МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
федеральное государственное бюджетное образовательное учреждение высшего образования
«Тольяттинский государственный университет»

Кафедра Прикладная математика и информатика
(наименование)

02.03.03 Математическое обеспечение и администрирование информационных систем
(код и наименование направления подготовки / специальности)

Мобильные и сетевые технологии
(направленность (профиль) / специализация)

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА (БАКАЛАВРСКАЯ РАБОТА)

на тему «Разработка мобильного приложения для занятий фитнесом»

Обучающийся Т.А. Бирюков _____
(Инициалы Фамилия) (личная подпись)

Руководитель к.т.н., доцент, Д.Г. Токарев
(ученая степень (при наличии), ученое звание (при наличии), Инициалы Фамилия)

Тольятти 2025

Аннотация

Тема выпускной квалификационной работы: «Разработка мобильного приложения для занятий фитнесом».

Целью данной работы является проектирование и создание мобильного приложения, направленного на помощь пользователям в поддержании физической формы. Данное приложение предоставляет пользователям возможность эффективно организовывать свои тренировки, анализировать результаты и повышать качество выполнения упражнений.

Во введении раскрывается актуальность темы, которая обусловлена необходимостью разработки мобильного приложения для компании, где выполнялась работа для ВКР. Это приложение будет интегрировано с платформой, над которой в компании ведётся работа, что позволит расширить её экосистему за счет добавления функций, связанных со здоровьем и физической активностью.

В первой главе рассматривается анализ предметной области, исследуется компания, где проходила ВКР, и формулируются ключевые требования к разрабатываемому приложению. Для наглядного представления взаимодействия пользователя с системой была составлена диаграмма использования с применением CASE-средств.

Вторая глава посвящена этапам проектирования приложения, выбору подходящей среды разработки и языка программирования, а также разработке пользовательского интерфейса. На этом этапе были созданы диаграммы состояний и классов, что позволило логически структурировать работу приложения.

Третья глава описывает процесс реализации приложения, включая разработку ключевых экранов, внедрение сторонних библиотек для анализа данных и воспроизведения обучающих материалов, а также тестирование функционала. Проведенные испытания подтвердили корректную работу всех

компонентов приложения.

В заключении представлены итоги проделанной работы, в которых подробно рассматриваются достигнутые результаты и предлагаются возможные направления для дальнейшего развития и улучшения функционала приложения. Основным результатом выполненного проекта стала полноценная система, предназначенная для помощи пользователям в составлении индивидуальных программ тренировок. Кроме того, внедрение обучающих видео и текстового описания значительно повышает качество выполнения упражнений, обеспечивая пользователей наглядными и понятными инструкциями. Полученная система демонстрирует высокую степень практической применимости и может быть использована широкой аудиторией, заинтересованной в поддержании физической формы и здорового образа жизни.

Abstract

The title of the graduation work is Development of a Mobile Application for Fitness Training.

This work aims to design and implement a mobile application intended to assist users in maintaining their physical fitness.

The object of the graduation work is a fitness-oriented mobile application.

The subject of the graduation work is the process of designing, developing, and testing the system integrated with the company's existing platform.

Much attention is given to identifying user needs, and structuring the functional components of the application.

This work may be divided into several logically connected parts which are analysis of the subject area, system design and development of the user interface and implementation of key functionalities.

The first part describes in details the analysis of the domain, the study of the company where the work was carried out, and the formulation of key requirements for the application. We examine how user interaction can be visualized through UML use case diagrams created using CASE tools.

The second part outlines the stages of application design, including the selection of a suitable development environment and programming language, as well as the creation of the user interface. Diagrams of states and classes were developed to logically structure the operation of the application.

The third part discusses the implementation process, covering the development of key screens, integration of third-party libraries for data analysis and playback of training materials, and functional testing. The conducted tests confirmed the correct operation of all components.

In conclusion, we’d like to stress that the main result of the work is a ready-to-use system that allows users to create personalized workout programs, track progress, and improve the quality of exercises using instructional videos.

Оглавление

Введение 6
Глава 1 Анализ предметной области 8
1.1 Описание компании 8
1.2 Выбор case-средств для описания бизнес-процессов 12
1.3 Формирование требований для разрабатываемых приложений 14
Глава 2 Проектирование мобильного приложения 17
2.1 Выбор средств разработки 17
2.2 Выбор языка программирования 20
2.3 Проектирование интерфейса пользователя 22
2.4 Описание средств для решения поставленных задач 25
Глава 3 Реализация мобильного приложения 29
3.1 Разработка мобильного приложения 29
3.2 Тестирование приложения 37
Заключение 46
Список используемой литературы и используемых источников 48

Введение

В мире современных технологий кардинально меняются подходы к поддержанию физической формы. Мобильные приложения предлагают автоматизированные решения с функциями анализа тренировок, напоминаниями о занятиях и визуализацией прогресса.

Актуальность данной работы обусловлена растущим спросом на инструменты для персонального управления фитнес-активностью, однако данное приложение будет интегрировано в платформу, которая разрабатывается в «ВорлдИнтерТех РУС». Разработанное приложение призвано решить эти задачи, предоставив удобный и интуитивно понятный инструмент для организации тренировок.

Объектом исследования является процесс разработки мобильного приложения для занятий фитнесом.

Предмет исследования – методы и технологии, которые применяются при создании фитнес-приложений для платформы Android.

Цель данной работы заключается в создании мобильного приложения, которое предназначено для организации тренировок, позволяя пользователям создавать индивидуальные программы, отслеживать прогресс и просматривать видеоуроки для улучшения техники выполнения упражнений. Для успешной реализации этой цели необходимо решить ряд задач, среди которых: исследование предметной области, анализ деятельности компании, выявление требований к будущему приложению, а также обоснование выбора инструментов разработки. Важным этапом также является проектирование интерфейса пользователя, создание диаграмм классов и состояний, непосредственная разработка приложения и его тестирование для обеспечения стабильной и качественной работы.

В первой главе работы проводится анализ предметной области, в

котором описывается деятельность компании «ВорлдИнтерТех РУС», подробно рассматривается один из её бизнес-процессов, с использованием инