой последовательность переходов от неформального словесного описания информационной структуры предметной области к формализованному описанию объектов предметной области в терминах некоторой модели данных.

Цель процесса проектирования баз данных состоит в получении проекта базы данных, который бы:

1) адекватно отображал предметную область;

2) удовлетворял информационным требованиям пользователей;

3) соответствовал правилу «Три НЕ» (неизбыточность, непротиворечивость, независимость от приложений).

В общем случае, можно выделить следующие этапы проектирования:

1) предпроектное обследование предметной области;

2) семантическая структуризация предметной области;

3) выбор правил структурирования данных и инструментария;

4) логическая структуризация данных;

5) физическая структуризация данных.

Предпроектное обследование предметной области предполагает восприятие, изучение и описание информационных процессов предметной области, а также выявление информационных потребностей пользователей и формулирование требований к содержанию и обработке данных.

Семантическая структуризация предметной области предполагает:

1) неформализованное словесное описание предметной области с использованием таблиц, формул, схем;

2) описание связей между данными с использованием некоторой формализованной системы нотаций (системы условных обозначений, языка зависимостей).

Указанные связи являются основой выбора модели данных и проектных решений относительно структуризации данных.

Выбор правил структурирования данных и инструментария предполагает выбор конкретной модели данных (сетевой, иерархической, реляционной и т.д.).

Логическая структуризация данных предполагает описание структуры базы данных в терминах выбранной модели данных. При этом построенная модель является машинно-независимой, а также независимой от конкретной системы управления базами данных.

Физическая структуризация данных предполагает:

1) выбор конкретной системы управления базы данных для построения базы данных;

2) описание структуры базы данных в терминах языка манипулирования данных выбранной системы управления базы данных.

На данном этапе определяются носители, методы доступа и способы защиты данных и др.

Инфологические (или семантические) модели – это модели концептуального уровня, выражающие информацию о предметной области в виде, независимом от используемой системы управления базами данных.

Целью инфологического проектирования является обеспечение наиболее естественных для человека способов сбора и представления той информации, которую предполагается хранить в создаваемой базе данных.

Инфологическая модель должна обладать тремя важными свойствами:

1) она должна быть согласованной с инфраструктурой бизнеса и верной во всех сферах применения;

2) при ее расширении новые данные должны определяться без изменения ранее определенных;

3) она должна удобно адаптироваться как к точкам зрения пользователей, так и к многообразию структур хранения данных и доступа к ним.

Наиболее широко распространённым методом создания семантических моделей является модель «сущность-связь» (ERM). В основе ER-модели лежит деление реального мира на отдельные различимые объекты (сущности), обладающие определенными характеристиками (атрибутами) и находящиеся в определенных связях друг с другом. Соответственно основными конструктивными элементами ER-модели модели являются: сущность, атрибут и связь.

В настоящий момент не существует единой общепринятой системы обозначений для конструктивных элементов ER-модели и разные авторы изданий и CASE-средства применяют разные графические обозначения для конструктивных элементов ER-модели (нотации): например, нотацию Чена, нотацию Мартина, нотацию IDEF1X, нотацию IE, нотацию Баркера, и др.

Проведение концептуального (инфологического) инфологического моделирования предметной области предполагает выполнение следующих действий:

1) идентификация и описание сущностей;

2) определение и описание атрибутов;

3) определение и описание отношений связи между сущностями.

Целью этапа идентификации и описания сущностей является выявление и определение отдельных различимых объектов предметной области (сущностей).

Результатом данного этапа являются:

1) глоссарий сущностей ER-модели;

2) изображение блоков сущностей на ER-диаграмме.

Сущность – множество реальных или абстрактных предметов (людей, объектов, мест, событий, состояний, идей, пар предметов и т.д.), обладающих общими атрибутами или характеристиками. Отдельный элемент этого множества называется экземпляром сущности.

Сущности могут быть независимыми или зависимыми. Сущность является независимой, если каждый экземпляр сущности может быть однозначно идентифицирован без определения его отношений с другими сущностями. Сущность называется зависимой, если однозначная

идентификация экземпляра сущности зависит от его отношения к другой сущности (т.е. независимая сущность не нуждается в информации из другой сущности для идентификации уникального экземпляра).

Каждой сущности присваивается уникальное имя. Именем сущности является грамматический оборот существительного (существительное, у которого могут быть прилагательные и предлоги), описывающий представляемое сущностью множество предметов.

Существительное должно употребляться в единственном, а не во множественном числе. Сокращения и акронимы допускаются, но при этом имя сущности в модели должно быть осмысленным и согласованным. Формальное определение сущности, а также список синонимов или псевдонимов должны быть приведены в глоссарии модели. Хотя одна и та же сущность может быть изображена на любом числе диаграмм, на каждой конкретной диаграмме она должна быть представлена только один раз.

Целью этапа идентификации и описания атрибутов является их выявление и документирование. Результатом данного этапа является:

1) глоссарий атрибутов;

2) определение первичных ключей сущностей ER-модели;

3) изображение атрибутов сущностей на диаграмме.

Атрибут – поименованная характеристика сущности, используемая для определения того, какая информация должна быть собрана о сущности. Наименование атрибута должно быть уникальным для конкретного типа сущности, но может быть одинаковым для различного типа сущностей.

Сущность может обладать одним или несколькими атрибутами, которые либо принадлежат сущности, либо наследуются через отношение связи. Ни один из экземпляров сущности не может обладать более чем одним значением для связанного с сущностью атрибута.

Атрибуты могут быть простые и составные. Составной атрибут – это атрибут, который может быть в дальнейшем разделен на несколько простых атрибутов.

Атрибуты могут быть однозначные и многозначные. Однозначный атрибут – это такой атрибут, который может принимать единственное значение. Многозначный атрибут – это атрибут, который может принимать несколько значений.

С целью формального определения атрибутов (имени, описания, способа реализации) составляется глоссарий атрибутов.

Уникальный идентификатор сущности – это минимальный набор атрибутов, по значениям которых можно однозначно найти требуемый экземпляр сущности. Минимальность означает, что исключение из набора любого атрибута не позволяет идентифицировать сущность по оставшимся атрибутам. Такой идентификатор также называют возможным идентификатором сущности.

При существовании нескольких возможных (потенциальных) идентификаторов один из них обозначается в качестве первичного, а остальные – альтернативных идентификаторов. Если существует только один возможный идентификатор, то он является, конечно, первичным идентификатором.

При выборе первичного идентификатора из набора возможных учитываются следующие факторы:

1) частота использования (желательно выбирать в качестве первичного идентификатора атрибуты, которые чаще всего используются);

2) длина идентификатора (в качестве первичного идентификатора желательно выбирать самый короткий из возможных, а также тот, в который входит минимальное количество атрибутов (оптимально – атомарный атрибут));

3) стабильность (желательно выбирать в качестве первичного идентификатора атрибуты, которые не изменяются);

4) мнемоничность (при прочих равных условиях следует отдавать предпочтение тем из возможных идентификаторов, которые легче запомнить или с которыми легче работать (например, при выборе символьных атрибутов в качестве первичных идентификаторов часто возникают проблемы с вводом ошибочных значений, в числовых атрибутах вероятность ошибки при вводе значения минимальна)).

Если в сущности нет ни одной комбинации атрибутов, подходящей на роль потенциального идентификатора, или потенциальные идентификаторы не отвечают критериям выбора из их состава первичного идентификатора, то в сущность добавляют отдельный атрибут – суррогатный идентификатор (искусственный ключ). Как правило, для такого атрибута выбирают числовой тип данных. Также стоит отме-

тить, что некоторые разработчики вместо поиска потенциальных идентификаторов и выбора из них первичного в каждую сущность добавляют искусственный атрибут, который в дальнейшем и используют в качестве первичного идентификатора.

Целью этапа идентификации и описания отношений связей является выявление и определение основных отношений связей между сущностями. Результатом данного этапа являются:

1) матрица отношений связи;

2) глоссарий отношений связи;

3) изображение линий связей на диаграмме

Отношение связи – это ассоциация между сущностями, при которой каждый экземпляр одной сущности ассоциирован с произвольным (в том числе нулевым) количеством экземпляров второй сущности.

Различают классы связей и экземпляры связей. Классы связей – это взаимоотношения между классами сущностей, а экземпляры связи – взаимоотношения между экземплярами сущностей.

Связь характеризуется следующим набором параметров: именем, кратностью, типом, классом принадлежности, степенью участия.

Именем отношения связи является грамматический оборот глагола или отглагольного существительного описывающие то, как соотносятся сущности между собой.

Кратность (мощность, кардинальность, степень) связи – число экземпляров сущностей, которое может быть ассоциировано через отношение связи с экземплярами другой сущности.

Различают следующие кратности связи:

1) связь 1:1 (одиночный экземпляр сущности одного типа связан с одиночным экземпляром сущности другого типа);

2) связь 1:M (единый экземпляр сущности одного типа связан со многими экземплярами сущности другого класса);

3) связь M:N (несколько экземпляров сущности одного класса связаны с несколькими экземплярами сущности другого класса).

Различаю следующие типы отношения связи:

1) идентифицирующая (связь от зависимой сущности к независимой);

2) неидентифицирующая (связь между независимыми сущностями).

Другой важной характеристикой отношения связи помимо ее степени является класс принадлежности входящих в нее сущностей. Существует обязательный и необязательный классы принадлежности.

Если каждый экземпляр одной сущности (А) связан с экземпляром другой сущности (В), то класс принадлежности сущности А является обязательным. Если не каждый экземпляр одной сущности (А) связан с экземпляром другой сущности (В), то класс принадлежности сущности А является необязательным.

Степень участия – количество сущностей, участвующих в связи. Связь, существующая между двумя сущностями, называется бинарной связью. Связь, существующая между n сущностями, называется n-арной связью. Рекурсивная связь – это связь между экземплярами одной сущности.

Для определения связей, которые наблюдаются между ранее определенными сущностями, используется матрица отношений, которая представляет собой двумерный массив, вдоль осей которого записываются имена сущностей. При наличии связи между сущностями на пересечении их осей ставится знак «Х».

С целью формального определения отношений связи (её имени, описания, степени и класса принадлежности) составляется глоссарий отношений связи между сущностями

Рассмотрим в качестве примера концептуальное проектирование базы данных мобильного приложения, которое предусматривает возможность ведения преподавателем учета информации об оценках определенных студентов из определенных студенческих групп по определенным контрольным мероприятиям (рейтингам, зачетам, экзаменам и т.д.) по определенным учебным дисциплинам.

В процессе разработки инфологической модели были выявлены следующие сущности: студенческая группа, студент, контрольное мероприятие, дисциплина, оценка. Список сущностей предметной области, их синонимы, а также формальное описание приведены на рис. 4.12.

В процессе разработки инфологической модели были выявлены атрибуты каждой сущности. Результаты идентификации атрибутов сущностей и их определение приведены на рис. 4.13.

Для каждой сущности определены потенциальные идентификаторы и с целью выбора первичного идентификатора произведена

оценка каждого их них с точки зрения его частоты использования, длины, стабильности и мнемоничность.

Наименование сущности (синоним сущности)	*Описание сущности*
Студенческая группа (Группа)	*Объединение студентов для общего участия в учебной деятельности*
Студент	*Учащийся ВУЗа*
Контрольное мероприятие (Форма контроля)	*Форма текущего контроля или промежуточной аттестации*
Дисциплина	*Самостоятельная отрасль знаний, изучаемая студентами*
Оценка	*Результат проведенного контрольного мероприятия*

Рис. 4.12. Глоссарий сущностей ER-модели

Сущность	*Наименование атрибута (синоним)*	*Описание атрибута (способ реализации, значение по умолчанию, диапазон возможных значений)*
Группа	*Наименование*	*Принятое в ВУЗе наименование студенческой группы*
	Куратор	*Фамилия, имя и отчество куратора студенческой группы*
Студент	*Фамилия*	*Фамилия студента*
	Имя	*Имя студента*
	Отчество	*Отчество студента*
	Группа	*Студенческая группа, в которой состоит студент (реализуется через связь с сущностью «Группа»)*
Контроль	*Наименование*	*Принятое в Вузе наименование формы текущего контроля или промежуточной аттестации*
	Максимум	*Минимальная оценка за контрольное мероприятие*
	Минимум	*Максимальная оценка за контрольное мероприятие*
Дисциплина	*Наименование*	*Наименование дисциплины по учебному плану*
	Часы	*Количество часов дисциплины по учебному плану*
Оценка	*Дата*	*Дата сдачи контрольного мероприятия*
	Студент	*Студент, знания которого оцениваются (реализуется через связь с сущностью «Студент»)*
	Дисциплина	*Дисциплина, по которой проводится оценка знаний (реализуется через связь с сущностью «Дисциплина»)*
	Контроль	*Форма контроля, с помощью которой оцениваются знания (реализуется через связь с сущностью «Контроль»)*
	Результат	*Отметка по результатам сдачи контрольного мероприятия*

Рис. 4.13. Глоссарий атрибутов сущностей ER-модели

При несоответствии всех потенциальных идентификаторов сущности одновременно 4 критериям в качестве первичного идентификатора выбирается искусственно созданный атрибут «Код». Результаты идентификации потенциальных идентификаторов и выбора первичного идентификатора для каждой сущности приведены на рис. 4.14.

Сущность	*Потенциальный идентификатор*	*Критерий выбора (1 - частота использования, 2 - длины, 3- стабильности, 4 - мнемоничность)*				*Первичный идентификатор*
		1	*2*	*3*	*4*	
Группа	*Наименование*	*да*	*нет*	*да*	*да*	*нет*
	Код	*да*	*да*	*да*	*да*	*да*
Студент	*Фамилия + Имя*	*да*	*нет*	*нет*	*да*	*нет*
	Код	*да*	*да*	*да*	*да*	*да*
Контроль	*Наименование*	*да*	*нет*	*да*	*да*	*нет*
	Код	*да*	*да*	*да*	*да*	*да*
Дисциплина	*Наименование*	*да*	*нет*	*да*	*да*	*нет*
	Код	*да*	*да*	*да*	*да*	*да*
Оценка	*Дата + Студент + Дисциплина + Контроль*	*да*	*нет*	*нет*	*нет*	*нет*
	Код	*да*	*да*	*да*	*да*	*да*

Рис. 4.14. Определение первичных идентификаторов сущностей ER-модели

В процессе разработки инфологической модели были выявлены и описаны отношения связи между сущностями.

Результаты идентификации атрибутов сущностей приведены на рис. 4.15. Фрагмент описания отношений связи между сущностями приведены в табл. 4.16.

Результатом разработки инфологической модели является концептуальная ER-диаграмма, приведенная на рис. 4.17.

Под даталогической понимается модель, отражающая логические взаимосвязи между элементами данных безотносительно их содержания и физической организации. При этом даталогическая модель разрабатывается с учётом специфики конкретной предметной области на основе её инфологической модели, а также с учетом специфики конкретной модели данных, которую поддерживают СУБД.

Сущности	*Группа*	*Студент*	*Контроль*	*Дисциплина*	*Оценка*
Группа		*X*			
Студент	*X*				*X*
Контроль					*X*
Дисциплина					*X*
Оценка		*X*	*X*	*X*	

Рис. 4.15. Матрица отношений связи между сущностями ER-модели

Наименование отношений связи	*Определение отношений связи*
Студент – Группа	*Бинарная связь «Студент состоит в группе».* *Каждый студент обязан состоять только в одной группе (со стороны сущности «Группа»: кратность связи – «один», класс принадлежности - обязательный).* *В каждой группе может быть любое число студентов (со стороны сущности «Студент»: кратность связи – «много», класс принадлежности - необязательный).*
Оценка - Студент	*Элемент кватернарной связи «Студент получил оценку по дисциплине за выполнение контрольного мероприятия»* *Каждая оценка выставляется обязательно персонально (только одному) студенту (со стороны сущности «Студент»: кратность связи – «один», класс принадлежности - обязательный).* *Каждому студенту может быть выставлено любое количество оценок (со стороны сущности «Оценка»: кратность связи – «много», класс принадлежности - необязательный)*

Рис. 4.16. Глоссарий отношений связи между сущностями ER-модели (фрагмент)

Цель логического проектирования базы данных заключается в выборе способа отображения объектов предметной области через абстрактные объекты модели данных, чтобы это отображение не противоречило семантике предметной области и было, по возможности, лучшим (эффективным, удобным и т. д.).

Применительно к реляционной модели данных логическое проектирование баз данных представляет собой обоснованное принятие решения о том, из каких отношений должна состоять база данных, какие атрибуты должны быть у этих отношений и как данные отношения связаны между собой.

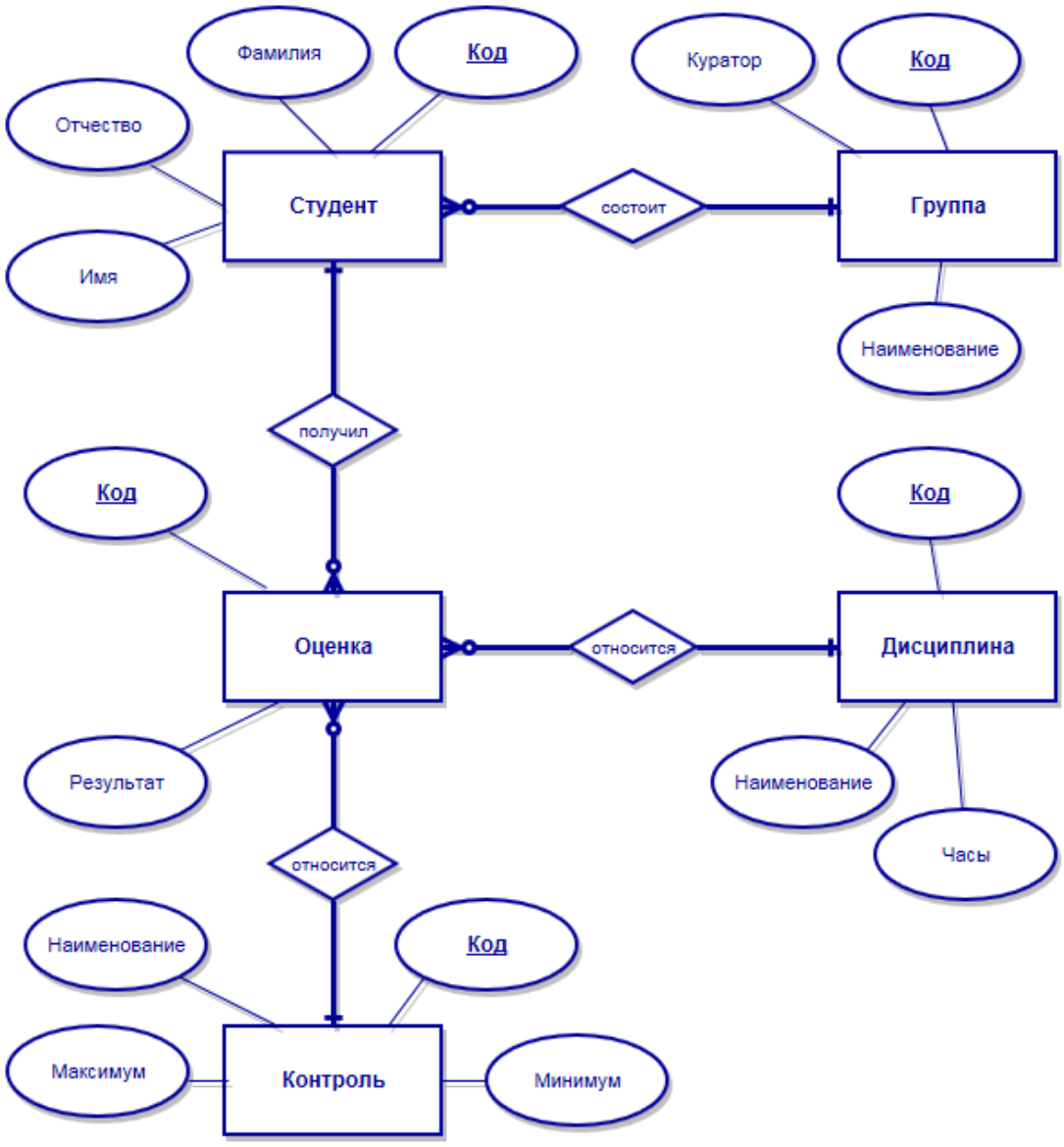


Рис. 4.17. Инфологическая модель (ER-диаграмма)

Основные методы проектирование реляционных баз данных:

1) с использованием инфологической модели «сущность-связь»;

2) с использованием универсального отношения.

Результатом выполнения этапа логического проектирования (вне зависимости от способа проектирования) являются схемы базы данных концептуального и внешнего уровней архитектуры.

Корректной считается такая схема базы данных, в которой отсутствуют избыточные функциональные зависимости. Избыточная функ-

циональная зависимость - зависимость, заключающая в себе такую информацию, которая может быть получена на основе других зависимостей, имеющихся в базе данных.

Если в базе данных присутствуют избыточные функциональные зависимости, то приходится прибегать к процедуре декомпозиции (разложения) имеющегося множества отношений. При этом порождаемое множество содержит большее число отношений, которые являются проекциями отношений исходного множества.

Обратимый пошаговый процесс замены данной совокупности отношений другой схемой с устранением избыточных функциональных зависимостей называется нормализацией.

Нормализация предназначена для приведения структуры базы данных к виду, обеспечивающему минимальную логическую избыточность, и не имеет целью уменьшение или увеличение производительности работы или же уменьшение или увеличение физического объёма базы данных. Конечной целью нормализации является уменьшение потенциальной противоречивости, хранимой в базе данных информации.

При нормализации совокупность отношений поочередно заменяется на совокупность отношений находящихся в первой, второй, третей, четвертой и пятой нормальных формах.

Большинство современных CASE-средств моделирования данных, как правило, поддерживает несколько графических нотаций построения информационных моделей. В частности, система ERwin фирмы Computer Associates поддерживает две нотации: IDEF1x и IE (англ. Information Engineering – информационное проектирование). Данные нотации являются взаимно-однозначными, т.е. переход от одной нотации к другой и обратно выполняется без потери качества модели. Отличие между ними заключается лишь в форме отображения элементов модели.

Отношения в диаграмме IDEF1X и IE описываются графическим объектом либо в виде прямоугольника с прямыми углами (для независимого отношения) либо в виде с округлыми углами (для зависимых отношений). Каждый прямоугольник, отображающий собой отношение, разделяется горизонтальной линией на две части: ключевую область (верхнюю часть) и область данных (нижнюю часть)

Для отражения ассоциаций (связей) между кортежами разных отношений в реляционной модели данных используется дублирование их

ключей. Атрибуты, представляющие собой копии ключей других отношений, называются внешними ключами (внешние ключи определяются как атрибуты первичных ключей родительского объекта, переданные дочернему объекту через их связь). Передаваемые атрибуты называются мигрирующими.

Связи и в IDEF1X и в IE отображаются в виде линии между двумя отношениями. Внешний вид связи указывает на ее мощность, тип и обязательность. Связи и в IDEF1X и в IE могут быть только либо унарными, либо бинарными (связи большей степени приводятся к бинарному виду).

Мощность и обязательность связи рассмотрены ранее. Различают следующие типы связи:

1) идентифицирующая (атрибуты одной сущности, называемые внешним ключом, входят в состав дочерней и служат для идентификации ее экземпляров, т.е. входят в ее первичный ключ)

2) неидентифицирующая (внешний ключ имеется в дочерней сущности, но не входит в состав первичного ключа);

В общем виде алгоритм логического проектирования схемы реляционной базы данных, в которой все отношения находятся в третьей нормальной форме, на основе инфологической модели «сущность-связь» выглядит следующим образом:

1) преобразование элементов инфологической модели в элементы даталогической модели по формальным правилам;

2) проверка модели с помощью правил нормализации;

3) определение требований поддержки целостности данных.

Целью первого этапа является преобразование элементов инфологической модели в элементы даталогической реляционной модели. Результатом данного этапа является схема отношений проектируемой реляционной базы данных, представленная в виде диаграммы «сущность-связь».

Преобразование инфологической модели в датологическую модель, как правило, осуществляется по формальным правилам:

Во-первых, каждая сущность преобразовывается в отношение. Имя сущности становится именем отношения.

Во-вторых, каждый простой атрибут становится атрибутом отношения с тем же именем. Если сущность имеет составной атрибут, то возможны два варианта:

1) составному свойству ставится в соответствие отдельный атрибут;

2) каждому из составляющих элементов составного свойства ставится в соответствие отдельный атрибут.

Выбор варианта зависит от характера обработки данных. При реализации запросов проще объединить поля, чем выделить часть поля. Если предполагается использование компонентов атрибута, лучше выбрать второй вариант, иначе – первый вариант.

Многозначные атрибуты удаляются. Многозначность устраняется путем введения новой сущности и связи «один-ко-многим». Каждому из многозначных атрибутов ставится в соответствие отношение, атрибутами которого будут идентификатор, выбранный в качестве первичного ключа, и многозначный атрибут. Ключ этого отношения будет составным, включающим оба эти атрибута.

Компоненты уникального идентификатора сущности преобразовываются в первичный ключ. Если в состав уникального идентификатора входят связи, то к числу атрибутов первичного ключа добавляется копия уникального идентификатора сущности, находящейся на дальнем конце связи (внешний ключ).

Во-третьих, связи с атрибутами удаляются (преобразовываются, в сущности). Избыточные связи удаляются (связь является избыточной, если одна и та же информация может быть получена не только через нее, но и с помощью другой связи).

Рекурсивную связь заменяют, определив дополнительную сущность и необходимое количество связей.

Бинарные, N-арные и иерархические связи организуются путем добавления новых и/или объединения существующих сущностей и добавления дополнительных атрибутов, соответствующим внешним ключам.

Организация типовых бинарных связей при преобразовании инфологической модели в датологическую приведена в табл. 4.1.

Таблица 4.1 – Организация типовых бинарных связей при преобразовании инфологической модели в датологическую

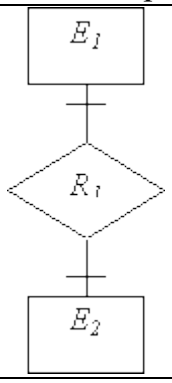
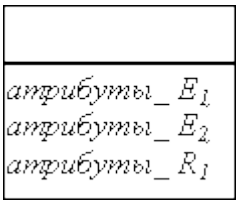
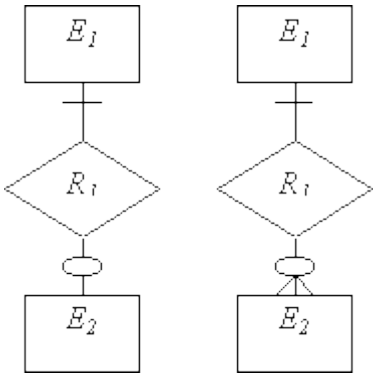
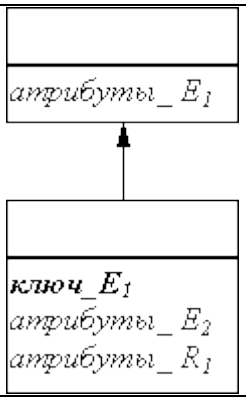
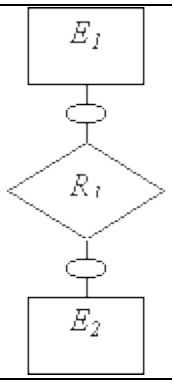
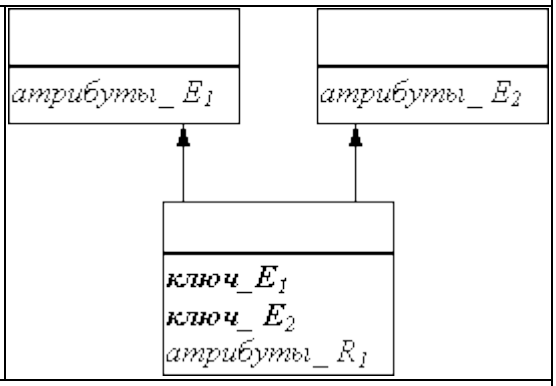
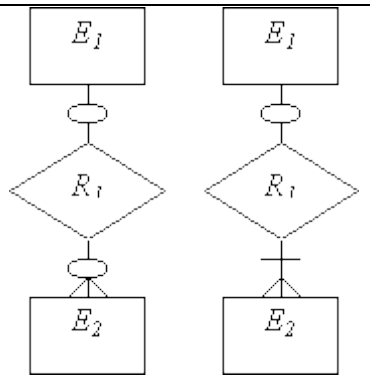
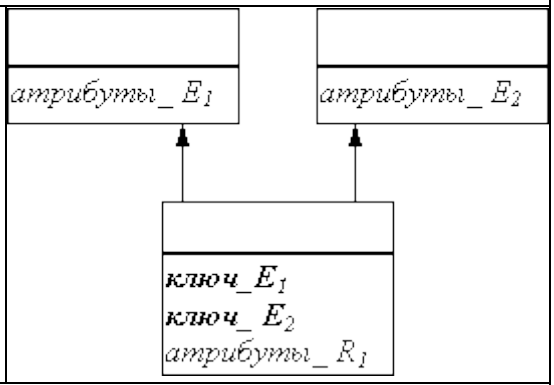
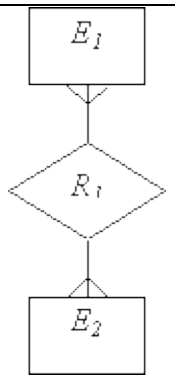
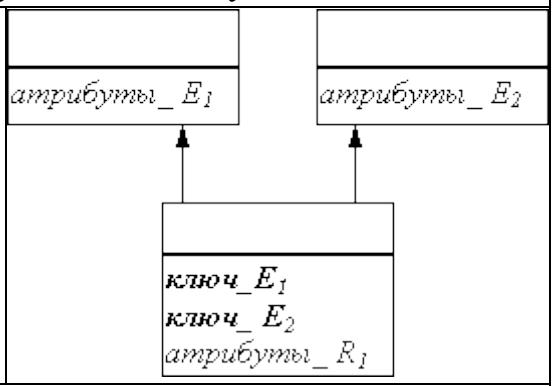
<table>
<tr><td rowspan="2">Тип связи</td><td>Пример связи</td><td>Отношения</td></tr>
<tr><td colspan="2">Правило построения отношений</td></tr>
<tr><td rowspan="2">Тип 1
(1,1):(1,1)</td><td> 
</td><td> 
</td></tr>
<tr><td colspan="2">Требуется только одно отношение. Первичным ключом данного отношения может быть ключ любой из сущностей.</td></tr>
<tr><td rowspan="2">Тип 2
(1,1):(0,1)
(1,1):(0,n)</td><td> 
</td><td> 
</td></tr>
<tr><td colspan="2">Для каждой сущности создается свое отношение, при этом ключи сущностей служат ключами соответствующих отношений. Кроме того, ключ сущности с обязательным классом принадлежности добавляется в качестве внешнего ключа в отношение, созданное для сущности с необязательным классом принадлежности</td></tr>
<tr><td rowspan="2">Тип 3
(0,1):(0,1)</td><td> 
</td><td> 
</td></tr>
<tr><td colspan="2">Необходимо использовать три отношения: по одному для каждой сущности (ключи сущностей служат первичными ключами отношений) и одно отношение для связи. Отношение, выделенное для связи, имеет два атрибута - внешних ключа – по одному от каждой сущности.</td></tr>
</table>

Таблица 4.1 – Организация типовых бинарных связей при преобразовании инфологической модели в датологическую (продолжение)

Тип связи	Пример связи	Тип связи
	Правило построения отношений	
Тип 4 (0,1):(0,n) (0,1):(1,n)		
	Формируются три отношения: по одному для каждой сущности, причем ключ каждой сущности служит первичным ключом соответствующего отношения, и одно отношение для связи. Отношение, выделенное для связи, имеет два атрибута - внешних ключа - по одному от каждой сущности.	
Тип 5 n : m		
	В этом случае всегда используются три отношения: по одному для каждой сущности, причем ключ каждой сущности служит первичным ключом соответствующего отношения, и одно отношение для связи. Последнее отношение должно иметь среди своих атрибутов внешние ключи, по одному от каждой сущности.	

Связь «один-к-одному» между сущностями встречается редко. Если класс принадлежности обеих сущностей является обязательным, то для отображения обеих связанных сущностей можно использовать одно отношение (тип 1).

Однако таким решением злоупотреблять не следует. Если для каждого объекта потребуются свои связи или в запросах потребуется информация по каждой сущности, то выбранное решение усложнит или замедлит работу с базой данных.

Если для каждой сущности создаются отдельные отношения, то информацию о связях можно отразить, включив в одно из отношений идентификатор из другого отношения (причем это можно сделать в любом из отношений)

Если класс принадлежности одной из сущностей является необязательным, то идентификатор сущности с необязательным классом добавляется в отношение, соответствующее сущности с обязательным классом принадлежности (тип 2).

Если класс принадлежности обеих сущностей является необязательным, то, чтобы избежать наличия пустых атрибутов, следует использовать три отношения: по одному для каждой сущности и одно – для отображения связи между ними (тип 3).

Преобразование бинарной связи «один-ко-многим» (1:N) зависит только от класса принадлежности N-связной сущности. Если он является обязательным, то можно использовать два отношения (по одному для каждой сущности). В отношение для N-связной сущности добавляется идентификатор 1-связной сущности (тип 2).

Если класс принадлежности N-связной сущности является необязательным, то для отображения связи создается третье отношение, которое будет содержать ключи каждой из связанных сущностей (тип 3).

Для бинарной связи многие-ко-многим (M:N) потребуются три отношения: по одному для каждой сущности и одно дополнительное – для отображения связи между ними. Последнее отношение будет содержать идентификаторы связанных объектов. Ключ этого отношения будет составным (тип 4).

В случае N-арной связи необходимо использовать (n+1) отношение – по одному для каждой сущности, и одно для связи. Идентификатор каждой сущности станет первичным ключом соответствующего отношения. Отношение, порождаемое связью, будет иметь среди своих атрибутов ключи каждой сущности. Если связь имеет атрибуты, то они становятся атрибутами отношения связи.

В случае иерархической связи необходимо возможны два варианта построения реляционных отношений.

Согласно первому для иерархической структуры создается одно отношение, которое содержит атрибуты связи и всех сущностей. Недостаток такого способа - для каждого кортежа часть атрибутов всегда будет неопределенна.

По второму способу генерируется по одному отношению для каждой дочерней сущности. Каждое из этих отношений включает атрибуты родительской сущности и связи кроме атрибутов – дискриминантов. Недостатком данного способа является невозможность получить в одном запросе избыточное количество кортежей.

Результатом этапа проверки полученной модели с помощью правил нормализации является нормализованная схема отношений проектируемой реляционной базы данных, представленная в виде диаграммы «сущность-связь».

Построенные на первом этапе реляционные отношения, не являются окончательной схемой базы данных. Их необходимо проверить на избыточные функциональные зависимости и привести к BCNF (нормальной формы Бойса-Кодда) или нормальной форме более высокого порядка.

Целью этапа определения требований поддержки целостности данных является выявление и определение основных требований к поддержке целостности данных. Результатом данного этапа являются:

1) перечень требований обеспечения целостности данных;

2) перечень требований обеспечения ссылочной целостности.

Ограничения целостности данных представляют собой ограничения, которые вводятся с целью предотвращения помещения в базу противоречивых данных.

К этим ограничениям относятся:

1) обязательные данные – атрибуты, которые всегда должны содержать одно из допустимых значений (NOT NULL);

2) домены – наборы допустимых значений для атрибута (тип данных, допустимые значения, значение по умолчанию);

3) бизнес-правила (бизнес-ограничения) – ограничения, принятые в рассматриваемой предметной области;

4) ссылочная целостность – набор ограничений, обеспечивающих соответствие каждого внешнего ключа одного отношения первичному ключу другого отношения.

Рассмотрим в качестве примера даталогическое проектирование базы данных мобильного приложения для оценки успеваемости студентов, концептуальная модель которого была рассмотрена ранее.

В процессе разработки даталогической модели было проведено преобразование элементов инфологической модели в элементы даталогической модели по формальным правилам. Список преобразований элементов инфологической модели в элементы даталогической модели приведен на рис. 4.18.

Признак		*Результат преобразования*
Наименование	*Наличие*	
Наличие сущностей	*Есть*	*Каждая сущность **«Группа»**, **«Студент»**, **«Дисциплина»**, **«Контроль»**, **«Оценка»** была преобразована в отношение с аналогичным именем*
Наличие связи с атрибутами	*Нет*	-
Наличие простых атрибутов у сущностей	*Есть*	*Каждый простой атрибут всех сущностей был преобразован в атрибут отношения с аналогичным именем*
Наличие составных атрибутов у сущностей	*Нет*	-
Наличие многозначных атрибутов у сущностей	*Нет*	-
Наличие уникальных идентификаторов	*Есть*	*Компоненты уникальных идентификаторов сущности были преобразованы в первичные ключи отношений*
Наличие в составе уникальных идентификаторов связей	*Нет*	-
Наличие избыточных связей	*Нет*	-
Наличие рекурсивных связей	*Нет*	--
Наличие бинарных связей типа 1	*Нет*	-
Наличие бинарных связей типа 2	*Есть*	*Бинарные связи **«Студент – Группа»**, **«Оценка – Студент»**, **«Оценка – Дисциплина»**, **«Оценка – Контроль»** преобразованы по типу 2: для каждой сущности создается свое отношение, при этом ключи сущностей служат ключами соответствующих отношений. Кроме того, ключ сущности с обязательным классом принадлежности добавляется в качестве внешнего ключа в отношение, созданное для сущности с необязательным классом принадлежности*
Наличие бинарных связей типа 3	*Нет*	-
Наличие бинарных связей типа 4	*Нет*	-
Наличие бинарных связей типа 5	*Нет*	-
Наличие n-арных связей	*Нет*	-
Наличие иерархических связей типа 5	*Нет*	-

Рис. 4.18. Список преобразований элементов инфологической модели в элементы даталогической модели

Результат преобразования элементов инфологической модели в элементы даталогической модели по формальным правилам приведен на рис. 4.19.

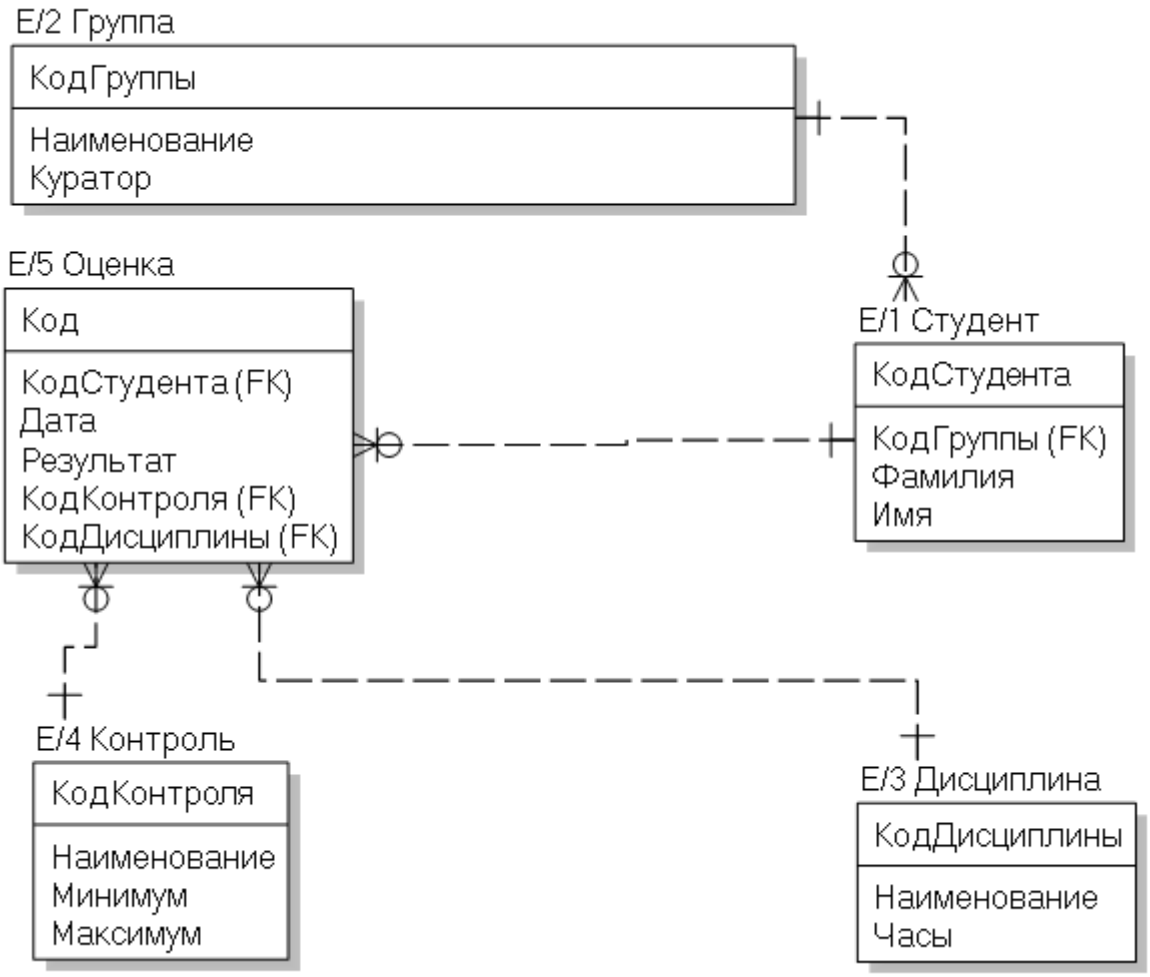


Рис. 4.19. Даталогическая диаграмма «Сущность-связь» в нотации IE (до нормализации)

В процессе разработки даталогической модели была проведена проверка полученной схемы отношений реляционной базы данных на наличие избыточных функциональных зависимостей, а также приведено её приведение к нормальной форме BCNF (Бойса-Кодда).

Результат проверки даталогической модели с помощью правил нормализации:

1) все отношения находятся в 1НФ, так как в любом допустимом значении всех отношений каждый его кортеж содержит только одно значение для каждого из атрибутов (приведение не требуется);

2) все отношения находятся в 2 НФ, так как они находятся в 1NF и каждый их неключевой атрибут функционально полно зависит от её потенциального ключа (приведение не требуется);

3) все отношения находятся в 3 НФ, так как они находятся в 2NF и отсутствуют транзитивные функциональные зависимости неключевых атрибутов от ключевых (приведение не требуется);

4) все отношения находятся в BCNF, так как они находятся в 3NF и отсутствуют зависимости атрибутов первичного ключа от неключевых атрибутов (приведение не требуется).

Окончательная схема отношений реляционной базы данных приведена на рис. 4.20.

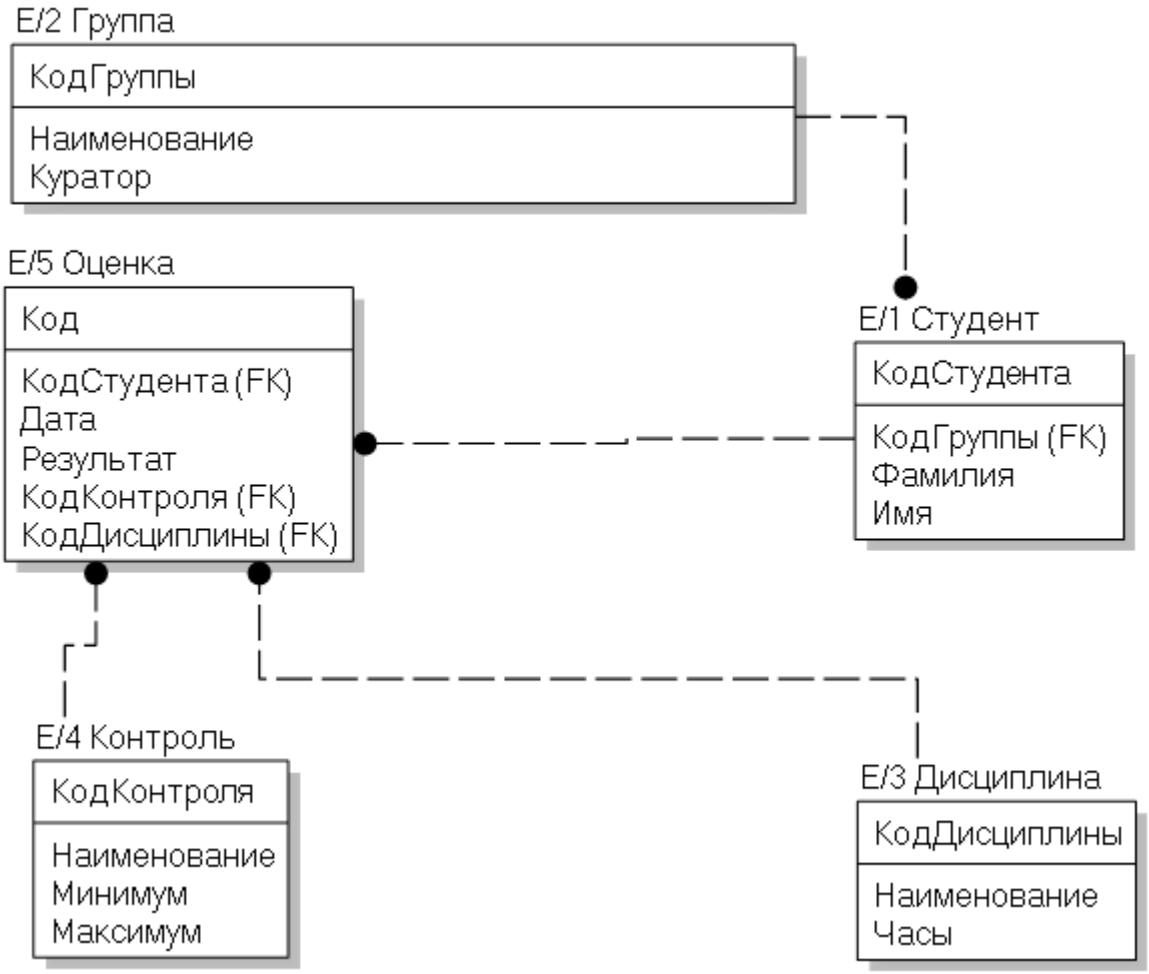


Рис. 4.20. Даталогическая диаграмма «Сущность-связь» в нотации IDEF1x (после нормализации)

В процессе разработки даталогической модели было проведено определение требований поддержки целостности данных. Фрагмент результатов определения требований поддержки целостности данных приведены на рис. 4.21. Требования к обеспечению ссылочной целостности данных – на рис. 4.22.

Сущность	*Наименование атрибута (синоним)*	*Описание требований поддержки целостности данных (обязательность данных, тип данных, допустимые значения, значение по умолчанию, бизнес-правила (бизнес-ограничения), ссылочная целостность)*
Группа	*КодГруппы*	*Данные являются обязательными, имеют числовой тип (целое число), интервал допустимых значений: больше 0; бизнес-правила не установлены, значение по умолчанию: уникальное.*
	Наименование	*Данные являются обязательными, имеют строковой тип (длина – 20 символов), интервал допустимых значений: не пустое значение; бизнес-правила не установлены, значение по умолчанию: «Наименование группы».*
	Куратор	*Данные являются необязательными, имеют строковой тип (длина – 30 символов), интервал допустимых значений: любой; бизнес-правила не установлены, значение по умолчанию: нет.*
Студент	*КодСтудента*	*Данные являются обязательными, имеют числовой тип (целое число), интервал допустимых значений: больше 0; бизнес-правила не установлены, значение по умолчанию: уникальное.*
	Фамилия	*Данные являются обязательными, имеют строковой тип (длина – 50 символов), интервал допустимых значений: не пустое значение; бизнес-правила не установлены, значение по умолчанию: «Фамилия студента».*
	Имя	*Данные являются обязательными, имеют строковой тип (длина – 20 символов), интервал допустимых значений: не пустое значение; бизнес-правила не установлены, значение по умолчанию: «Имя студента»*
	Отчество	*Данные являются необязательными, имеют строковой тип (длина – 20 символов), интервал допустимых значений: любой; бизнес-правила не установлены, значение по умолчанию: нет.*
	КодГруппы	*Данные являются обязательными, имеют числовой тип (целое число), интервал допустимых значений: больше 0; бизнес-правила не установлены, значение по умолчанию: 1.*

Рис. 4.21. Требования поддержки целостности данных (фрагмент)

Отношение		*Требования к обеспечению ссылочной целостности данных*
Родительское	*Дочернее*	
Группа	*Студент*	*При вставке кортежа в дочернее отношение во внешний ключ обязательно должно быть внесено значение, соответствующее одному из значений первичного ключа родительского отношения* *При вставке кортежа в дочернее отношение во внешний ключ заносится значение по умолчанию. Значение по умолчанию должно соответствовать одному из значений первичного ключа родительского отношения (т.е. в родительском отношении должен присутствовать кортеж, значение первичного ключа которого соответствует значению по умолчанию)* *Изменение значения первичного ключа в родительском отношении возможно лишь в том случае, если это значение не соответствует ни одному из значений внешнего ключа в дочернем отношении* *При изменении значения внешнего ключа в дочернем отношении, оно обязательно должно соответствовать одному из значений первичного ключа родительского отношения* *При изменении значения первичного ключа в родительском отношении во внешние ключи всех связанных кортежей дочернего отношения заносится его новое значение* *При изменении значения первичного ключа в*
Студент	*Оценка*	
Дисциплина	*Оценка*	
Контроль	*Оценка*	

Рис. 4.22. Требования к обеспечению ссылочной целостности данных (фрагмент)

Результатом разработки даталогической модели является глоссарий атрибутов и ER-диаграмма.

Фрагмент глоссарий атрибутов приведен на рис. 4.22, а ER- диаграмма – на рис. 4.23.

Под физической понимается модель, отражающая описание данных средствами конкретной СУБД.

Цель физического проектирования заключается в преобразовании логической схемы проектируемой базы данных в схемы базы данных для конкретной СУБД с учетом её специфики (синтаксиса, семантики и возможностей) с обеспечением определенного уровня производительности.

Специфика конкретной системы управления базами данных определяется:

Атрибут	*Домен*	*Тип*	*Описание атрибута*
Сущность: E/3 Дисциплина			
КодДисцип-лины	*INTEGER*	*Not Null*	*Уникальный идентификатор дисциплины (данные являются обязательными, имеют числовой тип (целое число), интервал допустимых значений: больше 0; бизнес-правила не установлены, значение по умолчанию: уникальное)*
Фамилия	*VARCHAR (50)*	*Not Null*	*Фамилия студента (данные являются обязательными, имеют строковой тип (длина – 50 символов), интервал допустимых значений: не пустое значение; бизнес-правила не установлены, значение по умолчанию: «Фамилия студента»)*
Часы	*INTEGER*	*Null*	*Количество часов дисциплины по учебному плану (данные являются необязательными, имеют числовой тип (целое число), интервал допустимых значений: больше 0; бизнес-правила: не более 180, значение по умолчанию: 72)*

Рис. 4.23. Требования к обеспечению ссылочной целостности данных (фрагмент)

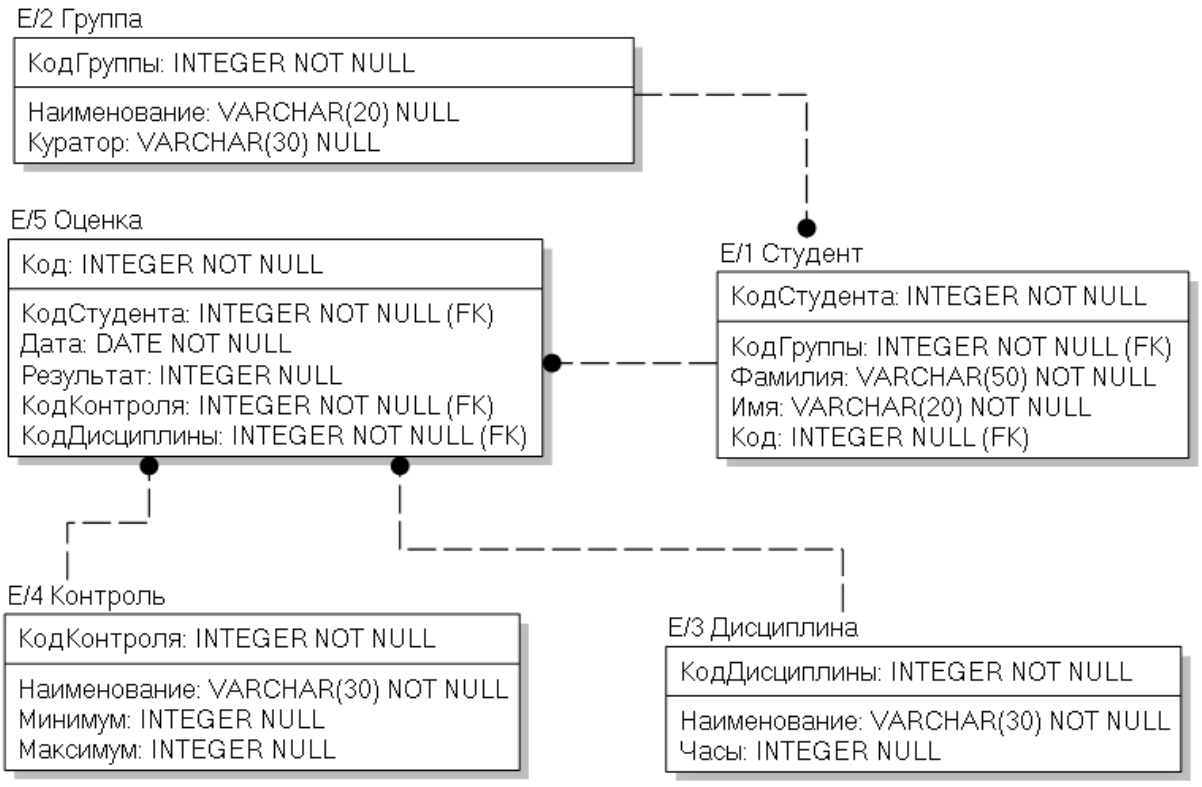


Рис. 4.24. Требования к обеспечению ссылочной целостности данных (фрагмент)

1) спектром возможностей и ограничений при создании и использовании основных компонентов реляционных СУБД, которыми являются: таблицы, представления, индексы, умолчания, правила ограничения целостности, пользователи, роли, хранимые процедуры и триггеры;

2) поддерживаемых методов управления дисковой памятью, разделения баз данных по файлам и устройствам, доступа к данным и т.д.

Таблицы базы данных предназначены для хранения данных, являются реализацией отношения (соответственно, поля таблицы являются реализацией атрибутов, записи таблицы – кортежей). Различают пользовательские таблицы (содержат данные из предметной области) и системные таблицы (содержат служебную информацию).

С таблицами тесно связаны и не существуют вне их:

1) индексы (предназначены для упорядоченности табличных данных по различным критериям, обеспечивая быстрый доступ к записям и повышая производительность операций по поиску и выборке данных);

2) значения по умолчанию (определяют предустановленные значения полей);

3) ограничение целостности (определяют ограничения на возможные значения полей или их типа);

4) пользователи и роли пользователей (определяют список пользователей, которым разрешены операции с компонентами базы данных и то к какой группе они относятся);

5) триггеры (специальные виды хранимых процедур, предназначенные для производства некоторых специальных действий при выполнении операций вставки, удаления и редактирования данных).

Представления базы данных представляют собой подмножество записей из одной или нескольких таблиц и предназначены для осуществления быстрой выборки данных, заданной структуры.

Хранимые процедуры представляют собой наборы команд языка структурированных запросов и предназначены для выполнения по команде пользователей (при и наличии у них прав).

В общем виде алгоритм физического проектирования (реализации даталогической схемы проектируемой базы данных на конкретной СУБД) выглядит следующим образом:

1) выбор конкретной системы управления базы данных для построения базы данных;

2) введение контролируемой избыточности;

3) перенос логической схемы данных в среду целевой СУБД;

4) реализация бизнес-правил и требований ссылочной целостности;

5) определение индексов;

6) разработка механизмов защиты;

7) организация мониторинга функционирования базы данных и ее настройка

Выбор целевой СУБД представляет собой сложную задачу, в рамках которой альтернативные варианты оцениваются по различным параметрам, в т.ч. быстродействия, стоимости, затрат ресурсов мобильного устройства, безопасности и т.д.

Проектирование физической организации базы данных предполагает определение возможных способов файловой организации для таблиц и выбор наилучшего из них. Оценка качества физической организации базы данных осуществляется на основе компромиссного значения следующих показателей:

1) производительности выполнения транзакций (этот показатель представляет собой количество транзакций, которые могут быть обработаны за заданный интервал времени);

2) времени ответа (характеризует временной промежуток, необходимый для выполнения одной транзакции);

3) размера используемой оперативной и постоянной памяти мобильного устройства, а также других его ресурсов.

На основании указанных показателей принимаются решения об оптимизации производительности базы данных путем снижения требований к уровню нормализации таблиц или определения индексов в таблицах, ускоряющих выборку данных из базы.

Намеренное приведение схемы отношений базы данных в состояние, не соответствующее критериям нормализации, обычно проводимое с целью ускорения операций чтения из базы за счет добавления избыточных данных, называется денормализацией.

Снижение требований к уровню нормализации отношений осуществляется с учетом следующих соображений: абсолютно правильно

спроектированная реляционная схема базы данных мешает эффективному выполнению операций в конкретной прикладной области при использовании конкретного сервера баз данных (обычно это связано с особенностями синхронизации параллельно выполняемых транзакций). Поэтому иногда приходится жертвовать нормализованной и работать с ненормализованной схемой реляционной базы данных.

Денормализацию следует рассматривать как расширение нормализованной модели данных, которое повышает производительность операций пользователей.

При запросе пользователями данных из нескольких таблиц (такие запросы реализуются через операцию реляционной алгебры – соединение таблиц) производительность работы с базой данных уменьшается. Соответственно сокращение количества таблиц, участвующих в таких запросах, позволяет повысить эффективность работы с базой данных. Устранение операции соединения таблиц достигается за счет перенос полей из одной таблицы в другую таблицу, объединения нескольких таблиц в одну.

При работе с базой данных в некоторых ситуациях приходится определять значения одних полей с использованием значений других полей (такие поля называются расчетными или производными). При физическом проектировании необходимо определить, должно ли производное поле храниться в базе данных или вычисляться каждый раз, когда в нем возникает необходимость. Выбор проектного решения осуществляется с учетом:

1) размера дополнительных затрат на хранение производных данных и поддержание их согласованности с реальными данными, на основе которых они вычисляются;

2) размера затрат на вычисление производных данных, если их вычисление выполняется по мере необходимости.

Из этих двух вариантов выбирается наименее дорогостоящий с учетом требований к производительности.

Некоторые СУБД имеют определенные ограничения на количество полей в таблице в целом, а также количество полей таблицы определенного типа. В этом случае приходится разбивать исходную таблицу на несколько связанных между собой таблиц. Также разделение таблиц имеет смысл использовать, если:

1) одни поля активней используется в запросах пользователей по сравнению с другими;

2) одни записи (определенной группы) таблиц активней добавляются, редактируются и удаляются (например, актуальные и архивные записи).

При наличии таблиц, которые гарантированно состоят из фиксированного числа записей (месяца, дни недели и т.д.), при физическом проектировании прибегают к объединению таких таблиц с другими (формируя вместо таблиц поля сформированных таблиц со списком возможных значений).

Основные способы введения контролируемой избыточности:

1) нисходящая денормализация (предполагает перенос поля из родительской таблицы в дочернюю, что устраняет операции соединения таблиц);

2) восходящая денормализация (предполагает перенос поля из дочерней таблицы в родительскую, что операции соединения таблиц);

3) внутритабличная денормализация (предполагает введение избыточных полей в одной таблице, что устраняет операции расчета значений производных полей);

4) денормализация методом разбиения (предполагает разбиение одной таблицы на несколько таблиц, что устраняет дополнительные операции ввода/вывода и снимает наложенные со стороны СУБД ограничения);

5) денормализация методом слияния (предполагает объединение нескольких таблиц, что устраняет соединения таблиц или уменьшает число операций вставки).

На следующем этапе осуществляется перенос логической схемы базы данных в среду целевой СУБД с учетом её возможностей. Также на данном этапе осуществляется модификация логической схемы с учетом семантики и синтаксиса, принятой в целевой СУБД, а именно, соблюдение правил наименования таблиц, полей, типов данных, а также корректировку типа данных, если целевая СУБД не поддерживает требуемые типы данных или не имеет возможность адекватного хранения данных данного типа.

Проектирование таблиц базы данных и связей между ними средствами выбранной СУБД, в частности предполагает определение:

1) имен таблиц;

2) списка полей (в т.ч. их имен, типов данных, принимаемых значений по умолчанию, диапазона возможных значений, допустимости значения NULL);

3) определение первичных ключей и (если таковые существуют) внешних ключей;

4) списка производных полей и описание способов их вычисления.

Бизнес-правило – это правило, позволяющее обеспечить качество, точность данных и ограничения ни них со стороны предметной области при обновлении информации. Реализация бизнес-правил в среде выбранной системы управления базами данных совместно с обеспечением требований ссылочной целостности является следующим этапом разработки базы данных. Для этих целей используются триггеры и хранимые процедуры. Часть бизнес-правил реализуется в коде мобильного приложения.

Определение индексов осуществляется с учетом следующих соображений: чем больше индексов существует над таблицами базы данных, тем более вероятным будет быстрое выполнение запросов по выборке данных и тем медленнее будут выполняться операции модификации базы данных и наоборот.

Поэтому требуется тщательный предварительный анализ наиболее важных запросов, для которых использование индексов является критически важным, а во всех остальных случаях использование индексов постараться исключить.

Оценка качества потенциального индекса осуществляется на основании следующих показателей:

1) показателя кардинальности столбца таблицы (определяется как число различных уникальных значений в столбце таблицы);

2) показатель полезности индекса (определяется как отношение кардинальности столбца к количеству записей в таблице).

Чем выше значения показателя кардинальности столбца таблицы, тем больше будет число записей в индексе. Чем выше показатель полезности индекса, тем меньше требуется операций для получения выборки строк из таблицы.

При разработке базы данных мобильного приложения необходимо оценить с помощью приведенных показателей все потенциаль-

ные индексы и выбрать только те, которые имеют максимальный уровень полезности. Такой подход обеспечивает наиболее быстрый поиск данных. Отметим, что уникальные индексы (например, обслуживающие первичные ключи) обладают максимальным индексом полезности.

В целом, индексы рекомендуется создавать в случае, когда по столбцу или группе столбцов:

1) часто производится поиск при работе с базой данных;

2) часто строятся объединения таблиц;

3) часто производится сортировка.

Разработка стратегии защиты базы данных предполагает выбор способа предоставления санкционированного раздельного доступа к базе данных как ценному ресурсу. Это в большей степени касается разработки серверных решений. Основным способом защиты локальных баз является их шифрование.

Для реализации всех возможностей целевой реляционной СУБД используется формальный непроцедурный язык программирования SQL (англ. structured query language – «язык структурированных запросов»).

Язык SQL представляет собой совокупность инструкций, каждая из которых содержит:

1) описание действия в форме оператора;

2) описание данных в форме одного или нескольких предложений, уточняющих информацию о действии.

Язык SQL стандартизован [20], однако следует учитывать, что реализация инструкций SQL на конкретной СУБД может его не придерживаться.

Операторы SQL по выполняемым задачам классифицируют на:

1) операторы определения данных;

2) операторы манипуляции данными;

3) операторы определения доступа к данным;

4) операторы управления транзакциями.

Операторы языка определения данных используется для управления (создания, модификации и удаления) объектами базы данных.

В стандарте SQL различают следующие виды объектов:

1) база данных (DATABASE);

2) таблица (TABLE);

3) столбец (COLUMN);

4) индекс (INDEX);

5) представление (VIEW);

6) хранимая процедура.

Для каждой конкретной системы управления базы данных существует свой набор объектов базы данных, который может значительно расширять набор объектов, предусмотренный стандартом.

Язык SQL содержит следующие конструкции определения данных:

1) CREATE;

2) ALTER;

3) DROP.

Оператор «CREATE» используется для создания объекта базы данных, оператор «ALTER» используется для изменения существующего объекта базы данных или самой базы данных, оператор «DROP» используется для удаления существующего объекта базы данных или самой базы данных.

Операторы манипуляции данными используется для манипулирования (добавления, изменения, удаления и извлечения) данных внутри объектов реляционной базы данных.

Язык SQL содержит следующие конструкции:

1) INSERT;

2) UPDATE;

3) DELETE;

4) SELECT.

Оператор «INSERT» используется для добавления строк в таблицу, оператор «UPDATE» используется для изменения значений полей в таблице, оператор «DELETE» удаление строк из таблицы, оператор «SELECT» используется извлечение данных из одной или нескольких таблиц, а также для предварительного соединения двух и более таблиц.

Операторы определения доступа к данным предназначены для осуществления административных операций, присваивающих или отменяющих право (привилегию) использовать базу данных, таблицы и другие объекты базы данных, а также выполнять те или иные операторы SQL.

Язык SQL содержит следующие конструкции:

1) GRANT;

2) REVOKE.

Оператор «GRANT» используется для присвоения привилегии, оператор «REVOKE» используется для отмены привилегии.

Операторы управления транзакциями предназначены для обработки транзакций (группы последовательных операций с базой данных, которая представляет собой логическую единицу работы с данными).

Одним из наиболее распространённых наборов требований к транзакциям является: атомарность, согласованность, изолированность и устойчивость.

Язык SQL содержит следующие конструкции:

1) COMMIT,

2) ROLLBACK,

3) SAVEPOINT.

Оператор «COMMIT» используется для применения транзакции, оператор «ROLLBACK» используется для отмены («отката») всех изменений, сделанных в контексте текущей транзакции, «SAVEPOINT» используется для деления транзакции на более мелкие участки.

Рассмотрим в качестве примера физическое проектирование базы данных мобильного приложения для оценки успеваемости студентов, концептуальная и даталогическая модель которого была рассмотрена ранее.

В качестве системы управления базами данных выбрана SQLite. SQLite – реляционная система управления базами данных, которая не использует клиент-серверную архитектуру, являясь программной библиотекой, которая может быть встроена в код мобильного приложения.

Для хранения описания структуры базы данных и всех хранящихся там данных используется один файл. Одновременно к файлу базы данных может выполнятся несколько запросов на чтение, но запись в файл возможна только в том случае, если СУБД не обслуживает каких-либо других запросов, что не является недостатком для пользователей мобильных устройств так как, они являются единственными кто с этой базой данных работает.

В процессе разработки физической модели была проведена проверка схемы отношений реляционной базы данных на необходимость

проведения денормализации, а также приведено преобразование. Результат проверки и требуемые приведения приведены на рис. 4.25.

Способ контролируемой избыточности	*Результат анализа.* *Требуемые приведения.*
Нисходящая денормализация	*Нет необходимости переноса полей из родительской таблицы в дочернюю таблицу.* *Преобразование не требуется.*
Восходящая денормализация	*Нет необходимости переноса полей из дочерней таблицы в родительскую таблицу.* *Преобразование не требуется.*
Внутритабличная денормализация	*Нет необходимости добавления полей в таблицы (нет производных данных).* *Преобразование не требуется.*
Денормализация методом разбиения	*Нет необходимости разбиения одной таблицы на несколько таблиц* *Преобразование не требуется.*
Денормализация методом слияния	*Нет необходимости объединения нескольких таблиц* *Преобразование не требуется.*

Рис. 4.25. Проверка схемы отношений реляционной базы данных на необходимость проведения денормализации

В процессе разработки физической модели был осуществлен перенос логической схемы базы данных в среду целевой СУБД SQLite.

При переносе выполнялись требования к наименованию объектов базы данных, а также к количественным характеристикам таблиц, их полей, а также типам данных [21].

Реализация требований ссылочной целостности обеспечена внутренними средствами СУБД.

В процессе разработки физической модели было предложены индексы для увеличения производительности работы с базой данной и проведена оценка эффективности их использования. Фрагмент результатов описания и оценки эффективности использования индексов приведены на рис. 4.26.

СУБД SQLite не предусматривает защиту на уровне пользователя. Средство шифрования в SQLite сочетает в себе два средства – кодирование и пароли баз данных. При использовании пароля для шифрования базы данных все данные становятся нечитаемыми в других программах. Для того чтобы использовать такую базу данных,

пользователи должны вводить пароль. SQLite реализует также поддержку программ шифрования сторонних поставщиков.

Сущность (предполагаемое количество записей)	*Наименование индекса*	*Описание индекса (состав, обслуживание ключа, уникальность, порядок сортировки, кардинальность, полезность индекса). Решение о применении индекса*
Группа (100)	*ИндексКодГруппы*	*(Состав: «КодГруппы»; ключевой, уникальный, по возрастанию, 100, **1**) Принято решение: использовать*
	ИндексНаименование	*(Состав: «Наименование»; уникальный, по возрастанию, 100, **1**) Принято решение: использовать*
Студент (2500)	*ИндексКодСтудента*	*(Состав: «КодСтудента»; ключевой, уникальный, по возрастанию, 2500, **1**) Принято решение: использовать*
	ИндексФИО	*(Состав: «Фамилия», «Имя», «Отчества»; не уникальный, по возрастанию, 2490, **0,99**) Принято решение: использовать*
	ИндексКодГруппы (в одной группе в среднем обучается 25 студентов)	*(Состав: «КодГруппы»; ключевой, не уникальный, по возрастанию, 2500, **0,04**) Принято решение: использовать*
Контроль (10)	*ИндексКодКонтроля*	*Состав: «КодКонтроля»; ключевой, уникальный, по возрастанию, 10, **1**) Принято решение: использовать*
	ИндексНаименование	*(Состав: «Наименование»; уникальный, по возрастанию, 10, **1**) Принято решение: использовать*
Дисциплина (30)	*ИндексКодДисциплины*	*Состав: «КодДисциплины»; ключевой*

Рис. 4.26. Описание и оценка эффективности использования индексов (фрагмент)

Таким образом, в процессе разработки физической модели было предложена защита базы данных с помощью шифрования. Установлен пароль по умолчанию «7777777».

В процессе разработки физической модели были сформированы инструкции определения данных на языке SQL, связанные с созданием объектов базы данных.

Для работы с базами данных SQLite создано множество инструментов, которые работают на различных платформах. Описание инструкций по созданию новых объектов проектируемой базы данных в приложении SQLiteStudio приведено на рис. 2.27 и 2.28.

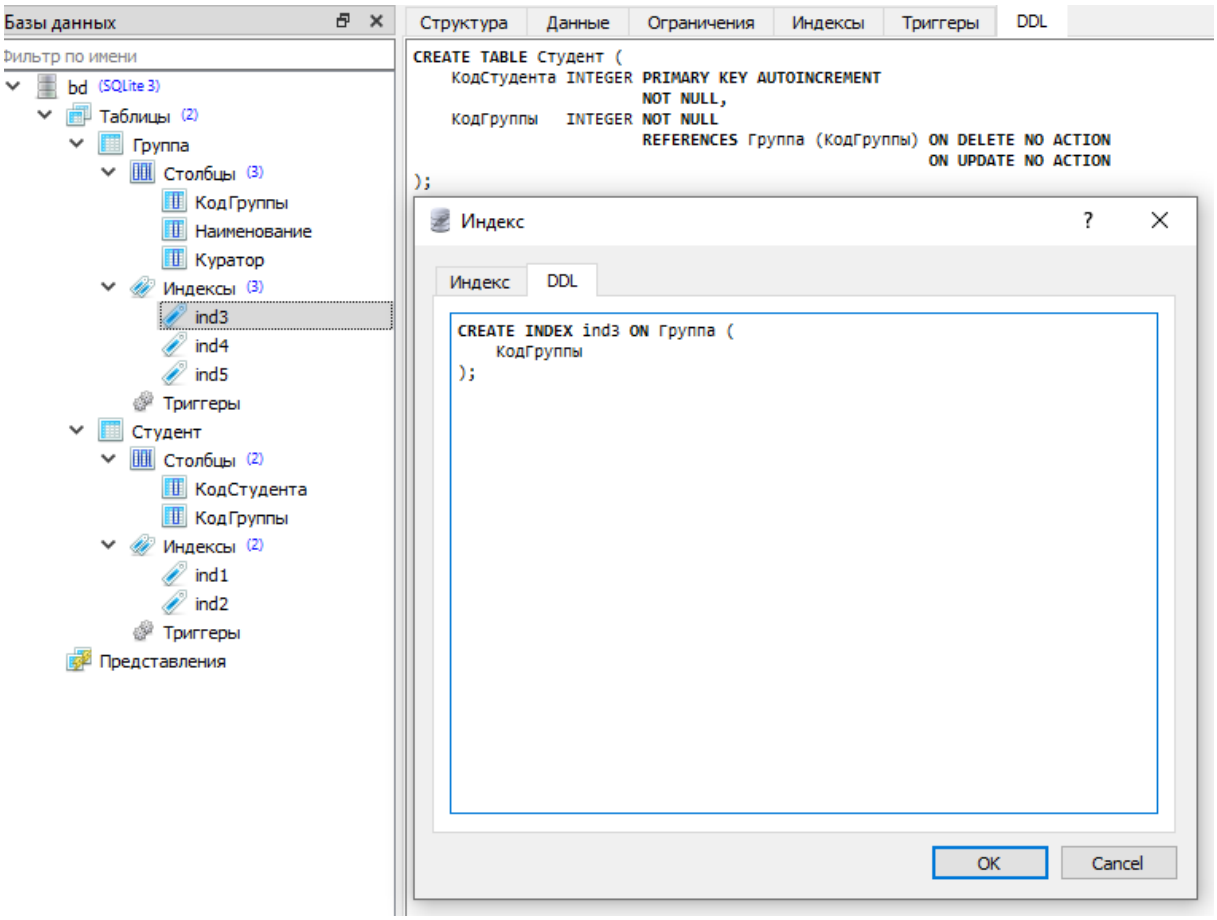


Рис. 4.28. Описание инструкций по созданию новых объектов проектируемой базы данных (фрагмент генерации SQL-запроса на создание таблицы «Группа» и индекса для его поля «КодГруппы»)

SQLiteStudio является кроссплатформенным CASE-средством, которое обладает следующими возможностями:

1) открытый исходный код;

2) поддержка шифрования файла базы данных;

3) поддержка выполнения пользовательских функций SQL на таких языках как: JavaScript, Python или Tcl;

4) одновременная работа с несколькими базами данных;

5) хранение истории SQL-запросов;

6) возможность импорта данных в базу данных из различных форматов.

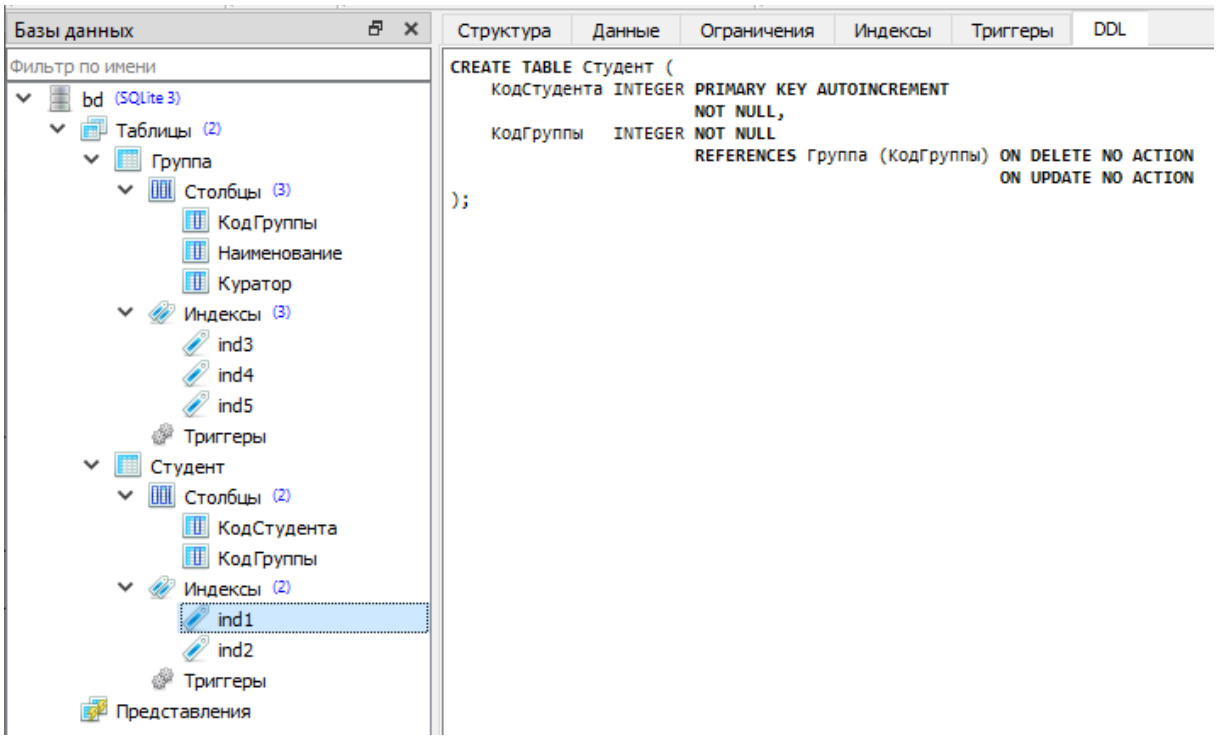


Рис. 4.29. Описание инструкций по созданию новых объектов проектируемой базы данных (фрагмент: генерация SQL-запроса на создание таблицы «Студент» с созданием связи с таблицей «Группа»)

Вопросы для обсуждения

1. Факторы, оказывающие влияние, на разработку мобильных приложений.

2. Мобильное приложение в качестве одного из каналов коммуникаций и продаж.

3. Способы распространения мобильных приложений.

4. Принципы разработки мобильных приложений.

5. Характеристики качества мобильных приложений.

6. Основные методологические подходы к проектированию мобильных приложений

7. Функционально-модульный подход к проектированию мобильных приложений: сущность, преимущества и недостатки

8) Метод функционального моделирования.

9) Метод описания происходящих в системе процессов.

10) Метод описания потока данных (DFD).

11. Объектно-ориентированный подход к проектированию мобильных приложений: сущность, преимущества и недостатки.

12. Метод объектно-ориентированного системного анализа

13. Методы объектно-ориентированного анализа.

14. Метод структурного проектирования.

15. Методология объектно-ориентированного анализа и проектирования.

16. Технология объектного

17. Объединенный метод UML: сущность, преимущества и недостатки.

18. Метод определения распределенных объектов на основе объектной модели CORBA.

19. Моделирование требований к системе.

20. Моделирование состояние компонентов системы.

21. Моделирование поведения компонентов системы.

22. Процесс разработки интерфейса мобильного приложения

23. Разработка дизайна взаимодействия с пользователем (UX-дизайн).

24. Разработка дизайна интерфейса мобильного приложения (UI-дизайн)

25. Методы моделирования пользовательских сценариев.

26. Методы прототипирования пользовательского интерфейса.

27. Инфологическое проектирование базы данных мобильного приложения.

28. Даталогическое проектирование базы данных мобильного приложения.

29. Проектирование физической организации базы данных мобильного приложения.

30. Модель «сущность-связь»

Практические задания

Задание 4.1.

Составьте следующие диаграммы для мобильного приложения, образ которого был описан при решении задания 3.8:

1) функциональную модель в нотации IDEF0;

2) модель процессов, происходящих в системе в нотации IDEF3;

2) модель потока данных в нотации DFD;

Задание 4.2.

Составьте диаграмму вариантов использования для мобильного приложения, образ которого был описан при решении задания 3.8.

Для одного из вариантов использования (на своё усмотрение) составьте следующие диаграммы:

1) диаграмму классов;

2) диаграмму последовательности.

Задание 4.3.

На основании решения заданий 4.1 и 4.2 составьте пользовательский сценарий мобильного приложения в форме потока задач.

Осуществите прототипирование мобильного приложения в форме:

1) скетчингов;

2) вайрфреймов;

3) интерактивных макетов.

Задание 4.4.

На основании решения заданий 4.1, 4.2 и 4.3 осуществите инфологическое моделирование предметной области, в т.ч.:

1) идентификацию и описание сущностей;

2) определение и описание атрибутов;

3) определение и описание отношений связи между сущностями;

4) составьте ER-диаграмму концептуального уровня.

Задание 4.4.

На основании решения задания 4.4 осуществите даталогическое моделирование предметной области, в т.ч.:

1) преобразование элементов инфологической модели в элементы даталогической модели по формальным правилам;

2) проверку модели с помощью правил нормализации;

3) определение требований поддержки целостности данных;

4) составьте ER-диаграмму логического уровня.

Задание 4.5.

На основании решения задания 4.4 осуществите даталогическое моделирование предметной области, в т.ч.:

1) выбор конкретную СУБД для построения базы данных;

2) ведение контролируемой избыточности;

3) перенос логической схемы данных в среду целевой СУБД;

4) реализация бизнес-правил и требований ссылочной целостности;

5) определение индексов;

6) разработка механизмов защиты.

Задание 4.6.

На основании решения задания 4.5 сформулируйте SQL-запросы на диалекте целевой СУБД, в т.ч.:

1) набор инструкций определения данных (создания всех таблиц, создание всех индексов, добавление нового поля, изменение существующего поля, удаление существующего поля, удаление индекса, удаления всех таблиц);

2) набор инструкций определения доступа к данным (создание пользователей, создание групп пользователей, добавление пользователей в группу, наделение пользователей определенными привилегиями при работе с отдельными объектами базы данных).

Задание 4.6.

На основании решения задания 4.5 сформулируйте SQL-запросы на диалекте целевой СУБД, которые позволяют:

1) осуществить добавление одной записи в каждую таблицу базы данных; осуществить редактирование одной записи из любой таблицы; удалить отредактированную запись;

2) продемонстрировать возможность координации совместного использования данных работающими одновременно пользователями;

3) осуществить простую выборку данных из базы данных, используя следующие запросы: а) без дополнительных ключевых слов (все поля таблицы, отдельные поля таблицы с указанием псевдонима, отдельные поля таблицы с использование арифметических выражений); б) с использованием ключевого слова WHERE (с использованием не менее 2 условий); в) с использованием ключевого слова GROUP BY; г) с использованием ключевого слова GROUP BY и HAVING; д) с использованием ключевого слова ORDER BY; е) с использованием клю-

чевых слов WHERE, GROUP BY и HAVING, ORDER BY; 4) осуществить выборку данных из нескольких таблиц базы данных, используя следующие операции соединения: а) внутреннее соединение; б) внешнее соединение; в) перекрестное; г) внутреннее соединение с использованием ключевых слов WHERE, GROUP BY и HAVING, ORDER BY;

5) осуществить перекрестные запрос на основе: а) одной таблицы данных; б) нескольких таблиц данных с использованием ключевых слов WHERE, GROUP BY и HAVING, ORDER BY.

Библиографический список

1. Маркин Д.О., Комашинский В.В., Баранов И.Ю. Модель управления профилем защиты мобильного устройства при доступе к услугам с разным уровнем конфиденциальности // Информационные технологии. – 2015 – 21(8). – с. 611- 618

2. Мостяев, А. И. Социальные особенности разработки мобильных приложений / А. И. Мостяев // Программные продукты и системы. – 2019. – № 2. – С. 238-243. – EDN SCIXJA.

3. Вендров А.М. Проектирование программного обеспечения экономических информационных систем: Учебник. - 2-е изд., перераб. и доп. — М.: Финансы и статистика, 2006. — 544 с: ил. - ISBN 5-279-02937-8

4. Zhang, D. and Adipat, B., (2009). Challenges, Methodologies, and Issues in the Usability Testing of Mobile Ap-plications, International Journal of HumanComputer Interaction, 18(3), 293–308 [online]. Available at: http://dx.doi.org/10.1207/s15327590ijhc1803_3 [Accessed 1st July 2014].

5. ПНСТ 277-2018. Российская система качества. Сравнительные испытания мобильных приложений для смартфонов. М., Стандартинформ, 2018.

6. Марка Дэвид А. Методология структурного анализа и проектирования SADT [Электронный ресурс] / Дэвид А. Марка, КлементМакГоуэн. URL: http://www.interface.ru/home.asp?artId=4449

7. Функциональное моделирование на базе стандарта IDEF0: метод. указания / сост. Д.Ю. Киселев, Ю.В. Киселев, А.В. Вавилин. – Самара: Изд-во СГАУ, 2014. – 20 с.

8. Создание функциональной модели информационной системы с помощью CASE-средства CA ERwin Process Modeler 7.3. – Пенза: ПГУ, 2010. – 66 с.

9. Цуканова О. А. Методология и инструментарий моделирования бизнес-процессов: учебное пособие – СПб.: Университет ИТМО, 2015. – 100 с.

10. Моделирование бизнес-процессов с помощью IDEF0, DFD, BPMN за 7 дней: учеб. пособие / И.В. Миндалѐв; Краснояр. гос. аграр. ун-т. – Красноярск, 2018. – 123 с

11. Черушева, Т. В.Проектирование программного обеспечения : учеб. пособие / Т. В. Черушева. – Пенза : Изд-во ПГУ, 2014. – 172 с. – ISBN 978-5-94170-859-8

12. Иванов Д. Ю., Новиков Ф. А. Основы моделирования на UML: Учеб. пособие. – СПб.: Изд-во Политехн. ун-та, 2010. – 249с

13. Bing Chen. Mobile APP Interface Design Process Analysis / Proceedings of the 2018 International Workshop on Education Reform and Social Sciences (ERSS 2018)

14. Богданова, В. С. Принципы разработки интерфейса мобильного приложения / В. С. Богданова // Наука и производство Урала. – 2021. – Т. 17. – С. 108-110. – EDN UTKZUG.

15. Демидов Дмитрий Григорьевич, Костакова Елена Сергеевна Требования к современным пользовательским интерфейсам // Вестник МГУП. 2016. №1. URL: https://cyberleninka.ru/article/n/trebovaniya-k-sovremennym-polzovatelskim-interfeysam (дата обращения: 29.05.2022)

16. Мобильный интерфейс: разработка дизайна приложений и проектирование макетов на приме-рах: https://yasno.mobi/blog/mobilnyy-interfeys-razrabotka-dizayna-prilozheniy-i-proektirovanie-maketov-na-primerakh/

17. Стерлягов С.П., Селютина Е.П. Применение user Experience/User interface моделирования для разработки мобильного приложения // МНИЖ. 2017. №8-3 (62). URL: https://cyberleninka.ru/article/n/primenenie-user-experience-user-interface-modelirovaniya-dlya-razrabotki-mobilnogo-prilozheniya (дата обращения: 01.06.2022).

18. Мархакшинов Аюр Лувсаншаравович, Тонхоноева Антонида Антоновна, Урмакшинова Елена Ронославовна Разработка бессерверных мобильных приложений // Вестник НГУ. Серия: Информационные

технологии. 2019. №4. URL: https://cyberleninka.ru/article/n/razrabotka-besservernyh-mobilnyh-prilozheniy (да-та обращения: 02.06.2022).

19. REST [Электронный ресурс] // Википедия: портал. URL: https://ru.wikipedia.org/wiki/REST (дата обращения: 02.05.2022).

20. SQL [Электронный ресурс] // Википедия: портал. URL: https://ru.wikipedia.org/wiki/SQL (дата обращения: 02.05.2022).

21. Database File Format [Электронный ресурс] // SQLite: портал. URL: https://www.sqlite.org/fileformat.html (дата обращения: 02.05.2022).

Глава 5. СРЕДА ИСПОЛНЕНИЯ МОБИЛЬНЫХ ПРИЛОЖЕНИЙ (НА ПРИМЕРЕ ОС ANDROID)

В данной главе рассматриваются следующие вопросы:

1. Архитектура ОС Android;

2. Архитектура приложений ОС Android;

3. Архитектура среды исполнения приложений в ОС Android

5.1. Архитектура ОС Android

Рассмотрим наиболее распространённую в настоящий момент операционную систему Android.

Операционная система (далее – ОС) Android представляет из себя модульную операционную систему, состоящую из отдельных взаимосвязанных относительно простых модулей (см. рис. 5.1). Особенности архитектуры ОС Android определяются спецификой её применения: установка на мобильные устройства разных производителей [1, 2].

Основными модулями ОС Android являются [1,3]:

1) ядро (Linux Kernel);

2) уровень аппаратных абстракций HAL (Hardware Abstraction Layer);

3) набор библиотек и среда исполнения (исполняющая система);

4) каркас программ (Java API FrameWork);

5) уровень системных и пользовательских приложений.

Модули перечислены в порядке следования от нижнего уровня архитектуры к верхнему.

Ядро (Linux Kernel) является основой модульного строения операционной системы и координирует выполнение большинства её базовых операций: управление памятью, энергосистемой и процессами, обеспечение безопасности, работу с сетью и драйверами и т.д.

Ядро ОС Android представляет собой «усеченное» ядро Linux, которое дополнено возможностями, необходимыми для эффективного управления питанием и ресурсами (так как это требуется для мобильных устройств, которые обычно работают от своего аккумулятора).

Уровень аппаратных абстракций (HAL) представляет собой созданный производителями мобильных устройств слой программного

обеспечения, который скрывает (или абстрагирует) особенности и различия аппаратных компонентов от верхних уровней операционной системы.

HAL состоит из нескольких библиотечных модулей, каждый из которых реализует интерфейс для определенного типа аппаратного компонента (например, камера, bluetooth и т.д.).

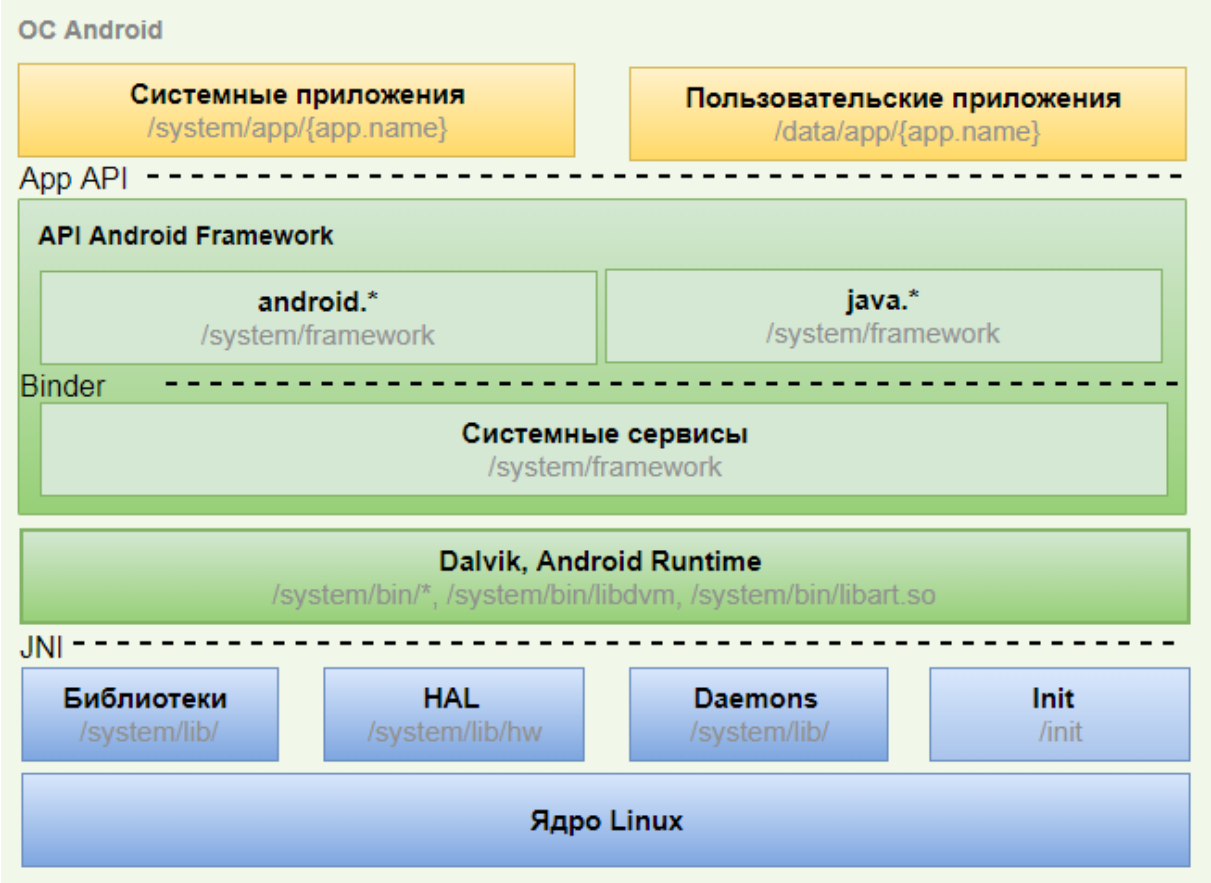


Рис. 5.1. Архитектура ОС Android

Благодаря обеспечиваемому уровнем аппаратных абстракций фильтру, различные аппаратные средства выглядят аналогично с точки зрения операционной системы; снимается необходимость специальной настройки операционной системы под используемое оборудование. Такой подход позволяет обеспечить легкую переносимость Android с одной аппаратной платформы на другую, т.е. использование на мобильных устройствах разных производителей [1].

Среда исполнения (Android Runtime) включает в себя библиотеки ядра, за счет которых обеспечивается значительная часть низкоуровневой функциональности и виртуальную машину (далее – ВМ), позволяющую запускать приложения, реализованные на языке Java [1].

Библиотеки ядра содержат базовые библиотеки Java, необходимые для работы Android API Framework и запуска приложений.

Архитектура среды исполнения реализована таким образом, что каждое приложение запускается в своем экземпляре программной среды, что позволяет обеспечить их изоляцию друг от друга (если сработает код с ошибками или вредоносное программное обеспечение, то они не смогут вывести из строя ОС).

Многие базовые компоненты и сервисы Android, такие как ART и HAL, построены из собственного кода, для которого требуются собственные библиотеки, написанные на языках C и C ++.

Библиотеки выполняют следующие функции [1]:

1) предоставляют реализованные алгоритмы для вышележащих уровней;

2) обеспечивают поддержку файловых форматов;

3) осуществляют кодирование и декодирование информации;

4) выполняют организацию отображения графики и т.д.

Библиотеки, как привило, скомпилированы под определенное аппаратное обеспечение мобильного устройства, с которым они поставляются производителем в предустановленном виде.

Android API Framework – программная платформа, определяющая структуру программной системы. Диктует правила построения архитектуры системных и пользовательских приложений.

Представляет собой строительные блоки необходимые для создания приложений для Android, что упрощает их разработку, за счет повторного использования основных системных компонентов и сервисов.

Архитектура ОС Android является фреймворк-ориентированной (framework-based): приложение не может быть запущено вне фреймворка или без него [4].

Уровень программ представляет набор предустановленных базовых системных программ и прикладных программ пользователя, которые он может инсталлировать самостоятельно, используя, в том числе, встроенный облачный сервис с каталогом приложений.

5.2. Архитектура приложений ОС Android

Программы (или приложения) пишутся на языке программирования Java или Kotlin и распространяются в формате архивных исполняемых файлов-приложений (Android Package, APK) [5, 6].

Каждое приложение Android скомпилировано и упаковано в один файл, который включает в себя всё, что необходимо для работы Android-приложения и позволяет установить приложение на любом устройстве под управлением системы Android:

1) описание приложения (файл AndroidManifest.xml);

2) байт-код приложения (один или несколько файлов в формате .dex);

3) цифровой сертификат приложения, удостоверяющий его создателя (файл CERT.RSA в каталоге META-INF);

4) контрольные суммы файлов приложения (файлы CERT.SF и MANIFEST.MF в каталоге META-INF);

5) описание ресурсов приложения в бинарном формате (файл – resources.arsc) ;

6) ресурсы приложения (декларативное описание интерфейса, изображения и другие данные в каталоге res и assets, соответственно имеющий сгенерированный идентификатор или нет);

7) файлы нативного программного кода (файлы Linux-библиотек в формате .SO, написанные на C/C++).

Файлы APK не шифруются, являются подмножеством формата архива ZIP. Упаковка осуществляется с помощью утилиты Android Asset Packaging Tool (aapt). Структура файла APK приведена на рис. 5.2.

При необходимости для обеспечения проверки авторства и целостности файл APK может быть подписан. Для этого в его структуру дополнительно встраивается файл электронной подписи, содержание которого основано на контрольной сумме содержимого .apk и отдельно сгенерированном закрытом ключе. Для подписи файла APK используется стандартный инструмент jarsigner, а затем для обеспечения побайтового выравнивания его содержимого используется инструмент zipalign.

Рассмотрим подробнее несколько компонентов APK, которые реализуют логику работы приложения.

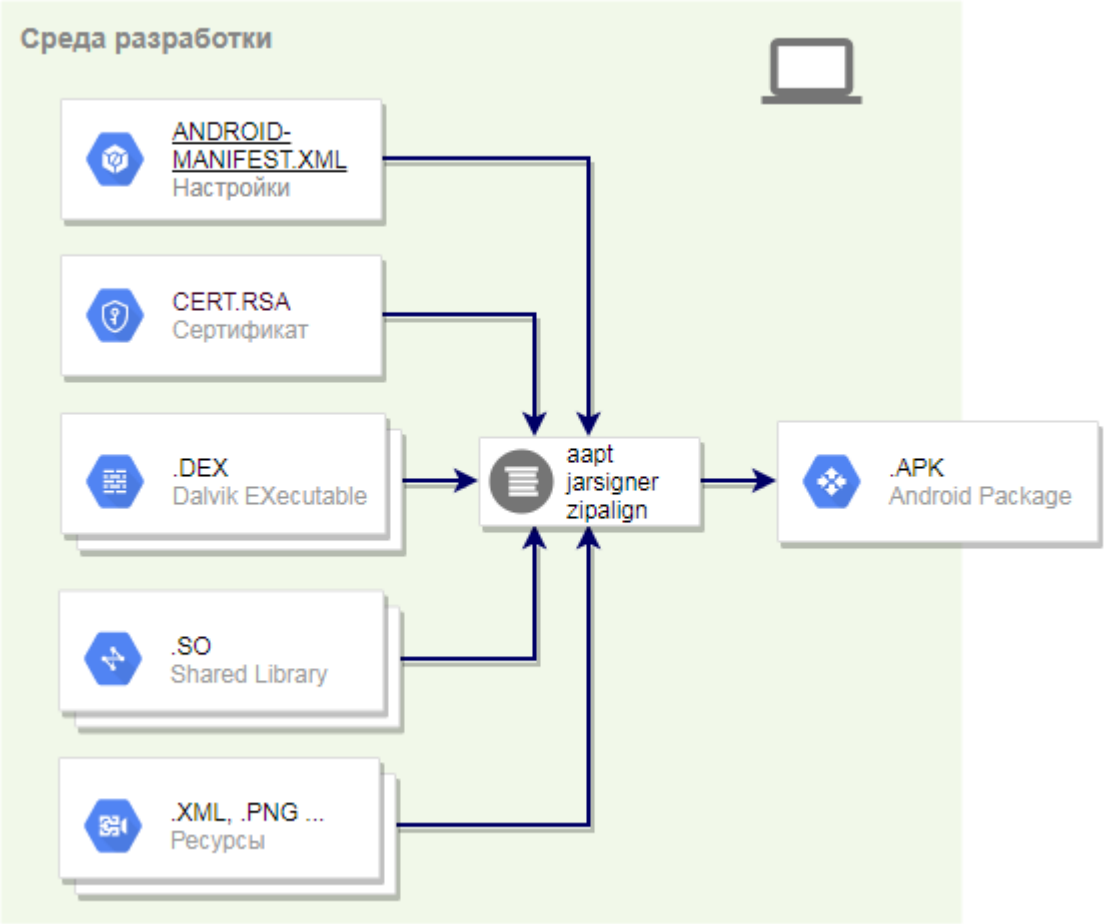


Рис. 5.2. Структура файла формата APK

Файлы формата .dex (Dalvik Executable) используются для хранения набора определений классов приложения и связанных с ними дополнительных данных, которые скомпилированы в байт-код.

Байт-код – стандартное промежуточное представление, в которое может быть переведена компьютерная программа автоматическими средствами с применением синтаксического и семантического анализа, для исполнения не реальным процессором, а ВМ.

По сравнению с исходным кодом на языке Java (или Kotlin), удобным для создания и чтения человеком, в байт-коде в явном виде закодированы типы, области видимости и другие конструкции. С технической точки зрения, байт-код представляет собой машинно-независимый код низкого уровня, генерируемый транслятором из исходного кода. Длина каждого кода операции в байт-коде составляет один байт.

В качестве ВМ в ОС Android до версии 4.4 использовался Dalvik (регистровый интерпретатор языка программирования Java, дополнен-

ный с версии 2.2 JIT компилятором). Соответственно байт-код, включенный в файлы DEX, состоит из инструкций для данной регистровой ВМ.

Начиная с версии 5 ОС Android в среде исполнения ART применятся AOT-компилятор, транслирующий байт-код в машинный код предварительно, до исполнения. В следствии этого, файлы формата .dex используются исключительно в качестве формата распространения программного кода, обеспечивая обратную совместимость новых приложений с устройствами под управлением ОС Android с ВМ Dalvik.

Создание файла в формате .dex предполагает компиляцию исходного кода на языке Java в байт-код Java (предназначенный для стекового интерпретатор языка программирования Java – Java Virtual Machine), а затем его компиляцию в байт-код Dalvik с применением утилит dx или v8.

Т.е. в ОС Android не используется байт-код Java, а используется иной подход (файл формата .dex и свой набор инструкции байт-кода).

Пример исходного кода на языке Java, байт-кода Java и пример байт-кода Dalvik приведены соответственно на рис. 5.3, 5.4 и 5.5.

```
public MainActivity() {
  super();
  currentPosition = 0;
}
```

Рис. 5.3. Пример исходного кода на языке Java (.java)

Файлы нативного программного кода (native code) представляют собой файлы, содержащие код, написанный на языках Assembler/ C/C++ и скомпилированный непосредственно в машинный код требуемой процессорной архитектуры в виде общих объектов (динамически связанных библиотек) ОС Linux в формате .so. Общие объекты позволяют не использовать статическое связывание, что даёт возможность вызова функции C/C++ из программы на Java, и наоборот.

```
public com.hfad.bitsandpizzas.MainActivity();
  Code:
    0:   aload_0
    1:   invokespecial   #5; //Method android/app/Activity."<init>":
()V
    4:   aload_0
    5:   iconst_0
    6:   putfield        #3; //Field currentPosition:I
    9:   return
```

Рис. 5.4. Пример байт-кода Java (.class)

```
0x0000: iput-object v1, v0, Lcom/hfad/bitsandpizzas/MainActivity;
com.hfad.bitsandpizzas.MainActivity$2.this$0 // field@4869
0x0002: invoke-direct {v0}, void java.lang.Object.<init>() //
method@13682
0x0005: return-void
```

Рис. 5.5. Пример байт-кода Dalvik (.class)

Файлы нативного программного кода используются для увеличения производительности приложения, или интеграции в приложение с минимальными затратами существующего кода, написанного на Assembler/ C/C++ (например, при разработке игр).

Файл AndroidManifest.xml содержит информацию о приложении, которая необходима ОС Android для его выполнения. В этом файле, который должен находиться в корневой папке приложения, должны быть объявлены:

1) уникальный идентификатор приложения;

2) компоненты приложения (операции, службы, приемники широковещательных сообщений и поставщиков контента);

3) реализующие каждый компонент имена классов и их параметры;

4) перечень требуемых приложению разрешений для доступа к защищенным частям API-интерфейса и взаимодействия с другими приложениями;

5) перечень требуемых для взаимодействия с компонентами данного приложения разрешений;

6) содержит связанный с приложением перечень библиотек; и др.

Обязательными элементами манифеста являются только элементы <manifest> и <application>. Оба они должны присутствовать в файле манифеста, при этом указать их можно только один раз.

ОС Android предоставляет пользовательским приложениям четыре типа поведения: активность (activity), получатель (receiver), служба (service) и поставщик контента (content provider). Каждый из четырех типов компонентов приложения может содержать свою, отличную от других семантику и указание на порядок использования внутри системы.

5.3. Архитектура среды исполнения приложений в ОС Android

Среда исполнения – инфраструктура, необходимая для запуска, выполнения и завершения программ. Понятие среды исполнения относится к совокупности ресурсов программного и аппаратного обеспечения, которые позволяют программному обеспечению выполняться в компьютерной системе.

Среда исполнения – это составной механизм операционной системы, разработанный для предоставления возможности выполнения программ, независимо от используемого языка программирования.

Среда исполнения обеспечивает выполнение программных инструкций как высокого уровня (взаимодействуя с основной структурой / библиотекой программного обеспечения), так и низкого уровня (взаимодействуя с базовой архитектурой набора аппаратных команд (instruction set architecture, ISA)).

Функции высокого уровня включают в себя проверку типов, генерацию кода, сборку мусора, а также услуги по отладке и оптимизации. Низкоуровневые функции, предоставляемые средой исполнения, включают: управление процессором, управление памятью, взаимодействие с внешними устройствами ввода/ вывода и т.д.

Для интерпретируемых и компилируемых языков программирования высокого уровня взаимодействие со средой исполнения организовано по-разному: для первых – непосредственно в интерпретаторе, в котором запущен код программы; для вторых – либо в выполняющей промежуточный код ВМ, либо с использованием набора подключаемых разделяемых библиотек среды выполнения.

В ОС Android Java код изначально переводится в интерпретируемую форму (байт-код), а затем используется компиляция для генерации машинного кода. При этом, во-первых, приложение не может быть запущено вне Android API Framework или без него, а во-вторых, в процессе генезиса ОС Android подходы к организации архитектуры среды исполнения приложений (далее – СИП) пересматривались, и для генерации машинного кода использовалось несколько взаимоисключающих подходов.

СИП в ОС Android у каждого приложения своя и загружается в память устройства в процессе запуска приложения. СИП выполняет следующие основные функции:

1) подготовка приложений к выполнению на устройстве (оптимизация и/или компиляция);

2) процесс установки приложения;

3) выполнение приложений на конкретной платформе устройства (интерпретация или выполнение машинного кода)

По мере развития ОС Android подходы к организации архитектуры СИП в контексте выполнения указанных функций подвергались неоднократному пересмотру. К настающему моменту представляется возможным выделить следующие подходы [1]:

1) применение высокоуровневой ВМ, интерпретирующей байт-код (Dalvik, ОС Android до версии 2.1);

2) применение высокоуровневой ВМ, интерпретирующей байт-код с одновременным использованием JIT компиляции (Dalvik, ОС Android до версии 4.4);

3) применение низкоуровневой ВМ и набора подключаемых разделяемых библиотек с осуществлением полной OAT компиляции байт-кода (ART, ОС Android до версии 6);

4) применение низкоуровневой ВМ и набора подключаемых разделяемых библиотек с осуществлением частичной OAT компиляции байт-кода с применением JIT-компилятора (ART, современные версии ОС Android). Основные подходы к организации среды приведены на рис. 5.6.

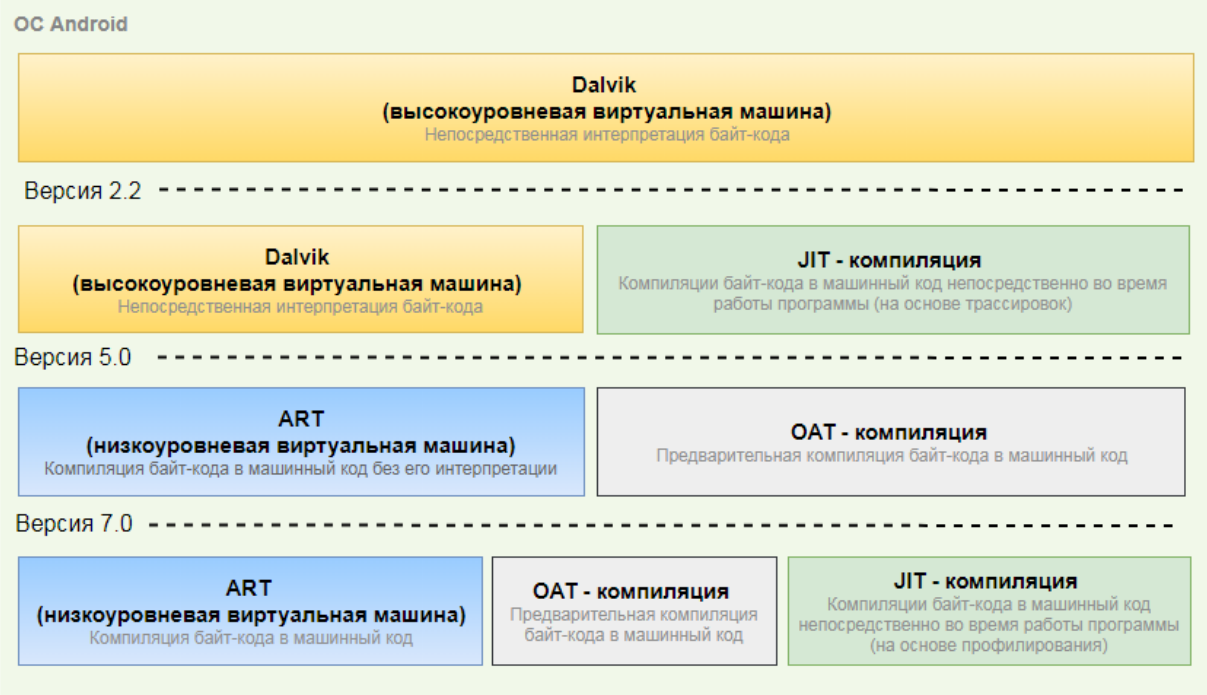


Рис. 5.6. Пример байт-кода Dalvik (.class)

Проведем анализ основных архитектур Dalvik и ART, а также сравним их между собой.

Dalvik – регистровая ВМ, предназначенная для выполнения программ, написанных на языке программирования Java (в отличие от стандартной ВМ Java, принадлежащей Oracle и являющейся стек-ориентированной). Строго говоря, Dalvik не является ВМ Java, так как она использует другой формат байт-кода.

Данная ВМ оптимизирована для низкого потребления и совместного использования памяти, а также ориентирована на процессоры RISC-архитектур, часто используемые в мобильных и встраиваемых устройствах, таких как коммуникаторы и планшетные компьютеры. Именно эти оптимизации позволили Dalvik превалировать, несмотря на строгие ограничения мобильных платформ, которые не позволили Java (в частности, J2ME) завоевать позиции за пределами ограниченных реализаций.

Чтобы максимально использовать ограничения со стороны низкого размера памяти и скорости процессора мобильного устройства, а также избежать сбои во время исполнения, файлы формата .dex проверяются на соответствие спецификации Dalvik и оптимизируются перед их интерпретацией ВМ. Это обычно происходит только один раз, когда

приложение запускается впервые на устройстве под управлением ОС Android с помощью системной утилиты dexopt.

Результатом процесса оптимизации является файл Optimized DEX (.odex). Файлы формата .odex не переносимы между различными версиями Dalvik или между устройствами.

Начиная с версии 2.2 в ОС Android добавлен JIT-компилятор (Just In Time, точно в срок), позволяющий оптимизировать выполнение приложений путем постоянного их профилирования при каждом запуске и динамической компиляции часто выполняемых коротких сегментов их байт-кода в собственный машинный код. В то время как Dalvik интерпретирует остальную часть байт-кода приложения, собственное выполнение этих коротких сегментов байт-кода, называемых «трассировками», обеспечивает значительное повышение производительности.

В версиях, начиная с Android 4.4 Kitkat, имеется возможность переключиться с Dalvik на ART (Android Runtime). В Android 5.0 Dalvik был полностью заменён на ART.

В отличие от Dalvik, ART предлагает использовать опережающую компиляцию (Ahead-of-Time, AOT), компилируя целые приложения в нативный код после их установки, и предполагает поддержку 64 разрядных архитектур. Исключая интерпретацию Dalvik и JIT-компиляцию на основе трассировки, ART повышает общую эффективность выполнения и снижает энергопотребление, что приводит к повышению автономности батареи на мобильных устройствах. В то же время ART обеспечивает более быстрое выполнение приложений, улучшенные механизмы выделения памяти и сборки мусора (GC), новые функции отладки приложений и более точное высокоуровневое профилирование приложений.

Для обеспечения обратной совместимости ART использует тот же входной байт-код, что и Dalvik, предоставляемый через стандартные файлы .dex как часть файлов APK , а файлы .odex до версии 7 ОС Android заменяются исполняемыми файлами .aot, содержащими как байт-код Dalvik, так и нативный код, подходящий для аппаратной архитектуры, работающей на устройстве. После того, как приложение скомпилировано на устройстве, оно запускается исключительно из скомпилированного исполняемого файла. Процесс установки приложения в версиях 5 и 6 ОС Android приведен на рис. 5.7

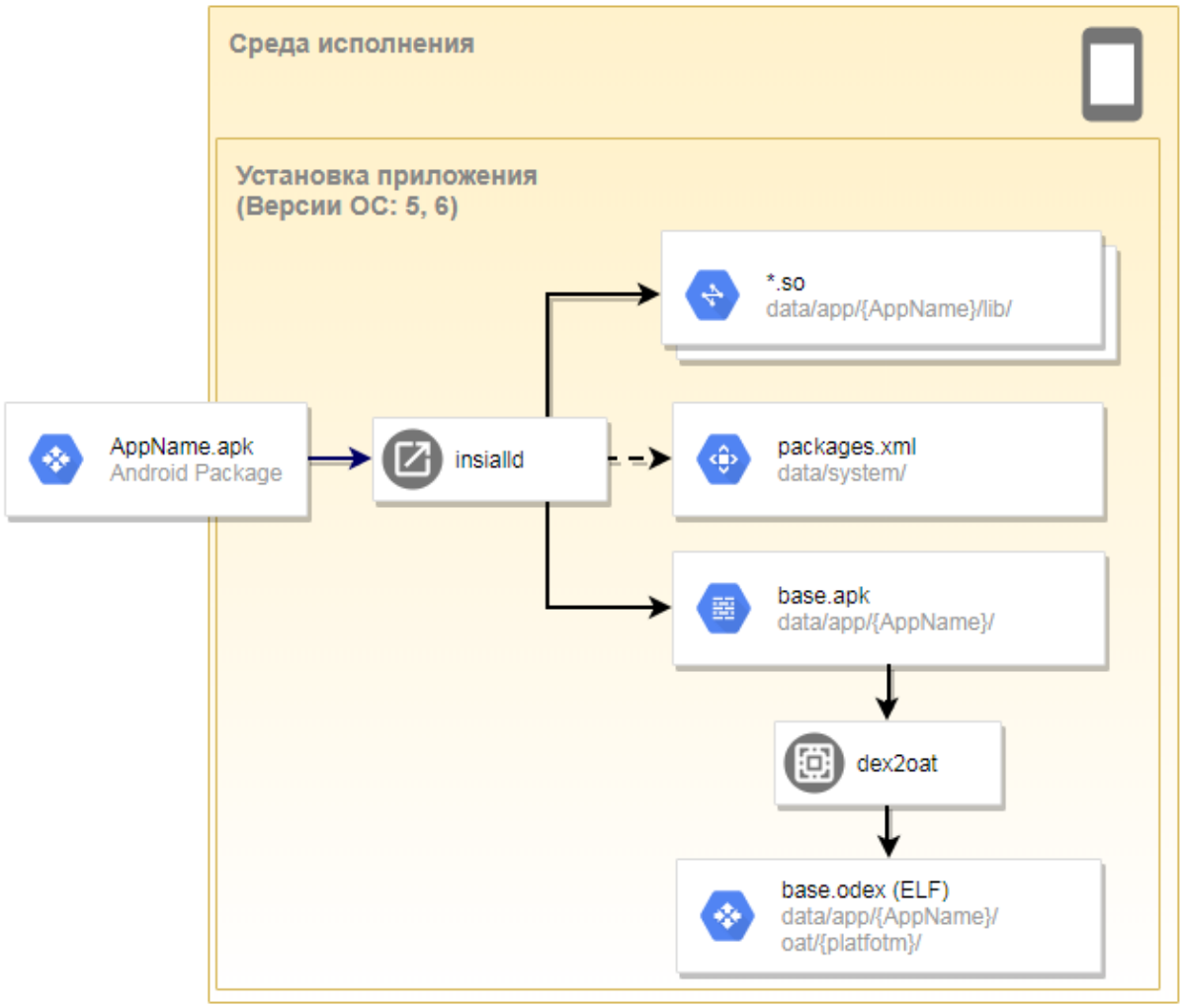


Рис. 5.7. Установка приложения в ОС версии 5 и 6

Формат OAT (Android RunTime Code) – это специальный формат ELF с некоторыми расширениями. В файле OAT есть раздел oatdata, в котором содержится информация о каждом классе, скомпилированном в собственный код. Собственный код находится в специальной секции со смещением, обозначенным символом oatexec. Соответственно, поиск информации о классе Java в разделе oatdata и его скомпилированном собственном коде можно провести через символ oatexec.

Когда приложение запускается, среда выполнения ART анализирует OAT-файл и загружает его в память. Для каждого объекта класса Java среда выполнения ART имеет соответствующий экземпляр класса C ++ Object для его представления. Первый член этого экземпляра указывает на экземпляр класса C ++ Class, который содержит подробную информацию о классе Java, включая поля, методы и т. д.

Каждый метод Java представлен экземпляром класса C ++ ArtMethod, который содержит адрес метода, права доступа, класс, к которому принадлежит этот метод, и т. д. Класс C++ ArtField используется для представления поля класса, включая класс, к которому принадлежит это поле, индекс этого поля в своем классе, права доступа, и т.д.

С ОС Android 7.0 представлен JIT-компилятор с профилированием кода для ART, который позволяет постоянно повышать производительность приложений Android при их запуске. Компилятор JIT дополняет нынешний компилятор Ahead of Time от ART и помогает улучшить производительность во время выполнения. Комбинация указанных режимов может быть настроена в процессе установки операционной системы.

Процесс компиляции приведен на рис. 5.8 и выгладит следующим образом:

1) приложение изначально устанавливается без какой-либо компиляции AOT (первые несколько раз приложение запускается, интерпретируется, и часто выполняемые методы компилируются);

2) когда устройство находится в режиме ожидания и заряжается, демон компиляции выполняет AOT-компиляцию часто используемого байт-кода на основе профиля, сгенерированного во время первых запусков;

3) при следующем перезапуске приложения будет использоваться код, управляемый профилем, и будет избегаться JIT-компиляция во время выполнения для уже скомпилированных методов, а методы, которые JIT-компилируются во время новых запусков, будут добавлены в профиль, который затем будет выбран демоном компиляции.

После включения в ОС Android гибридного OAT/JIT-компилятора количество файлов, их расширения и имена изменялись в разных выпусках, но начиная с версии 8 ОС Android генерируются следующие файлы: .vdex: содержит несжатый DEX-код APK, а также некоторые дополнительные метаданные для ускорения проверки (подробное описание); .odex : содержит скомпилированный код AOT для методов в APK; .art (необязательно): содержит внутренние представления ART

некоторых строк и классов, перечисленных в APK, которые используются для ускорения запуска приложения. Процесс установки приложения для версий ОС Android, начиная с 7, приведен на рис. 5.9.

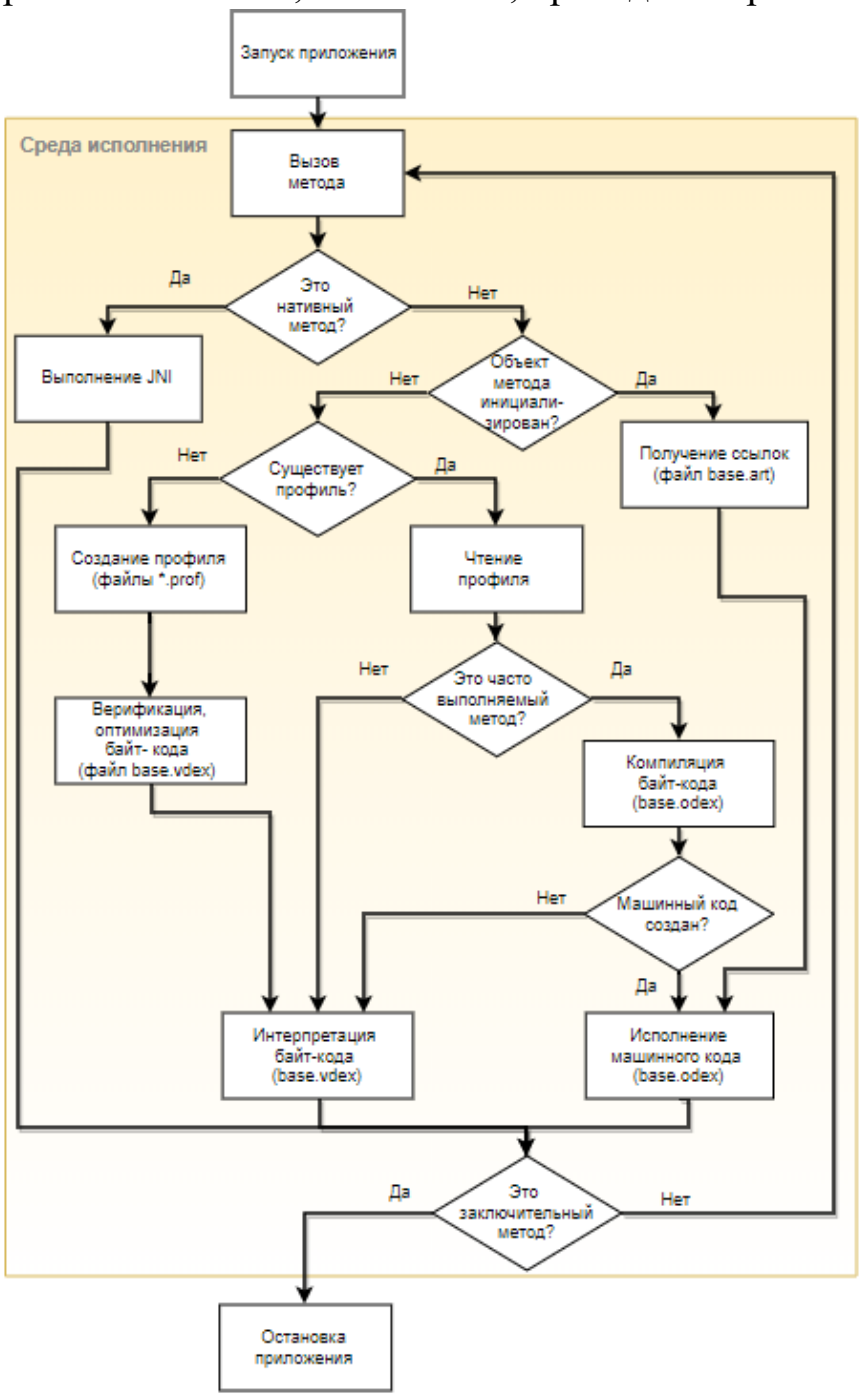


Рис. 5.8. Исполнение приложения в ОС с версии 7

Для компиляции файла .dex ART использует инструмент dex2oat, который использует два бэкэнда компилятора, каждый из которых использует метод компиляции AOT (с/без JIT). Один из них улучшен с помощью архитектуры LLVM (Low Level Virtual Machine, низкоуровневой ВМ), написанной на C++. В отличии от dexopt, который выполняется в один поток, dex2oat выполняется параллельно сразу в несколько потоков

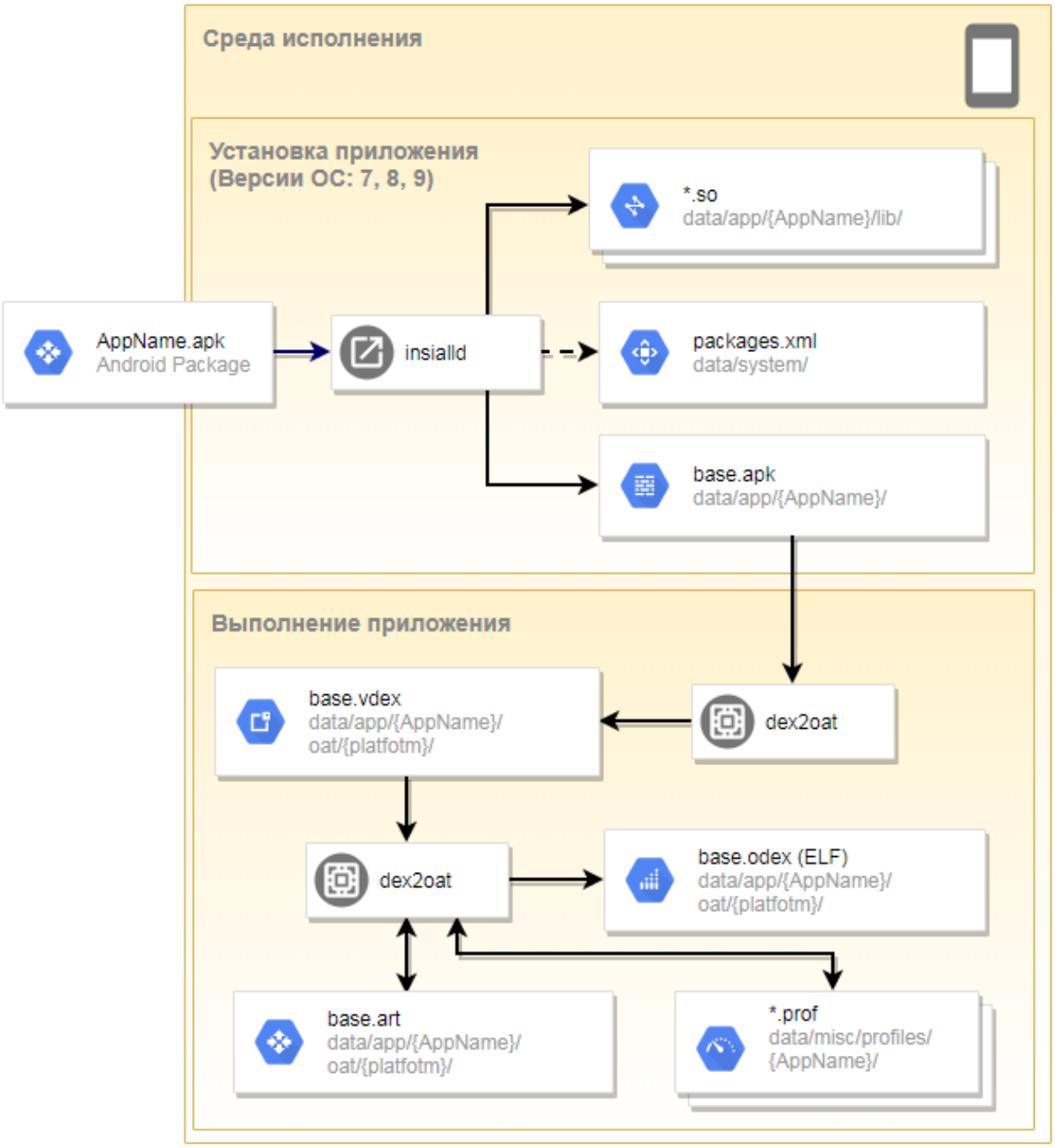


Рис. 5.9. Установка приложения в ОС версии 7, 8 и 9

Одним из основных вариантов ART для настройки этих двух категорий являются фильтры компилятора. Фильтры компилятора определяют, как ART компилирует код DEX и является опцией, передаваемой инструменту dex2oat.

Начиная с Android O, есть четыре официально поддерживаемых фильтра:

1) verify: только запустить проверку кода DEX;

2) quicken: запустить проверку кода DEX и оптимизировать некоторые инструкции DEX, чтобы повысить производительность интерпретатора;

3) speed: запустить проверку кода DEX и AOT-компилировать все методы;

4) speed-profile: запустить проверку кода DEX и методы AOT-компиляции, перечисленные в файле профиля.

В версии 8 ОС Android для анализа файлов .dex и их упорядочения в соответствии с профилем внедрен инструмент dexlayout. Dexlayout стремится использовать информацию профилирования во время выполнения для изменения порядка разделов файла .dex во время компиляции обслуживания в режиме ожидания на устройстве. Сгруппировав вместе части файла .dex, к которым часто осуществляется обращения во время выполнения приложения инструмент позволяет экономить оперативную память и сокращать время запуска приложения.

Поскольку информация профиля в настоящее время доступна только после запуска приложений, dexlayout интегрируется в компиляцию устройства dex2oat во время обслуживания в режиме ожидания

Процесс загрузки СИП остается неизменным при смене её архитектур и представлен на рис. 5.10

Когда система загружается, загрузчик загружает ядро в память и инициализирует системные параметры. После этого происходит несколько процессов. Ядро запускает программу Init – это родительский процесс для всех процессов в системе.

Так как устройство под управлением ОС Android не имеет консоли для доступа к оболочке, то init не запускает оболочку традиционным способом. Вместо этого демон-процесс adbd прислушивается к удаленным подключениям, запрашивающим доступ к оболочке, по необходимости ответвляя для них процессы оболочки.

Одним из низкоуровневых процессов-демонов, запускаемых init, является системный процесс app_process, который принято называть zygote (в виду его сходства с биологическим аналогом).

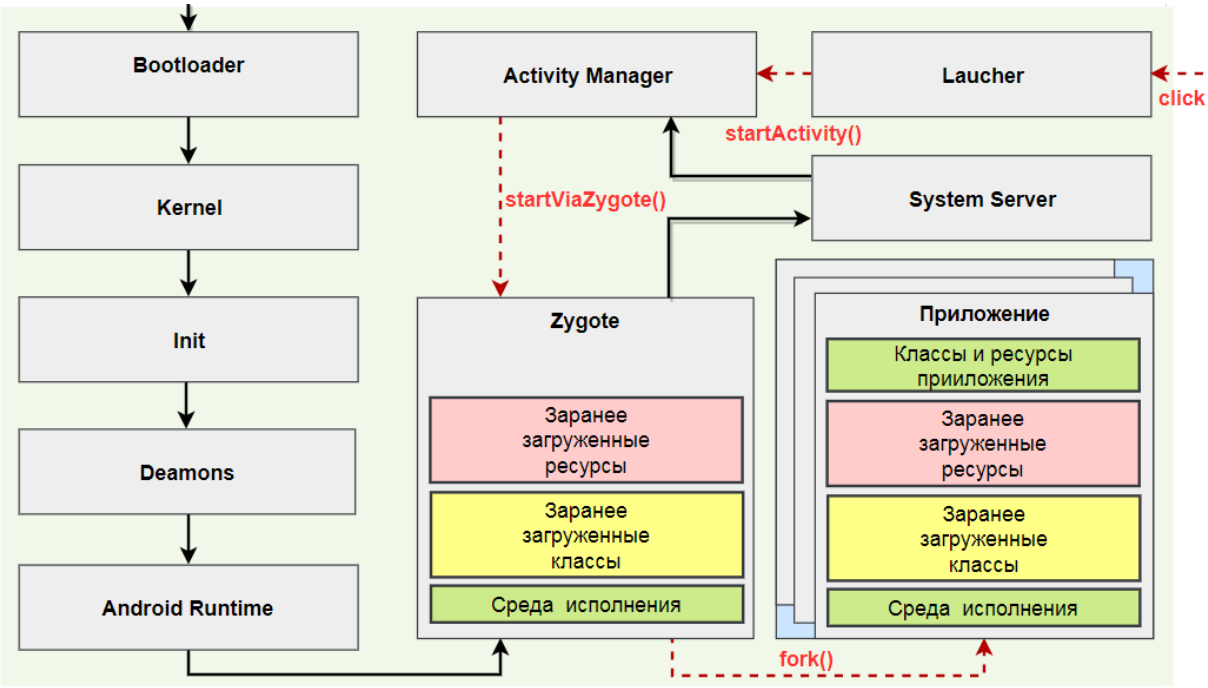


Рис. 5.10. Процесс загрузки Android и создание экземпляров среды

Демон zygote в свою очередь запускает ряд системных процессов и процессов приложений. Первым запускается процесс, запускающий все основные службы операционной системы (system_server).

Затем демон zygote связывает свой сокет (/ dev / socket / zygote) для прослушивания входящих запросов. Когда такие запросы будут поступать и будут содержать имя класса для загрузки, zygote разветвит с помощью fork() свой процесс и загрузит классы, создавая новое приложение. Но все эти приложения с точки зрения Linux в действительности будут являться экземплярами app_process, которые отбрасывают свои административные привилегии вызывая setuid() / setgid() перед загрузкой кода приложения, чтобы принять AID рассматриваемого приложения.

Все новые процессы, основанные на применении СИП (системные или прикладные), ответвляются от zygote, что позволяет им начинать выполнение с уже готовой к работе средой. Новый процесс явля-

ется точной копией исходного процесса zygote, со всеми его ранее инициализированными и уже установленными состояниями, и он сразу же готов к работе.

Zygote также осуществляет предварительную загрузку многих частей Android-среды, которые обычно используются в системе и приложениях, а также загружает ресурсы и другие часто востребуемые компоненты

Процесс zygote и все его дочерние процессы могут совместно использовать множество уже задействованных страниц оперативной памяти, необходимых для инициализации СИП и предварительной загрузки классов и ресурсов. Это совместное использование для Android-среды, где подкачка недоступна, играет особо важную роль: можно лишь потребовать подкачку чистых страниц (например, выполняемого кода) с диска (флеш-памяти), но любые уже задействованные страницы должны оставаться запертыми в оперативной памяти – они не могут быть выгружены на диск.

Загрузочный образ до ART, Android использовал Zygote для разветвления каждого процесса приложения, а также для предварительной загрузки и предварительной инициализации некоторых классов в целях оптимизации. В ART набор библиотек jar, которые должны быть предварительно загружены в каждый процесс приложения, один раз компилируется в так называемый загрузочный образ. Схема такой загрузки приведена на рис. 5.11.

В качестве инициализированной пустой СИП в ART используется файл «boot.art» (или несколько файлов «boot-{name}.art»), который содержит образ сжатой кучи предварительно инициализированных классов и связанных объектов. Базовый адрес загрузочного образа (art) фиксирован (в версии 9 он равен 0x700000). Файл boot.oat содержит скомпилированный код. Оба файла также генерируются dex2oat.

Несмотря на то, что для данного файла требуется дополнительное место для хранения, это ускоряет запуск приложения и создает возможность для системы заменять некоторые предварительно загруженные классы, способствуя повышению производительности при низкой оперативной памяти.

Когда zygote запускается, он загружает три файла (или группы файлов) /data/dalvik-cache/arm64/system@framework@boot*.] art,oat,vdex}.

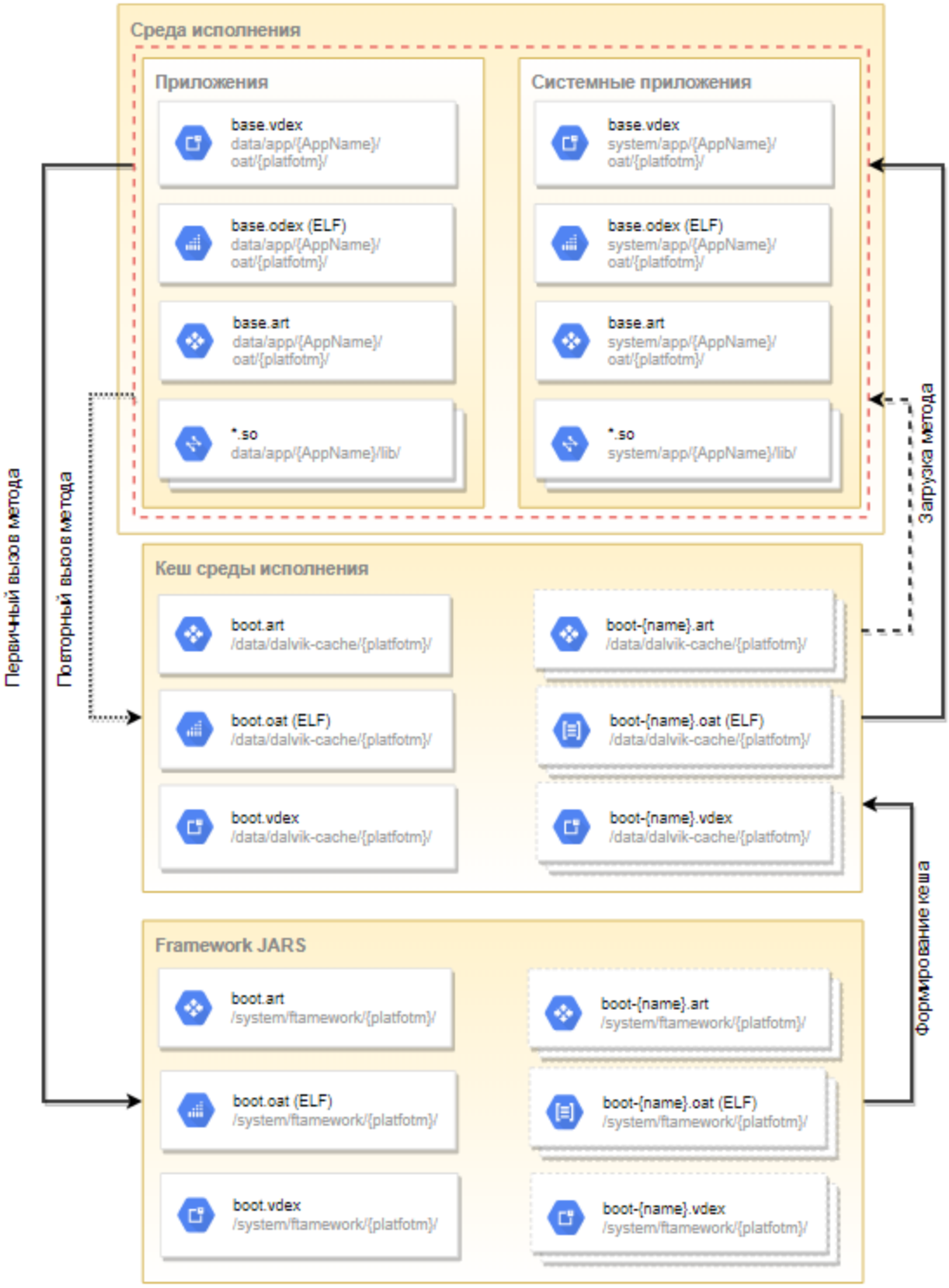


Рис. 5.11. Формирование загрузочных образов СИП

Таким образом, ART представляет собой СИП, основная идея которой состоит в том, чтобы загрузить общий объект ELF в предвари-

тельно инициализированный процесс приложения и выполнить приложение. Поскольку код уже скомпилирован, нет необходимости интерпретировать во время выполнения.

В связи с возвратом в СИП JIT интерпретация используется на начальном этапе до полной компиляции кода приложения (для этого в загрузочный образ включен файлы *.vdex, содержащий верифицированный и оптимизированный байт-кол). Файлы расположены в каталогах:

1) «/system/framework/oat/x86/*.vdex»;

2) «/system/framework/x86/*.vdex».

Временные копии файлов расположены в каталоге «data/dalvik-cache».

Также при установке приложения создается файл packages.xml, в котором прописываются все данные о приложении и который обеспечивает реализацию изоляции среды на верхнем уровне.

Вопросы для обсуждения

1. Особенности архитекторы операционной системы ОС Android.
2. Особенности реализации архитектуры среды исполнения мобильных приложений в ОС Android.
3. Архивный исполняемый файл-приложений: предназначение и состав.
4. Виртуальная машинав ОС Android.
5. Файлы нативного программного кода.
6. Файл AndroidManifest.xml: предназначение и структура.
7. Типы поведения пользовательского приложения, предоставляемые ОС Android
8. Использованием JIT и OAT компиляции
9. Процесс загрузки Android и создание экземпляров среды

Практические задания

Задание 5.1.

На основании решения заданий из главы 4 выполните описание архитектуры архивного исполняемого файла-приложения для целевой операционной системы Android.

Задание 5.2.

Сравните разные версии операционной среды Android с точки зрения способов установки и исполнения мобильных приложений.

Библиографический список

1. Воронин Алексей Александрович, Виноградов Дмитрий Викторович, Воронина Анна Аркадьевна Оценка защищенности среды выполнения операционной системы Android // Бюллетень науки и практики. 2019. №7. URL: https://cyberleninka.ru/article/n/otsenka-zaschischennosti-sredy-vypolneniya-operatsionnoy-sistemy-android (дата обращения: 02.05.2022).

2. Цифровая экономика / Н. В. Абдуллаев, Н. Л. Аванесян, Д. В. Виноградов [и др.]. – Москва : Компания КноРус, 2018. – 286 с. – ISBN 978-5-4365-3040-6. – EDN YPRLET.

3. Ломовский, А. В. Архитектура и основные компоненты платформы Android / А. В. Ломовский // Аллея науки. – 2018. – Т. 1. – № 9(25). – С. 909-912. – EDN YNUEWT.

4. Google Android: системные компоненты и сетевые коммуникации. — СПб.: БХВ-Петербург, 2012. — 384 с.: ил. — (Профессиональное рограммирование) – ISBN 978-5-9775-0666-3

5. Дейтел П., Дейтел Х., Уолд А. Android для разработчиков. 3-е изд. – СПб.: Питер, 2016. – 512 с.: ил. – ISBN 978-5-496-02371-9

6. Аделекан И. Kotlin: программирование на примерах: Пер. с англ. — СПб.: БХВ-Петербург, 2020. – 432 с. – ISBN 978-5-9775-6673-5

Глава 6. ИНФОРМАЦИОННАЯ БЕЗОПАСНОСТЬ СРЕДЫ ИСПОЛНЕНИЯ ПРИЛОЖЕНИЙ В ОС ANDROID

В данной главе рассматриваются следующие вопросы:

1. *Модель информационной безопасности среды исполнения;*
2. *Защитные механизмы среды исполнения;*
3. *Анализ подходов к проведению аудита безопасности среды исполнения*

6.1. Модель информационной безопасности среды исполнения приложений

На состояние информационной безопасности СИП существенное влияние оказывают ряд особенностей как мобильных устройств в целом [1]: 1) миниатюрность; 2) мобильность; 3) ограниченность вычислительных ресурсов; 4) мальтифункциональность (использование в качестве не только функции телефонных переговоров, но и фото- и видеокамеры, навигационного устройства, диктофона, переносной точки доступа, съемного носителя информации и др.), так и особенностей ОС Android как открытой платформы с возможностью её использования различными производителями мобильных устройств с необходимыми им модификациями.

Кроме того, на информационной безопасность (далее – ИБ) влияет ряд особенностей непосредственно СИП [2, 3]: использование в работе файлов различных форматов, использование нескольких подходов к запуску приложений, использование нескольких комбинирующих между собой подходов к верификации, оптимизации, интерпретации и компиляции кода приложения и др.).

Выбор методов и средств защиты СИП осуществляют с учетом потенциальных угроз ИБ, а также источников их возникновения.

Основные угрозы ИБ СИП соответствуют тем угрозам, которым подвержены мобильные платформы в целом: физические угрозы; аппаратные угрозы; программные угрозы системного и пользовательского уровня; сетевые угрозы, умышленное или непреднамеренное использование уязвимостей, слабых мест и ошибок, содержащихся в программных компонентах операционной системы.

В качестве источников угроз необходимо в первую очередь рассматривать вредоносное программное обеспечение, целью или средством достижения цели, которого, является нарушение информационной безопасности СИП.

При этом в качестве нарушителя информационной безопасности необходимо рассматривать как физических лиц, не имеющих физический и/или логический доступ к информационным ресурсам мобильного устройства (временных пользователей, разработчиков, внешних злоумышленников), так и имеющих такой доступ (пользователи и администраторы мобильного устройства, программисты-разработчики программного обеспечения).

Перечень информационных ресурсов, размещенных в мобильном устройстве и подлежащих защите, приведен в таблице 6.1.

Таблица 6.1. Защищаемые информационные ресурсы [4]

Тип информационного ресурса	Описание информационного ресурса
Данные о контактах	Номер телефона. ФИО, электронная почта, группа контакта, дата рождения контакта, почтовый адрес, сведения о работе, псевдоним, веб-сайт и отношения
Данные о совершенных звонках	Журнал входящих/исходящих/пропущенных вызовов
SMS-данные	Журнал сообщений, содержимое входящих, исходящих SMS и MMS, а также черновики сообщений
Данные из органайзера	Встречи, заметки, списки дел, дни рождения; данные из стандартных приложений, таких как будильник, календарь и др.
Данные аккаунтов электронной почты	Контакты, электронные адреса, вложенные сообщения и файлы
Данные из браузеров	Временные файлы, список посещенных страниц, закладки браузеров
Журнал трафика и сессий	Пакеты, переданные через GPRS, LTE, Wi-Fi и др.
Медиафайлы	Файлы (фото, видео-, аудиофайлы, документы), хранящиеся в памяти телефона
Голосовые данные	Речь при совершении звонков, использование голосовых команд в интернет-сервисах
Данные телефона	Уникальный номер телефона, фирма-производитель, модель устройства и др.
Данные SIM-карты	Номер SIM-карты; юридическое лицо, обслуживающее мобильные станции; тарифный план и др.
Геолокационные данные	Текущее местонахождение, маршрут движения и др.
Данные с медиаустройств	Данные, полученные с динамика, микрофона, фронтальной и основной камеры и др.
Данные с карты памяти	Файлы (фото, видео-, аудиофайлы, документы), хранящиеся в карте памяти
Данные по использованию мобильных приложений	Использование социальных сетей, игровая статистика и др.

Так как СИП у каждого приложения своя и загружается в память устройства в процессе запуска приложения, то функционируя на основе разрешений, предоставляемых приложению ядром ОС, СИП потенциально может получить доступ к любому из вышеперечисленных типов информационных ресурсов. В том случае, если приложению в результате осуществления вредоносной атаки предоставлены привилегии административного доступа (root), такая угроза переходит из разряда потенциальных в разряд реально осуществимых.

К защищаемым информационным процессам, связанным с функционированием СИП можно отнести:

1) процесс загрузки СИП;

2) процесс установки приложения;

3) процесс подготовки приложения к выполнению на устройстве (оптимизация и/или компиляция);

4) процесс выполнения приложений на конкретной платформе устройства (интерпретация или выполнение машинного кода).

Рассмотрим проблемы информационной безопасности указанных информационных процессов.

Процесс загрузки СИП app_process функционирует под правами администратора. Для того что бы иметь возможность влиять на данный процесс, злоумышленник должен получить данные права, используя уязвимости отдельных компонентов ОС, в т.ч. и непосредственно загрузчика.

На первом этапе злоумышленнику требуется получить либо физический доступ к устройству, либо осуществить удаленную атаку, используя методы социальной инженерии или методы сетевых атак. Второй этап предполагает использование уязвимостей компонентов ОС с целью получения повышенных привилегий. Основные способы реализации атак на отдельные компоненты ОС с целью получения root-доступа подробно описаны в специальной литературе [5, 6, 7].

Ряд существовавших ранее уязвимостей в app_process, позволяли злоумышленнику форсировать применение механизма fork и осуществить большое количеством запросов на создание фиктивных экземпляров среды исполнения, тем самым израсходовав все ресурсы памяти устройства [8]. Поскольку среда исполнения не прерывала работу в этой ситуации, то код вредоносного приложения выполнялся с тем же UID, что и app_process, то есть с административными правами [9].

При наличии прав администратора злоумышленник может переопределить процесс app_process в памяти устройства и внести любые изменения в СИП каждого запускаемого приложения [5].

Так троян Backdoor.AndroidOS.Triada использует процесс app_process для внедрения своего кода в код всех приложений, установленных на устройстве [10]. Для этих целей внутри адресного пространства своего процесса вредоносная программа осуществляет модификацию образа файла framework.odex, осуществляя тестирование возможности модификации в памяти устройства процесса app_process. Затем, выявив PID процесса app_process, в который загружен образ «framework.odex» вредоносная программа проводит его модификацию с использованием системной утилиты ptrace, используя для этого ранее выявленные особенности размещения в памяти framework.odex. Таким образом, вредоносный код оказывается внедренным в память любого запускаемого на мобильном устройстве приложения.

Атаке может также подвергаться не только сам процесс app_process, но и файлы его загрузочных образов. Так как файлы, содержащие скомпилированные образы предварительно загружаемых классов и находящиеся в каталоге «/data/dalvik-cache/{platform}/», загружаются в память только при запуске процесса app_process, то для реализации атаки необходимо перезаписать данные файлы и добиться перезапуска процесса app_process, например, осуществить атаку на отказ в обслуживании на родительский процесс system_server [11].

Поскольку платформа Android поддерживает подпись приложений с использованием самозаверяющих сертификатов, злоумышленник может легко внести изменения в содержание файлов, входящих в APK, используя методы обратного инжиниринга. Именно поэтому, переупаковка – одна из наиболее важных и распространенных проблем безопасности СИП [12, 13].

Злоумышленник может внедрить вредоносный код, чтобы исправить существующие функции или даже добавить новые функции, как в двоичные файлы приложения (.so), так и в файлы, содержащие байт-код (.dex).

Методы переупаковки, которые можно использовать на платформе Android, позволяют замаскировать вредоносный код как обычное приложение. Для скрытия вредоносного кода и обхода механизмов

анализа информационной безопасности злоумышленник может использовать обфускацию кода, динамическую загрузку зашифрованных модулей с помощью методов класса DexClassLoader, перемешивание кода и использование нативных библиотек [5, 13].

Провести различие между переупакованным вредоносным и обычным приложением практически невозможно, потому что переупакованное приложение обычно работает так же, как и легитимное. Примерами вредоносного программного обеспечения, основанного на переупаковке APK, являются трояны Geimini, InfectionAds и многие другие.

При обновлении приложения СИП осуществляет проверку подписи исходной и новой версии приложения, продолжая установку только в том случае, если они совпадают. При этом разрешения исходного приложения вне зависимости от того является оно пользовательским или системным полностью наследуются его новой версией. Такая особенность СИП является привлекательным вектором атаки для злоумышленника, не имеющего root-доступа к устройству.

Структура APK файла как разновидности ZIP формата состоит из трех блоков: записи файла, основной каталог, завершающий метки. При этом записи файла могут находиться в произвольном порядке.

Такая организация файла позволяет злоумышленникам изменять код в приложениях, не затрагивая их подписи. Для этого в начало файла APK необходимо поместить вредоносный файл DEX.

В результате СИП из-за кода DEX в заголовке файла APK распознает его как файл DEX и приступает к его извлечению и установке. При этом СИП исходное содержимое APK, которое следует за файлом DEX игнорирует. И одновременно с этим другой процесс СИП рассматривает файл как действительный файл APK и получает доступ к его остальным элементам: библиотекам, ресурсам, подписи и к манифесту.

Таким образом, СИП обладала серьёзной уязвимостью, связанной с возможностью неправильной интерпретацией файлов DEX / APK. Указанная особенность предопределила название данной уязвимости в честь римского бога двойственности – Януса (CVE-2017-13156).

В [14] описана еще одна уязвимость СИП, которая приводила к тому, что злоумышленник, не обладая root-доступом, мог запретить устанавливать на устройство легальные приложения.

СИП при установке приложения пользователя в первую очередь создаёт учетную запись пользователя, группу и каталог личных данных для установки приложения, а только затем проводит верификацию файлов приложения. В результате, если злоумышленник имел возможность предоставить под именем легального приложения код DEX с преднамеренными ошибками, то СИП прерывал установку. При этом ранее созданная инфраструктура приложения в ОС не удалялась. Это приводило к тому, что при установке легального приложения возникал конфликт и пользователь не мог его установить без получения прав суперпользователя.

Для атак на СИП могут использоваться уязвимости пользовательских приложений и их внешних компонентов, обусловленные уровнем качества разработки и тестирования мобильных продуктов. К этой категории уязвимостей относится известный список Mobile OWASP-10: некорректное использование платформы, небезопасное хранение данных, небезопасная передача данных, небезопасная аутентификация, слабая криптографическая стойкость, небезопасная авторизация, низкое качество клиентского кода, модификация кода, обратная разработка, скрытая функциональность. Многие из приведенных (см. приложение) и им подобных уязвимостей рассмотрены в следующих работах [15,16, 17].

Также следует отметить, что большинство критических и опасных уязвимостей обычно находятся в мобильном бэкэнде (например, веб-сервисах и API внешних библиотек), а не внутри мобильного [16].

В процессе подготовки к исполнению и исполнения приложения (в зависимости от применяемой на устройстве комбинации JIT/OAT) злоумышленник может атаковать файлы подготовленные СИП для интерпретации и непосредственного выполнения (base.vdex, base.odex, base.art).

В работе [18] было впервые продемонстрирована возможность изменения поведения приложений и СИП за счет замены сгенерированного ART файла измененным файлом OAT. Для этих целей на устройстве вручную запускалась утилита dex2oat, которая проводила

манипуляции с файлами. Основной целью такой атаки называлось скрытие получения root-доступа к ОС мобильного устройства.

В работе [19] продемонстрирована возможность осуществления атаки на файл приложения base.art и base.odex. Злоумышленник может воспользоваться процессом загрузки данных файлов в память устройства для того что бы заменить их содержание на вредоносное (динамический файл base.art в виду сложности подделки предлагается удалить, а статический файл base.odex – изменить). Указанные манипуляции не требуют переустановки файла APK .

Так как СИП проверяет легитимность файла base.odex перед загрузкой, то сначала осуществляется проверка CRC32 для каждого class.dex в APK (base.apk и сравнение их с контрольной суммой в OatDexFile один за другим (это гарантирует, что файл base.odex изначально сформрован из APK). Затем, если проверка прошла, image_file_location_oat_checksum в OATHeader base.odex извлекается для сравнения с adler32_checksum в OATHeader в boot.oat на устройстве (это гарантирует, что файл base.odex генерируется на устройстве).

Приведенный процесс проверки уязвим, поскольку контрольные суммы могут быть заменены правильными, чтобы выполнить проверку, если злоумышленник хорошо разбирается в структуре OAT.

Также в работах [19, 20] указывается на то, что внедрение вредоносного кода может быть проведено на нескольких уровнях вызова метода JAVA, который размещен в коде приложения или в коде образов СИП): на уровне описания классов (подделка vtable), на уровне внутренней структуры класса (подделка точки входа ArtMethod) и на уровне машинного кода (подделка нативного кода метода).

Все эти атаки требуют привилегий root для доступа к папке изолированной программной среды приложения, процессу приложения и загрузочным образам в папке с файлами СИП.

Атака на образы СИП и выполняемые файлы приложения реализована вредоносной программой Backdoor.AndroidOS.Triada, которая была описана ранее.

При использовании режима интерпретации верифицированный и оптимизированный байт-код приложения (.vdex) выполняется с использованием ВМ СИП. При этом сегмент кода приложения, который в данный момент выполняется, СИП компилирует в физический машинный код.

Таким образом, JIT-компилятор является одним из немногих типов программ, которые можно запускать в среде с неисполняемыми данными. Это приводит к вероятности осуществления атак, связанных с выполнением интерпретатором не кода, а данных (атаки, использующие возвратно-ориентированное программирование, ROP) [3].

Данный тип атак получил широкое распространение на платформе Android. Примером может служить эксплойт Stagefright. Целями атак является также код, размещенный в файлах формата .elf (например, .odex, .so).

Функция JIT позволяет запускать динамический код во время выполнения. Вместо того, чтобы интерпретировать виртуальный код (байт-код), компилятор компилирует весь код (или его часто используемые части) в машинно-зависимый код.

LLVM, используемая при компиляции файлов приложения, также содержит ошибки, которые потенциально могут быть использованы злоумышленником для нарушения информационной безопасности СИП.

Кроме того, используемые компиляторы тесно связаны с микрокодом процессора платформы и ошибки в логике работы современных процессоров мобильных устройств также могут использоваться для совершения атак. Например, уязвимости CPU Spectre и Meltdown эксплуатируют особенности спекулятивного выполнения современных процессоров.

Рассмотренные способы реализации угроз с указанием уровня доступа, требуемых для их осуществления приведены в таблице 6.2.

В СИП выявляются регулярно ошибки, которые приводят к тому, что злоумышленник может:

1) вызывать зависание приложения при его выполнении (CVE-2017-0780);

2) получить доступ к данным, открытым только для установленных на устройстве приложений, которые обладают необходимыми разрешениями (CVE-2017-13309);

Таблица 6.2 Способы реализации угроз на СИП

Способы реализации угроз	Требуемый уровень доступа
Процессы загрузки СИП:	
- форсирование применение механизма fork;	root
- изменение загрузочных образов СИП (framework.odex, boot.odex, boot.vdex и т.д.);	root
- изменение загрузочных образов СИП в памяти устройства	root
Процессы СИП, обеспечивающие установку и обновление приложений:	
- установка файла apk, содержащего вредоносный код;	-
- изменение содержимого и переупаковка легального файла base.apk;	-
- изменение содержимого легального файла base.apk после распаковки;	root
- обход подписи приложения;	-
- запрет установки легальных приложений;	-
- перезапись файла platform.xml.	root
Процессы СИП, обеспечивающие исполнение приложений:	
- замена или внедрение вредоносного кода в исполняемые файлы приложения (base.odex, base.vdex, base.art);	root
- форсирование использования ресурсов устройства;	-
- осуществление сговора, в т.ч. за счет использования уязвимостей или избыточных разрешений легальных приложений	-

3) обойти требования к взаимодействию с пользователем и получить доступ к дополнительным разрешениям (CVE-2017-13274, CVE-2017-13176);

4) выполнять произвольный код в контексте непривилегированного процесса (так, уязвимость CVE-2016-6703, основана на ошибке проверки размера данных массива исполняемого приложения в определенном контексте выполнения, что позволяет злоумышленнику выполнять произвольный код в контексте непривилегированного процесса с помощью специально созданных данных).

Таким образом, основные направления реализации злоумышленником угроз СИП предполагают:

1) создание программных средств, выполняющих обращение к защищаемым ресурсам в обход механизмов защиты;

2) осуществление модификации механизмов защиты, позволяющая реализовать угрозы ИБ;

3) внедрение в программные средства загрузки и работы СИП программных механизмов, нарушающих предполагаемую структуру и функции СИП.

6.2. Защитные механизмы среды исполнения

В ОС Android реализована многоуровневая система безопасности, которая по замыслу разработчиков должна обеспечить гибкость, необходимую для открытой платформы и защиту для всех пользователей.

ОС Android реализует принцип предоставления минимальных прав: каждое приложение расценивается как потенциально опасное и запускается в изолированном окружении (песочнице).

Для изоляции процессов в ОС Android используется подход, который является традиционным для Linux: каждое приложение запускается в собственном процессе, который имеет собственную СИП. Данный подход к изоляции процессов осуществляется через пользовательский идентификатор системы Linux (UID), который назначается всем без исключения приложениям в т.ч. системным.

Для низкоуровневых частей системы (root, system_server и т.д.) предопределены стандартные UID-идентификаторы. Приложения пользователя получают его из свободного диапазона при первой установке.

Однако ОС Android предоставляет приложению несколько возможностей для получения доступа к данным других приложений.:

1) если два приложения подписаны одним разработчиком, то они могут запускаться в одной СИП и иметь общий доступ к файлам;

2) приложение может запросить разрешение на доступ к тем данным, которое приложение или непосредственно ОС может предоставить.

В целом модель изоляции процессов в ОС Android оценивается эффективной, однако отмечается, что при получении злоумышленником административных привилегий она становится бесполезной, так как возможности root пользователя ничем не ограничены. Поэтому одним из защитных механизмов на мобильных устройствах является невозможность пользователю эксплуатировать его, используя права суперпользователя.

Разрешение – четко определенная возможность, предоставляемая его приложению во время его исполнения. Различают следующие виды разрешений: нормальные (обычные) и опасные. Первые не несут с точки зрения разработчиков ОС прямой угрозы конфиденциальности (доступ к сети, доступ к управлению оповещениями и т.д.). Вторые потенциально могут быть использованы злоумышленником для нанесения вреда пользователю (доступ к камере, звонкам, контактам, смс и т.д.

Разрешения непосредственно связаны с UID-идентификатором процесса, в котором выполняется приложение и как следствие барьер безопасности в форме разрешения всегда оказывается на его границе.

Для получения доступа к определенной низкоуровневой функциональности ОС или других приложений UID-идентификатор процесса добавляется в группу, для которой определена возможность реализации тех или иных защищенных разрешением функций. Такое сопоставление осуществляется через файл platform.xml, который размещается в «/system /etc/platform-xml/».

В работах [21, 22] отмечается неэффективность действующей системы управления разрешениями, причинами которой являются:

1) непонимание пользователями сущности используемой модели разрешений;

2) принятие пользователями риска в ущерб информационной безопасности;

3) её бесполезность при получении прав суперпользователя или возможности перезаписи файла platform.xml.

Изоляция процессов друг от друга не предполагает отсутствие взаимодействия между ними. Такое взаимодействие в ОС Android основано на использовании механизма межпроцессного взаимодействия (Binder IPC), который основан на клиент-серверной модели: один процесс может вызывать подпрограмму в другом процессе, используя Binder, чтобы идентифицировать метод для вызова и передачи аргументов между процессами. Binder IPC является специфичным для ОС Android (стандартный механизм Linux SysV IPC не используется).

Эффективность и безопасность Binder IPC неоднократно ставилась под сомнение. Например, в исследовании «Man in the Binder» [11] было выявлено, что любая информация, которая передаётся с исполь-

зованием Binder IPC может быть перехвачена. Описание основных векторов атак на механизм межпроцессного взаимодействия в ОС Android приведена в работе [24].

Файлы APK, которые предназначены для распространения приложения, могут быть подписаны личным закрытым ключом разработчика, что предполагает определенную защиту от подделки приложения. ОС, расположенная на устройстве, проверяет подпись приложения на основе открытого сертификата, который поставляется с установочным пакетом. Данный механизм также не является идеальным и не однократно подвергался вредоносным атакам, ряд которых были рассмотрены ранее.

Многопользовательский режим предоставляет возможность использовать устройство нескольким пользователями и при этом у каждого присутствует свое отведенное пространство, ресурсы, которые недоступны остальным.

Применение SELinux как части ядра Linux в ОС Android позволило реализовать принудительный контроль доступа на основе специально определенных политик на различных уровнях системных вызовов. Именно на основе SELinux построена система разрешения как варианта одной из его функций - мандатного управления доступом.

Изоляции процессов в ОС Android дополняется механизмом фильтрации системных вызовов для приложений и их СИП: ограничение передаваемых параметров или полная блокировка ряда системных вызовов, которые могут быть использованы нарушителями информационной безопасности.

Данная технология является стандартной для ядра Linux и используется в ОС Android достаточно давно и, начиная с версии 8 внедрена в app_process, ответственный за порождение процессов всех приложений.

ОС Android также обеспечивает СИП защитными технологиями DEP & ASLR, что препятствует эксплуатации уязвимостей как в приложениях, так и в самой системе.

ASLR – специальный защитный механизм, который предполагает рандомизацию расположения адресного пространства с целью предотвращения выполнения кода. Достигается это за счет того, что осуществляется случайное назначение базовых точек различных областей памяти (например, стека, кучи, библиотек и т. д.).

DEP предназначен для предотвращения атак, основанных на методах возвратно-ориентированного программирования и связанных с выполнением данных (определенных секторов памяти, которые предназначены не для хранения операций, а данных).

Системные обновления предполагают получение регулярных обновлений ОС Android, которые должны устранить выявляемые уязвимости в системе [24]. Однако, существует определенная проблема с обновлением, суть которой заключается в том, что многие компании партнеры Google – производители мобильных устройств не заинтересованы в своевременном обновлении свои продуктов, несмотря на ежемесячные пакеты исправлений, которые выпускает Google. По приблизительным оценкам, 68% всех устройств Android не имеет возможности получить обновления [25].

В ОС Android предусмотрен специальный механизм безопасности (начиная с версии 4.4 применяется в уведомительном режиме, а с версии 7 – в принудительном), который проверяет подлинность и целостность важнейших компонентов системы: первичного загрузчика, вторичного загрузчика, ядра Linux и системы в целом.

Для проверки используются следующие подходы:

1) проверка цифровых подписей (загрузчики);

2) проверка контрольной суммы (ядро и система в целом).

Механизм контроля целостности при загрузке является многоуровневым: каждый компонент в процессе загрузки проверяется целостность и подлинность следующего этапа перед началом его выполнения, в т.ч. проверяется возможный откат в версии.

Файлы загрузочных образов и исполняемые файлы приложений содержат контрольные суммы, которые проверяются перед использованием. ART проверит два вида контрольных сумм. Одним из них является контрольная сумма в OATHeader, которая должна быть равна контрольной сумме boot.oat платформы Android, чтобы убедиться, что файл кеша создан на устройстве. Другой находится в структурах OatDexFile. Эти контрольные суммы должны совпадать с контрольными суммами, рассчитанными с помощью соответствующих классов * .dex в base.apk. Однако, ряд авторов неоднократно указывали на простоту обхода данного защитного механизма за счет их подделки [18, 19].

В ОС Android реализован собственный механизм защиты от вредоносного ПО – Google Play Protect, который реализует защиту на нескольких уровнях, осуществляя сканирование приложений: в момент размещений в Play Market; в момент предшествующий установки приложения на устройства пользователя, периодически в течение времени использования приложения пользователем. Однако эффективность данного механизма также подвергается сомнению [26]. Результаты исследования основных механизмов защиты СИП приведены в таблице 6.3.

Таблица 6.3. Основные механизмы защиты СИП

Защищаемые процессы	Механизм защиты
Загрузка СИП	Запрет владельцу работы под правами суперпользователя Защита от рандомизации разметки адресного пространства (ASLR) Контроль при загрузке компонентов ОС Контроль целостности загрузочных образов Google SafetyNet Attestation
Установка и обновление приложения	Песочница Валидация подписи приложения Антивирусное программное обеспечение Статический и динамический анализ кода и разрешений Google Play Protect
Исполнение приложения	Песочница Разрешения Binder IPC Seccomp (блокировка/ограничение опасных системных вызовов) Механизм предотвращения выполнения данных (DEP) Защита от рандомизации разметки адресного пространства (ASLR) Контроль целостности исполняемых файлов и их загрузочных образов

6.3. Подходы к проведению аудита безопасности среды исполнения приложений

Аудит информационной безопасности – независимая оценка текущего состояния системы информационной безопасности, устанавливающая уровень ее соответствия определенным критериям (требованиям и ограничениям), и предоставление результатов в виде рекомендаций [27].

Аудит безопасности включает в себя распознавание, запись, хранение и анализ информации, связанной с действиями, относящимися к безопасности. Записи аудита, получаемые в результате, могут быть проанализированы, чтобы определить, какие действия, относящиеся к безопасности, происходили, и кто из пользователей за них отвечает [28]. Основные элементы аудита информационной безопасности приведены на рис. 6.1.

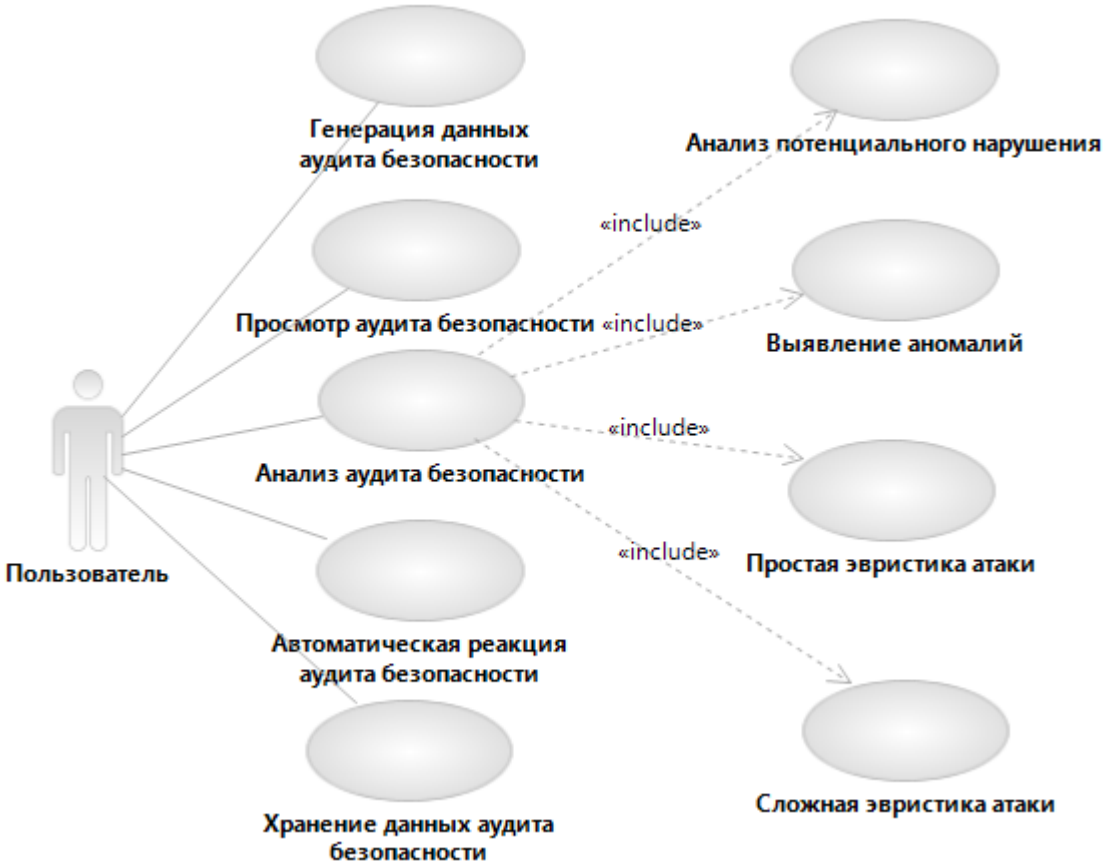


Рис. 6.1. Элементы аудита информационной безопасности

Генерация данных аудита безопасности предполагает регистрацию возникновения событий, относящихся к безопасности.

Анализ аудита безопасности предполагает оценку показателей функционирования системы и данных аудита в целях поиска возможных или реальных нарушений безопасности. Данный компонент включает:

1) анализ потенциального нарушения (используется для определения совокупности событий, потенциально подвергаемых аудиту, появление которых (каждого отдельно или в совокупности) указывает на потенциальные нарушения выполнения);

2) выявление аномалии, основанное на профиле (шаблоне предыстории использования);

3) простая эвристика атаки (обнаружение возникновения характерных событий, которые свидетельствуют о значительной угрозе для информационной безопасности);

4) сложная эвристика атаки (обнаружение характерного события или последовательности событий, которые свидетельствуют о значительной угрозе для информационной безопасности).

Анализ аудита безопасности может осуществляться как в режиме реального времени, так и при анализе данных аудита в пакетном режиме.

Анализ аудита безопасности может использоваться для поддержки как обнаружения проникновения (подача сигнала), так и автоматической реакции на потенциальное нарушение безопасности (прерывание процесса с выявленным нарушением, прекращение обслуживания, блокирование или закрытие учетных данных пользователя).

Под аудитом безопасности СИП понимается процесс сбора и анализа информации об СИП, о её компонентах и системных процессах с ней связанных с целью качественной или количественной оценки уровня её защищенности от атак злоумышленников [29].

Как было отмечено ранее, в загружающем СИП системном процессе app_process неоднократно выявлялись уязвимости, которые потенциально могли быть или были реализованы на практике злоумышленниками.

Существующие механизмы защиты ОС Android позволяют снизить к минимуму риск атак на системный процесс загрузки СИП, но не исключить выявление новых уязвимостей.

Поэтому одним из подходов к аудиту СИП широко используемом в антивирусном программном обеспечении является поиск уязвимостей на основе информации об установленной конфигурации ОС Android и её сравнение с эталонными данными [29]. Для этого поставщики антивирусного программного обеспечения либо самостоятельно создают или используют существующие базы данных уязвимостей для различных версий и выпусков ОС, например, база уязвимостей ФСТЭК. Данный подход не требует root-доступа к устройству, однако при взломе злоумышленником мобильного устройства не является эффективным, так как при получении информации о характеристиках ОС Android она может быть изменена.

Для аудита файлов, которые содержат образы предварительно загружаемых классов и находящиеся в каталоге «/data/dalvik-cache/{platform}/», необходимо иметь root-доступ. При его наличии может быть осуществлена побитовое сравнение файлов (проверка контрольных сумм, сгенерированная ОС может быть подделана и поэтому является не эффективной).

При отсутствии root-доступа может быть проверен образ этих файлов в памяти СИП конкретного приложения. Для этих целей может быть использован существующий инструментарий внедрения кода в изолированную часть памяти СИП и сравнение данного образа с эталонным [19].

Как отмечалось ранее, исходной точкой атаки на СИП является вредоносное программное обеспечение, содержащиеся в файле APK приложения. Поэтому одним из подходов к аудиту СИП заключается в обнаружении переупаковки приложения.

Для этих целей APK-файл, который загружается в песочницу приложения (каталог: «/data/app/{AppName}/base.apk») и оттуда устанавливается, сравнивается с эталоном, хранящегося в защищенном месте (например, в Google Play или в публичном, или частном депозитарии).

Для обнаружения переупаковки используются следующие методы: обнаружение на основе полного побитового сравнения, обнаружение на основе нечеткого хеширования [31, 32], обнаружение на основе графа программных зависимостей [33, 34], обнаружение на основе хеширования функций [35], обнаружение на основе динамически

внедренного кода [36], обнаружение на основе использования статических и динамических водяных знаков [37, 38, 39].

Для обнаружения вредоносных компонентов в содержимом APK-файлов и предотвращения исполнения вредоносного кода используется различные методы, основанные на идентификации злоупотреблений или аномалий.

К методам обнаружения злоупотреблений относят:

1) методы предотвращения утечки данных;

2) методы анализа разрешений;

3) сигнатурные методы;

4) комбинированные методы.

Методы предотвращения утечки данных позволяют выявить утечку конфиденциальных сведений. Идентификация таких данных осуществляется техниками проверки контента и контекста (например, предварительно определенными ключевыми словами, шаблонами, регулярными выражениями, размещением в конфиденциальной информации специальных меток и т.д.) [8, 40]. Недостатком данной группы методов относят их ограниченность теме классами вредоносного кода, возможности которого рассматриваются.

Методы, основанные на анализе разрешений, используют только информацию из манифеста приложения для того, чтобы оценить опасность его использования. Как правило, полученный список разрешений подвергаемого аудиту приложения сопоставляется с множеством предопределенных правил, которые представляют собой опасные конфигурации запрашиваемых привилегий [41, 42].

В некоторых реализациях систем аудита используются техники обработки естественных языков для анализа описаний приложения и его разрешений. Затем обработанные описания сопоставляются с семантическими графами, представляющими привилегии, с целью выявления несоответствия запрашиваемых разрешений описанию [43].

Данный метод даёт большое число ложноположительных и ложноотрицательных результатов: существует множество примеров, когда опасные вредоносные приложения не требовали предоставления им каких-либо разрешений. Для повышения эффективности данного метода предлагается его дополнять анализом API-вызовов [41].

Сигнатурные методы основаны на сравнении код приложения с заранее выявленными некоторыми характерными идентифицирующими свойствами вредоносного кода. Данный метод широко используется в антивирусном программном обеспечении, но не применим для выявления новых и неизвестных вредоносных программ.

Методы, основанные на выявлении аномалий, предполагают разработку моделей легитимной функциональности [44, 45]. Однако такой подход является затруднительным в реализации, так как требует учета множества шаблонов нормального состояния и поведения приложения.

При выявлении злоупотреблений и аномалий широко используются статистические методы, кластерный анализ, методы машинного обучения (нейронные сети, иммунные сети и т.д.) [46] и облачные вычисления, а также методы автоматического анализа: статический, динамический и смешанный (гибридный) [7, 14, 47].

Статистический метод основан на применении программных средств, осуществляющих поиск уязвимостей и угроз по шаблонам или эвристическим правилам. Способ позволяет обрабатывать программный код целиком за короткое время. Недостаток – подходит исключительно для поиска достаточно формализуемых уязвимостей. Отдельной задачей является разработка самих шаблонов для поиска, осуществляемая в основном вручную [48, 49].

Альтернативой статического метода является динамический, представляющий собой анализ не программного кода, а результатов его выполнения (в эмуляторе или реальной среде выполнения). В этом случае уязвимости обнаруживаются не напрямую, а по их влиянию на «нормальное» выполнение программного кода: сетевой трафик, взаимодействие с файловой системой, память и т.д. [50].

Данный метод требует больших вычислительных ресурсов и для полной автоматизации его применение затруднительно. Ускорение процесса осуществимо путем оптимальной выборки набора данных. Способ носит название – «фаззинг тестирование» (от англ. fuzzing или fuzz testing). Результатом работы, как правило, является отчет о программных исключениях кода, вызванных определенными условиями.

Техники, используемые в статическом анализе среды выполнения ОС Android:

1) извлечение метаинформации из манифеста приложения и предоставляющие информацию о запрашиваемых разрешениях, компонентах Activity, сервисах и зарегистрированных Broadcast receivers;

2) извлечение информации из пакетных установочных файлов;

3) модификация существующего байт-кода приложения (например, добавление трассирования функциональности в существующие приложения) и т.д.

Техники, используемые в динамическом анализе среды выполнения ОС Android [50]:

1) отслеживание помеченных данных (отслеживание потенциальных утечек конфиденциальной информации);

2) мониторинг системных вызовов (запись системных вызовов с помощью инструментирования ВМ, strace или модуля ядра);

3) трассировка методов (функций) (отслеживание вызовов Java-методов приложений в ВМ вызовов функций в машинных кодах через JNI, системных вызовов и прерывания).

Еще одним подходом к аудиту СИП является анализ привилегий, которые необходимо приложению для работы. Методы, которые реализуют данный подход, используют только информацию из манифеста приложения для того, чтобы оценить опасность его использования, в частности:

1) сопоставляют конфигурацию безопасности анализируемого приложения с множеством предопределенных правил [42]. Эти правила представляют из себя опасные конфигурации запрашиваемых привилегий и строк действий (action string), объявленных в манифесте приложения. Если приложение удовлетворяет хотя бы одному правилу, оно считается опасным;

2) проводят проверку соответствия запрашиваемых привилегий описанию приложения. Используются техники обработки естественных языков для анализа описаний. Затем обработанные описания сопоставляются с семантическими графами, представляющими привилегии [2];

3) проводят сопоставление соответствия запрашиваемых привилегий с сигнатурами привилегий ранее известных вредоносных приложений. Как дополнение к данному подходу используется статический анализ с целью выявления характерных для вредоносных программ API-вызовов [46, 51].

Аудит в процессе исполнения приложений предполагает: 1) мониторинг и анализ способности ВМ СИП выполнять только программный код, который соответствует предписанному набору её возможностей (например, ВМ может разрешить коду доступ только к определенному набору функций или данных); 2) мониторинг и анализ использования СИП ресурсов мобильного устройства; 3) мониторинг и анализ использования и хранения исполняемых файлов приложения и его данных. Для указанных целей используют как встроенные, так и сторонние инструменты.

С целью обнаружения и диагностирования ошибок, связанных с неправильным использованием памяти и потенциально опасным неопределенным поведением СИП применяют LLVM Sanitizers, который включает ряд компонентом мониторинга, например, AddressSanitizer, UndefinedBehaviorSanitizer. Данные инструментарий, как правило, используется для мониторинга работы скомпилированного системой или разработчиком машинного кода. Для использования данных инструментов требуется root-доступ.

Инструмент AddressSanitizer (или ASan), используя инструментарий компилятора и библиотеки времени выполнения, позволяет обнаруживать ошибки повреждения памяти, такие как переполнение буфера или доступ к висячему указателю (use-after-free). Для данных целей также можно использовать в настоящий момент не поддерживаемый ОС Android Valgrind.

Инструмент UndefinedBehaviorSanitizer (или UBSan), изменяя программу во время компиляции, определяет различные виды неопределенного поведения во время выполнения программы (например: использование смещенного или нулевого указателя, переполнение целого числа со знаком, преобразование в, из или между типами с плавающей запятой, которые будут переполнять назначение.

Для мониторинга и анализ использования СИП ресурсов мобильного устройства используется инструментарий Android Debug Bridge (adb): procrank, dumpsys meminfo, vmstat, который широко используется при разработке приложений.

Procrank показывает краткий обзор использования памяти системными процессами: размер виртуальной памяти (Vss); размер доступной памяти (Rss); размер собственной памяти процесса (Uss); размер памяти, совместно используемый с другими процессами (Pss).

Dumpsys позволяет получить общую статистику по использованию различных системных ресурсов в определенный момент времени, в т.ч. распределение памяти СИП различных приложений: параметры .so mmap, .dex mmap, .oat mmap, .art mmap и другие.

Vmstat позволяет получить информацию о процессах, памяти, подкачке страниц, блочном вводе-выводе, прерываниях и активности процессора.

Для мониторинга взаимодействия между процессами и ядром Linux, которое включает системные вызовы, доставку сигналов и изменения состояния процессов может быть использована утилита strace. Начиная с версии ОС Android 8 для возможности использования данного инструмента недостаточно обладать root-доступом, но и необходимо отключить SELinux и перезапустить среду выполнения, для того чтобы удалить фильтр seccomp, который в противном случае не позволит запустить strace.

Для мониторинга и анализа использования исполняемых и интерпретируемых файлов приложения (.oat, .vdex) широко используется подход, основанный на внедрении кода в память процесса приложения. Ряд инструментов, реализующих данный подход, имеют открытый исходный код и доступны на github [52, 53].

Авторы этих инструментов используют разные точки подключения для достижения цели отладки приложения. Так ARTDroid [52] предполагает подмену ссылки на целевой метод в виртуальной таблице исполняемого файла (vtable), которая используется для поиска и запуска виртуальных методов, в т.ч. связанных с API-интерфейсами ОС Android, а xHook для этих целей использует глобальную таблицу смещений (PLT) в памяти текущего процесса. ProbeDroid использует подмену адреса машинного кода для вызова целевого кода внутри исполняемого метода.

ARTist [53] основан на подмене первой инструкции в полном исходном коде модифицируемого метода Java в файле, который содержит не измененный, но возможно оптимизированный, байт-код DEX (.vdex).

Ряд авторов отмечают, что внедрение кода с одинаковым успехом может быть использовано как для мониторинга работы СИП, так и для атаки на неё [19]. Результат анализа основных направлений аудита СИП представлен на рис. 6.4.

Таблица 6.4. Анализ основных направлений аудита ИБ СИП

Основные направления аудита	Требуемый уровень доступа
Процессы загрузки СИП: - проверка на наличие root-доступа к ОС; - поиск и оценка защищенности СИП на основе списка известных уязвимостей; - проверка целостности временных загрузочных образов СИП; - проверка целостности временных загрузочных образов СИП в памяти устройства.	 - - root root
Процессы СИП, обеспечивающие установку и обновление приложений: - выявление дефектов безопасности в app.apk; - выявление и оценка возможных злоупотреблений в app.apk (утечек, избыточных разрешений, вредоносного кода); - обнаружение переупаковки файла base.apk; - проверка целостности файлов .dex , .so, packages.xml, platform.xml	 - - - root
Процессы СИП, обеспечивающие исполнение приложений: - проверка способности ВМ СИП выполнять только программный код, который соответствует предписанному набору её возможностей; - мониторинг и анализ использования СИП ресурсов мобильного устройства; - выявление аномалий в поведении приложения; - выявление внедрения вредоносного кода в процесс приложения; - контроль целостности загрузочных образов приложения; - выявления сговора между приложениями.	 root - root root root -

Как следует из анализа, реализовать большинство подходов к аудиту СИП можно только обладая правами суперпользователя, что не имеет высокой практической ценности.

Представляется, что наиболее эффективными и несложными в реализации могут быть следующие направления: проверка на наличие root-доступа к ОС; поиск и оценка защищенности СИП на основе

списка известных уязвимостей; выявление и оценка возможных злоупотреблений в пакетном файле; контроль целостности загрузочных образов приложения.

Вопросы для обсуждения

1. Факторы, влияющие на информационную безопасность мобильных приложений.
2. Основные угрозы информационной безопасности мобильных платформ.
3. Источники угроз информационной безопасности мобильных платформ.
4. Нарушители информационной безопасности мобильных платформ
5. Защищаемые информационные ресурсы мобильных устройств.
6. Защищаемые информационные процессы, связанным с функционированием среды исполнения приложений операционной системы Android.
7. Анализ уязвимостей информационных процессов, связанных с функционированием среды исполнения приложений операционной системы Android.
8. Способы реализации угроз на процессы, связанные с функционированием среды исполнения приложений операционной системы Android.
9. Архитектура система информационной безопасности в операционной системе Android.
10. Система управления разрешениями в операционной системе Android.
11. Система проверки подлинности и целостности важнейших компонентов в операционной системе Android.
12. Основные механизмы защиты среды исполнения приложений операционной системы Android.
13. Элементы аудита информационной безопасности.
14. Методы обнаружения злоупотреблений при проведении аудита информационной безопасности.
15. Техники, используемые в статическом анализе среды исполнения приложений операционной системы Android.

16. Техники, используемые в динамическом анализе среды исполнения приложений операционной системы Android.

17. Анализ основных направлений аудита среды исполнения приложений операционной системы Android.

18. Проверка на наличие root-доступа к операционной системе Android.

19. Поиск и оценка защищенности среды исполнения приложений на основе списка известных уязвимостей.

20. Выявление и оценка возможных злоупотреблений в пакетном файле мобильного приложения.

21. Контроль целостности загрузочных образов мобильного приложения.

Практические задания

Задание 6.1.

На основании решения заданий из главы 4 выполните описание защищаемых информационных ресурсов проектируемого мобильного приложения

Задание 6.2.

На основании решения заданий из главы 4 и 6.1 выполните описание необходимых разрешений для мобильного приложения

Библиографический список

1. Маркин Д.О., Комашинский В.В., Баранов И.Ю. Модель управления профилем защиты мобильного устройства при доступе к услугам с разным уровнем конфиденциальности // Информационные технологии. – 2015 – 21(8). – с. 611- 618

2. Цыганенко Н. П. Статический анализ кода мобильных приложений как средство выявления его уязвимостей // Труды БГТУ. Серия 3: Физико-математические науки и информатика. 2015. №6 (179). – стр. 200-203

3. Parikh Vivek, Mateti Prabhaker. ASLR and ROP Attack Mitigations for ARM-Based Android Devices. // Communications in Computer

and Information Science – 2017 – pp.350-363. DOI: 10.1007/978-981-10-6898-0_29

4. Кучин И. Ю., Иксанов Ш. Ш., Белов С. В., Нургалиев М. М. Усовершенствование дискреционной модели доступа мобильных приложений к сервисам операционной системы Android // Вестник АГТУ. Серия: Управление, вычислительная техника и информатика – 2016. – №1. – стр. 17-25

5. Shabtai, Asaf, Mimran Dudu, Elovici Yuval. Evaluation of Security Solutions for Android Systems. [Электронный ресурс] // arXiv.org: портал: – URL: https://arxiv.org/abs/1502.04870 (дата обращения: 01.06.2019)

6. Grace Michael, Zhou Yajin, Zhang Qiang, Zou Shihong, Jiang Xuxian. RiskRanker: Scalable and Accurate Zero-day Android. // 10Th International Conference on Mobile Systems, Applications. – 2012 – pp. 345-362. DOI: 10.1145/2307636.2307663

7. Joshua J. Drake. Android Hacker's Handbook. 2st Edition / Wiley, 2019. – 155 с. ISBN 978-1118608647

8. Сковорода А.А., Гамаюнов Д.Ю. Анализ мобильных приложений с использованием моделей привилегий и API-вызовов вредоносных приложений. // Прикладная дискретная математика, издательство Изд-во ТГУ (Томск) – № 36 – с. 84-105. DOI: 10.17223/20710410/36/7

9. Rasthofer S., Asrar I., Huber S., Bodden E. How Current Android Malware Seeks to Evade Automated Code Analysis. In: Akram R., Jajodia S. Information Security Theory and Practice. // WISTP 2015. Lecture Notes in Computer Science – 9311 – 2015 DOI:.1007/978-3-319-24018-3_12

10. Кузин М, Бучка Н. Атака на Zygote: новый виток эволюции мобильных угроз [Электронный ресурс] // Securelist.ru: портал. – URL: https://securelist.ru/attack-on-zygote-a-new-twist-in-the-evolution-of-mobile-threats/28121/ (дата обращения: 01.06.2019)

11. Nitay Artenstein, Idan Revivo. Man in the binder: he who controls IPC, controls the droid. [Электронный ресурс] // Blackhat.com портал:. – URL: https://www.blackhat.com/docs/eu-14/materials/eu-14-Artenstein-Man-In-The-Binder-He-Who-Controls-IPC-Controls-The-Droid.pdf (дата обращения: 01.06.2019)

12. Guyton Fred. A survey of Android security threats and machine learning techniques used for detection [Электронный ресурс] // arXiv.org:

портал: – URL: https://arxiv.org/abs/1502.04870 (дата обращения: 01.06.2019). – DOI:10.13140/RG.2.2.16138.98244.

13. Li Bodong, Zhang Yuanyuan, Li Juanru Yang, Wenbo Gu Dawu. AppSpear : Automating the Hidden-Code Extraction and Reassembling of Packed Android Malware. // Journal of Systems and Software. – 2018. – pp. 49-75. DOI:10.1016/j.jss.2018.02.040.

14. Steven Arzt, Stephan Huber. Denial-of-App Attack: Inhibiting the Installation of Android Apps on Stock Phones. // 4th ACM Workshop on Security and Privacy in Smartphones & Mobile Devices. – 2018 – pp. 21-26. DOI: 10.1145/2666620.2666621

15. Сафин Л. К., Чернов А. В., Александров Я. А., Трошина К. Н. Исследование информационной защищенности мобильных приложений // Вопросы кибербезопасности. – 2015. – 4 (12). – стр. 28-37

16. Зубков К. Н., Диасамидзе С. В. Проблемы защиты информации в приложениях для мобильных систем. // Intellectual Technologies on Transport. – 2017. – №2. с. 40-46

17. Lu L. et al. Chex: statically vetting android apps for component hijacking vulnerabilities. // Proceedings of the 2012 ACM conference on Computer and communications security. ACM – 2012 – pp. 229-240.

18. Paul Sabanal, Hiding behind ART. [Электронный ресурс] // Blackhat.com портал:. – URL: https://www.blackhat.com/docs/asia-15/materials/asia-15-Sabanal-Hiding-Behind-ART-wp.pdf (дата обращения: 01.06.2019).

19. Jia Wan, Mohammad Zulkernine, Phil Eisen, Clifford Liem. Defending Application Cache Integrity of Android Runtime. // International Conference on Information Security Practice and Experience. – 2017 - pp 727-746. DOI: 10.1007/978-3-319-72359-4_45

20. Marhusin, M.F., Larkin Henry, Lokan Chris, Cornforth David. An Evaluation of API Calls Hooking Performance. // 2018 International Conference on Computational Intelligence and Security. – 2018. – 1. DOI: 10.1109/CIS.2018.199.

21. A. P. Felt, E. Ha, S. Egelman, A. Haney, E. Chin, and D. Wagner. Android permissions: User attention, comprehension, and behavior. // 8th Symposium on Usable Privacy and Security (SOUPS'12) – 2012. - pp. 3:1–3:14. DOI:10.1145/2335356.2335360

22. A. P. Felt, E. Chin, S. Hanna, D. Song, and D. Wagner, Android permissions demystified. // 18th ACM conference on Computer and communications security (CCS'11) – 2011 - pp. 627–638. DOI: 10.1145/2046707.204677

23. Kaladharan Yadu, Mateti Prabhaker, Kp Jevitha. An Encryption Technique to Thwart Android Binder Exploits. // Intelligent Systems Technologies and Applications – 2 – pp.13-21. DOI: 10.1007/978-3-319-23258-4_2.

24. Цифровая экономика: монография / И.Б. Тесленко. – Москва : Русайнс, 2018. — 284 с. — ISBN 978-5-4365-3040-6.

25. Overwhelming Majority of Android Devices Don't Have Latest Security Patches. [Электронный ресурс] // Securityweek.com портал:. – URL: http://www.securityweek.com/overwhelming-majority-android-devices-dont-have-latest-security-patches (дата обращения: 01.06.2019).

26. Google Play Protect провалил тест на эффективность [Электронный ресурс] // Securitylab.ru портал:. – URL: https://www.securitylab.ru/news/498889.php (дата обращения: 01.06.2019).

27. Кульба В.В., Шелков А.Б., Гладков Ю.М., Павельев С.В. Мониторинг и аудит информационной безопасности автоматизированных систем. – М., 2009 (Научное издание / Учреждение Российской академии наук Институт проблем управления им. В.А. Трапезникова РАН). – 94 с.

28. ГОСТ Р ИСО/МЭК 15408-2-2013. Национальный стандарт Российской Федерации. Информационная технология. Методы и средства обеспечения безопасности. Критерии оценки безопасности информационных технологий. Часть 2. Функциональные компоненты безопасности" (утв. и введен в действие Приказом Росстандарта от 08.11.2013 N 1339-ст)

29. Аудит Android-устройств на уязвимости BlueBorne [Электронный ресурс] // Bytemag.ru портал:. – URL: https://www.bytemag.ru/articles/detail.php?ID=33941 (дата обращения: 01.06.2019).

30. Sean Finley and Xiaojiang Du. Dynamic Cache Cleaning on Android. // IEEE International Conference on Communications (ICC). – 2013 – pp. 6143-6147. DOI: 10.1109/ICC.2013.6655587

31. Li, Yuping, Sathya Chandran Sundaramurthy, Alexandru G. Bardas, Xinming Ou, Doina Caragea, Xin Hu, and Jiyong Jang. Experimental study of fuzzy hashing in malware clustering analysis [Электронный ресурс] // CSET 15. 8th Workshop on Cyber Security Experimentation and Test. – URL: https://pure.tue.nl/ws/files/46945161/855432-1.pdf (дата обращения: 01.06.2019).

32. Wu Xueping, Zhang Dafang, Su Xin, Li WenWei. Detect repackaged Android application based on HTTP traffic similarity. // Security and Communication Networks. – 2015 – DOI: 8. 10.1002/sec.1170

33. Chen, Jian, Manar H. Alalfi, Thomas R. Dean, and Ying Zou. Detecting Android Malware Using Clone Detection.//Journal of Computer Science and Technology. – 2015 – 30(5). – pp. 942-956,– DOI: 10.1007/s11390-015-1573-7

34. Boojoong Kang, Suleiman Y. Yerima, Kieran Mclaughlin, Sakir Sezer. N-opcode analysis for android malware classification and categorization. // International Conference On Cyber Security And Protection Of Digital Services (Cyber Security). – 2015 – pp. 42-56. – DOI: 10.1109/CyberSecPODS.2016.7502343

35. Zhauniarovich Yury, Gadyatskaya Olga, Crispo Bruno, La Spina, Francesco and Moser, Ermanno. FSquaDRA: Fast Detection of Repackaged Applications // Data and Applications Security and Privacy XXVIII: 28th Annual IFIP WG 11.3 Working Conference. – 2014 – pp. 130-145

36. Hu Haoshi. Detecting repackaged android apps using server-side analysis [Электронный ресурс] // Eindhoven University of Technology – URL: https://pure.tue.nl/ws/files/46945161/855432-1.pdf (дата обращения: 01.06.2019)

37. Zhou, Wu, Xinwen Zhang, and Xuxian Jiang. AppInk: Watermarking Android apps for repackaging deterrence. // 8th ACM SIGSAC Symposium on Information, Computer and Communications Security. – 2013 – pp. 42-56. – DOI: 10.1145/2484313.2484315.

38. Zeng Qiang, Luo Lannan, Qian Zhiyun, Du Xiaojiang, Li Zhoujun. Resilient decentralized Android application repackaging detection using logic bombs. // International Symposium – 2018 – pp. 422-538. DOI: 50-61. 10.1145/3179541.3168820

39. Haehyun ChoJiwoong BangMyeongju JiJeong Hyun Yi. Mobile application tamper detection scheme using dynamic code injection against

repackaging attacks. // The Journal of Supercomputing. . – 2016 – 72(9) – pp. 3629–3645. DOI: 10.1007/s11227-016-1763-2

40. Enck W., Gilbert P., Chun B.-G., et al. TaintDroid: an information flow tracking system for real-time privacy monitoring on smartphones // Communications of the ACM. – 2014 – 57(3) – pp. 99-106. DOI:10.1145/2494522

41. Сковорода А.А., Гамаюнов Д.Ю. Обзор методов обнаружения и противодействия вредоносным приложениям для мобильных платформ // Безопасность информационных технологий. – 2015 – 22(2). – стр. 92-111

42. Enck W., Ongtang M., McDaniel P. On Lightweight Mobile Phone Application Certification // Proceedings of the 16th ACM Conference on Computer and Communications Security. – NY, USA: ACM. – 2009. – pp. 235-245. DOI:10.1145/1653662.1653691

43. Zhou Y., Wang Z., Zhou W., Jiang X., Hey You. Get Off of My Market: DetectingMalicious Apps in Official and Alternative Android Markets // 19th Annual Network & Distributed System Security Symposium. The Internet Society – 2012 – pp. 78-92.

44. Shabtai A., Tenenboim-Chekina L., Mimran D., et al. Mobile Malware Detection through Analysis of Deviations in Application Network Behavior. // Computers & Security. – 2014. – 43. – pp.1-18. DOI:10.1016/j.cose.2014.02.009.

45. Kim H., Smith J., and Shin K. G. Detecting energy-greedy anomalies and mobile malware variants // 6th International Conference on Mobile Systems, Applications, and Services. – 2008. – pp. 239–252. DOI:10.1145/1378600.1378627.

46. Skovoroda A. and Gamayunov D. Securing mobile devices: malware mitigation methods // J. Wireless Mobile Networks, Ubiquitous Computing, and Dependable Applications. – 2015. - 6(2). – pp. 78–97. DOI:10.22667/JOWUA.2015.06.31.078

47. Tan D. J. J. et al. Securing Android: A Survey, Taxonomy, and Challenges. // ACM Computing Surveys (CSUR). – 2015 – 47 – pp. 50. DOI: 10.1145/2733306

48. Вартанов С. П., Ермаков М. К. Применение статической инструментации байт-кода языка Java для динамического анализа программ // Труды ИСП РАН. 2015. №1.

49. Александров Я.А., Сафин Л.К., Трошина К.Н., Чернов А.В. Статический бинарный анализ мобильных приложений для платформы Android по требованиям информационной безопасности // Вестник Московского университета. Серия 15. Вычислительная математика и кибернетика. – 2016 – №3. – стр. 44 – 49.

50. Вартанов С.П., Ермаков М.К., Герасимов А.Ю. Прикладное применение динамического анализа программ, исполняющихся в интерпретирующих средах. // Труды ИСП РАН. – 2017 г., 29 (1) – стр. 135-148.

51. Rosen Sanae, Qian Zhiyun, Morely Mao Z.. AppProfiler: A flexible method of exposing privacy-related behavior in android applications to end users.// CODASPY 2013. Proceedings of the 3rd ACM Conference on Data and Application Security and Privacy. – 2013 – pp.45-65. DOI: 10.1145/2435349.2435380

52. Valerio Costamagna, Cong Zheng. ARTDroid. A Virtual-Method Hooking Framework on Android ART Runtime. // Innovations in Mobile Privacy and Security, London – 2016 – 1575 – pp. 20-28. – ISSN 1613-0073-1575-3

53. Backes M., Bugiel S., Schranz O., Styp-Rekowsky P., Weisgerber S. ARTist: The Android Runtime Instrumentation and Security Toolkit. // IEEE European Symposium on Security and Privacy (EuroS&P), Paris – 2017 – pp. 481-495. DOI: 10.1109/EuroSP.2017.43

54. Cho Haehyun, Lim Jongsu, Kim Hyunki, Hyun Yi Jeong. Anti-debugging scheme for protecting mobile apps on android platform // The Journal of Supercomputing. – 2015 – 72(1) - pp 232–246. DOI: 10.1007/s11227-015-1559-9

55. Воронин Алексей Александрович, Виноградов Дмитрий Викторович, Воронина Анна Аркадьевна Оценка защищенности среды выполнения операционной системы Android // Бюллетень науки и практики. 2019. №7. URL: https://cyberleninka.ru/article/n/otsenka-zaschischennosti-sredy-vypolneniya-operatsionnoy-sistemy-android (дата обращения: 02.05.2022).

ЗАКЛЮЧЕНИЕ

В настоящее время мобильные устройства являются неотъемлемой частью повседневной жизни. Рост популярности мобильных устройств способствует их более широкому внедрению в бизнес:

1) разрабатываются мобильные приложения, ориентированные как на сегмент b2c, так и b2b;

2) мобильные устройства становятся частью информационной инфраструктуры предприятия;

3) растет число использования личных мобильных устройств для выполнения рабочих обязанностей и т. д.

Использование мобильных приложений и облачных вычислений для бизнеса стало частью общей стратегии цифровой трансформации, которая предполагает существенную трансформацию модели ведения хозяйственной деятельности, в рамках которой формируются инновационные подходы к организации производства, логистики, взаимоотношений с контрагентами и т. д.

Использование мобильных приложений совместно с облачными вычислениями в рамках цифровой трансформации предприятия рассматривается как стратегическая задача по улучшению общей конкурентной позиции бизнеса на рынке, а не просто как поиск и использование резервов для снижения операционных затрат (например, затрат на содержание и обслуживание собственной ИТ-решений).

Источником стратегических преимуществ использования облачных вычислений выступают:

1) ускорение вывода на рынок новой продукции;

2) снижение стоимости ошибок при реализации ИТ-решений;

3) упрощение организации совместной работы с партнерами и клиентами и др.

В учебном пособии была рассмотрена совокупность эффективных подходов, инструментов и методов, направленных на разработку приложений для мобильных устройств. Изложены методы управления связанными с мобильными приложениями процессами на всех стадиях его жизненного цикла: от формирования замысла до прекращения функционирования. Рассмотрены особенности разработки и обеспечения информационной безопасности мобильных приложений для устройства под управлением операционной системы Android.

ПРИЛОЖЕНИЕ

СПИСОК ПОТЕНЦИАЛЬНО ОПАСНЫХ СТРУКТУР ПОЛЬЗОВАТЕЛЬСКИХ ПРИЛОЖЕНИЙ

ГРУППА M1: НЕКОРРЕКТНОЕ ИСПОЛЬЗОВАНИЕ ПЛАТФОРМЫ

M1. Отсутствует защита от атак «tapjacking»

Мобильное приложение не обеспечивает защиту от кражи, необходимую для предотвращения атак «tapjacking».

По умолчанию ОС Android позволяет мобильному приложению отображать пользовательский интерфейс через пользовательский интерфейс другого приложения, установленного и запущенного на устройстве.

Когда пользователь касается экрана, приложение может передать событие касания другому приложению ниже уровня пользовательского интерфейса, который пользователь не видит, выступая в качестве прокси-сервера для передачи непреднамеренных действий касания (данная атака очень похожа на «clickjacking» с учетом особенностей мобильных устройств).

Чтобы добиться успеха, вредоносное приложение должно быть уже установлено на мобильном телефоне жертвы. Примером эксплуатации может служить вредоносное приложение, которое вынуждает пользователя невольно нажимать кнопку оплаты (или любую другую функциональность) чувствительного приложения во время игры или выполнять другие невинные действия на экране вредоносного приложения. Это может быть использовано злоумышленником, чтобы обманом заставить пользователей приложения выполнить некоторые конфиденциальные действия с законным приложением (например, отправить платеж), которые они в противном случае не намерены делать.

Экспортированные широковещательные сообщения

Мобильное приложение содержит экспортированный приемник, позволяющий другим приложениям, включая вредоносные, отправлять намерения без ограничений. По умолчанию широковещательные приемники экспортируются на Android, в результате любое приложение сможет отправить намерение в широковещательный приемник

приложения. Чтобы определить, какие приложения могут отправлять намерения в широковещательный приемник мобильного приложения, необходимо установить соответствующие разрешения в файле манифеста Android.

Экспортированные действия с недостаточной защитой

Мобильное приложение содержит экспортированные действия, которые могут быть загружены и выполнены другими приложениями, находящимися на мобильном устройстве, в том числе вредоносными, чтобы вызвать законную активность приложения. Активность (Activitiy) – это компонент Android, который позволяет взаимодействовать с приложением определенным образом (например, выполнять определенные действия или функции).

Экспортированные поставщики содержимого с недостаточной защитой

Мобильное приложение содержит незащищенные экспортированные контент-провайдеры, которые могут раскрывать конфиденциальные данные приложения при определенных условиях. Контент-провайдеры обычно используются для обмена данными между различными приложениями. При экспорте без должной защиты любое приложение, установленное на устройстве, в том числе вредоносное, сможет раскрыть данные уязвимого приложения, включая любую конфиденциальную информацию, содержащуюся в нем. Чтобы безопасно экспортировать контент-провайдера, необходимо ограничить доступ к нему, настроив атрибуты «android:protectionl_evel» или «android:grantUriPermissions» в файле манифеста Android.

Экспортированные сервисы с недостаточной защитой

Мобильное приложение содержит экспортированный сервис. По умолчанию в Android сервисы не экспортируются и не могут быть вызваны другими приложениями. Однако если фильтр intent определен в файле манифеста Android, он экспортируется по умолчанию. Особое внимание следует уделить экспортируемым сервисам, так как без специальных разрешений они могут использоваться любыми другими приложениями, в том числе вредоносными.

Использование неявного намерения

Мобильное приложение использует неявное намерение, которое может быть небезопасным при определенных условиях. Намерения позволяют мобильным приложениям взаимодействовать друг с другом, запрашивая выполнение различных действий, для которых они лучше подходят. Однако неявное намерение не указывает, какому конкретному приложению отправляется запрос на выполнение действия. Если вредоносное приложение установлено на устройстве жертвы, оно также может получить неявное намерение, ответить на него и выполнить какое-либо действие вместо или в дополнение к законному приложению.

Использование фильтра намерений

Мобильное приложение использует фильтр intent который может представлять серьезную угрозу безопасности при неправильной реализации и фильтрации. Разработчики не должны полагаться исключительно на фильтры intent в целях безопасности, поскольку они не накладывают ограничений на явные намерения. Фильтры намерения определены в файле AndroidManifest они позволяют разработчикам выбирать, какой тип намерений их компоненты приложения должны получать и обрабатывать.

Жестко закодированные конфиденциальные данные

Мобильное приложение содержит конфиденциальные данные жестко. Злоумышленник, имеющий доступ к файлу мобильного приложения, может легко извлечь эти данные из приложения и использовать их в дальнейших атаках.

Устаревший setPluginState в WebView

Мобильное приложение использует устаревший метод setPluginState в WebView. Метод "setPluginState" устарел в Android API 18-го уровня, потому что плагины больше не будут поддерживаться в будущем и не должны использоваться. Примером плагинов, встроенных в PDF-ридер, флеш-плагин и т. д

JS CORS включен в WebView

Совместное использование ресурсов между источниками (CORS) включено в WebView. JavaScript используемый в мобильном приложении, может отправлять и получать данные с произвольных удаленных хостов. Это может быть риск, если удаленный узел олицетворяется или скомпрометирован.

JS включен в WebView

Мобильное приложение включило JavaScript в WebView. По умолчанию JavaScript отключен в WebView. Если он включен, он может вызвать различные проблемы безопасности, связанные с JS, такие как атаки межсайтового скриптинга (XSS).

Удаленная загрузка URL в WebView.

Загрузка удаленного URL-адреса может быть опасной практикой в WebView. Необходимо проверить взаимодействие с WebView и убедится в надежности, целостности и надежности сторонних URL-адресов, используемых в мобильном приложении.

ГРУППА M2: НЕБЕЗОПАСНОЕ ХРАНЕНИЕ ДАННЫХ

Раскрытие потенциально конфиденциальных данных

Мобильное приложение может привести к утечке конфиденциальной информации во время ее выполнения.

Информация:

1) или считается чувствительной в пределах собственной функциональности продукта, такой как личное сообщение;

2) или же предоставляет информацию о продукте или его среде, которая может быть полезна при атаке, но обычно недоступна для злоумышленника, например, путь установки продукта, который доступен удаленно.

Жестко закодированные данные

Мобильное приложение содержит отладочную или иную техническую информацию, которая может быть извлечена и использована злоумышленником для облегчения дальнейших атак.

База данных SQLite с открытым текстом

Мобильное приложение использует незашифрованную базу данных SQLite. К этой базе данных может получить доступ злоумышленник с физическим доступом к мобильному устройству или вредоносное приложение с корневым доступом к устройству. Приложение не должно хранить конфиденциальную информацию в открытом виде.

Включен режим отладки в мобильном приложении

Режим отладки используется разработчиками приложений в процессе разработки и должен быть отключен, когда приложение находится в рабочей среде. Этот режим может предоставить техническую информацию и может облегчить обратный инжиниринг приложения.

Включено резервное копирование приложений

Мобильное приложение использует внешнюю функциональность резервного копирования (по умолчанию Android механизм резервного копирования), которые могут хранить внутри конфиденциальных данных из приложения. При определенных условиях это может привести к раскрытию информации (например, при взломе сервера резервного копирования или аккаунта Gmail)

Внешнее хранение данных

Мобильное приложение может получить доступ к внешнему хранилищу (например. SD-карта) в режиме чтения или записи. Данные приложения, хранящиеся на внешнем хранилище данных, могут быть доступны другим приложениям (в том числе вредоносным) при определенных условиях и нести риск повреждения или подделки данных.

Создание файлов, доступных для чтения или записи

Мобильное приложение создает файлы с правами на чтение и запись. Такие файлы могут быть доступны и изменены другими приложениями, в том числе вредоносными, что ставит под угрозу целостность данных приложения.

ГРУППА M3: НЕБЕЗОПАСНАЯ ПЕРЕДАЧА ДАННЫХ

Возможна атака «Человек посередине»

Мобильное приложение может быть уязвимо для атаки MITM (Man-in-the-Middle) в том случае, когда мобильное приложение подключается к серверной части (например, API или веб-службе), отсутствует или неправильно реализована проверка имени узла (например, когда злоумышленник может перехватить трафик, находясь в той же беспроводной сети).

В случае успешной эксплуатации злоумышленник сможет перехватывать и манипулировать HTTPS-трафиком. красть и фальсифицировать конфиденциальные данные, отправляемые или получаемые приложением.

Использование незашифрованного протокола HTTP

Мобильное приложение использует протокол HTTP для отправки и получения данных. Конструкция протокола HTTP не предусматривает никакого шифрования передаваемых данных, которое может быть легко перехвачено, если злоумышленник находится в той же сети или имеет доступ к каналу данных жертвы.

ГРУППА M5: СЛАБАЯ КРИПТОГРАФИЧЕСКАЯ СТОЙКОСТЬ

Слабое шифрование

Слабые или плохо реализованные алгоритмы шифрования могут поставить под угрозу хранение и передачу данных, используемых мобильным приложением

Использование слабого вектора инициализации

При использовании в шифровании режимов блокчейн шифров (CBC) или обратной связи шифров (CFB) вектор инициализации должен быть непредсказуемым и случайным.

Предсказуемый генератор случайных чисел

Мобильное приложение использует предсказуемый генератор случайных чисел (ГСЧ). При определенных условиях этот недостаток

может поставить под угрозу шифрование данных мобильного приложения или другую защиту, основанную на рандомизации. Например, если маркеры шифрования создаются внутри приложения, и злоумышленник может предоставить приложению прогнозируемый маркер для проверки, а затем выполнить конфиденциальное действие в приложении или его серверной части.

Жестко закодированные ключи шифрования

Жестко закодированные ключи шифрования могут поставить под угрозу безопасное хранение и передачу данных в мобильном приложении при определенных обстоятельствах.

Слабые алгоритмы хэширования

Мобильное приложение использует слабые алгоритмы хеширования. Слабые алгоритмы хеширования (например. MD2. MD4. MD5 или SHA1) могут быть уязвимы к конфликтам и другим слабостям безопасности и не должны использоваться, когда требуется надежное хеширование данных.

ГРУППА M7: НИЗКОЕ КАЧЕСТВО КЛИЕНТСКОГО КОДА

Используется десериализация объекта

Десериализация объекта, выполняемая на ненадежном ресурсе (например, предоставленном пользователем вводе или внешнем хранилище), может быть опасной, если данные для десериализации подделаны злоумышленником.

Динамическая загрузка кода

Мобильное приложение использует динамическую загрузку исполняемого кода. При определенных обстоятельствах, динамическая загрузка кода может быть опасна. Например, если код расположен на внешнем хранилище (например. SD-карте), что может привести к уязвимости внедрения кода, если внешнее хранилище доступно для чтения и/или записи, и злоумышленник может получить к нему доступ

ГРУППА: ПРОЧИЕ

WebView имеет доступ к поставщикам контента

Мобильное приложение позволяет WebView получить доступ к поставщикам контента. Если WebView может получить доступ к поставщикам содержимого, он может получить доступ к URI содержимого, например "content//", для связи с поставщиком содержимого. Если эта функция включена, обратите внимание на использование url содержимого в WebView.

Ведение журнала событий

Мобильное приложение использует ведение журнала. Android использует систему ведения журнала с возможностью выбора уровня детализации (verbose, info, debug, warning and error). Эти журналы могут содержать конфиденциальные данные и представлять опасность для приложения, если к ним обращается третья сторона

Взаимодействие с базами данных SQL

Мобильное приложение взаимодействует с базами данных SQL

Динамическая загрузка кода

При определенных обстоятельствах, динамическая загрузка кода может быть опасно. Если код расположен на внешнем запоминающем устройстве (например. SD-карте) это может привести к инъекции кода, если внешнее запоминающее устройство является читаемым и/или доступным для записи.

Использование KeyStore

Мобильное приложение использует KeyStore Android системный компонент для добавления открытых и закрытых пар ключей в систему.

Использование TrustManager

Мобильное приложение может разрешить использование некоторых доверенных центров сертификации. Trust Manager позволяет разработчикам принимать сертификаты, которым не доверяет Android.

Эта функция может значительно облегчить атаки MiTM. Если диспетчер доверия настроен, злоумышленник сможет перехватывать трафик HTTPS и управлятьим.

Использование WebView

Мобильное приложение использует WebView для отображения веб-страниц

Использование сетевых сокетов

Мобильное приложение использует сетевые сокеты.

Использование системной команды

Мобильное приложение использует системные команды. При определенных условиях она может привести к выполнению произвольных команд.

Использование схемы URI file URI в WebView

WebView мобильного приложения использует URI "file://" access, который позволяет получить доступ к файлам во внутренней и внешней памяти.

Используется кодировка Base64

Мобильное приложение использует кодировку Base64. Необходимо убедиться, что никакие конфиденциальные данные не использует base64 вместо правильного шифрования.

Компонент хранения DOM используется в WebView

WebView мобильного приложения может хранить данные локально через компонент хранения DOM.

Не используются методы анти-эмуляции

Мобильное приложение не использует методы анти-эмуляции или анти-отладчика (например, обнаружение корневых устройств или проверка подлинности контактов). Это может значительно облегчить отладку приложений и процессы обратного проектирования.

Не используется конфигурация сетевой безопасности

Мобильное приложение не использует конфигурацию сетевой безопасности для определения сертификатов и центров сертификации (ЦС), которые могут использоваться для различных сред (например, разработка, тестирование и производство). Функция настройки сетевой безопасности в Android позволяет разработчикам приложений настраивать параметры сетевой безопасности в безопасном декларативном файле конфигурации без изменения кода приложения.

Низкий уровень защиты

Мобильное приложение использует низкий уровень защиты. Уровни защиты используются с элементом разрешения в Манифесте Android. Это позволяет разработчикам устанавливать пользовательские разрешения. Если уровень защиты слишком низок, это может позволить другим приложениям получить доступ к связанным компонентам приложения.

Предкомпилированное выполнение кода.

Список выполнений предкомпилированного кода, выполненных мобильным приложением во время выполнения. При определенных обстоятельствах выполнение предварительно скомпилированного кода может быть опасным и должно использоваться с большой осторожностью, когда приложение обрабатывает ненадежные входные данные (т. е. управляемые злоумышленником).

Создаются временные файлы

Мобильное приложение создает временные файлы. Несмотря на то что по умолчанию файлы кэша обычно являются закрытыми, рекомендуется убедиться, что временные файлы надежно удалены, когда они больше не требуются приложению.

Учебное издание

ВИНОГРАДОВ Дмитрий Викторович

РАЗРАБОТКА МОБИЛЬНЫХ ПРИЛОЖЕНИЙ И ОБЛАЧНЫЕ СЕРВИСЫ

Учебное пособие

Издается в авторской редакции

Подписано в печать 27.06.22.
Формат 60×84/16. Усл. печ. л. 13,72. Тираж 50 экз.
Заказ
Издательство
Владимирского государственного университета
имени Александра Григорьевича и Николая Григорьевича Столетовых.
600000, Владимир, ул. Горького, 87.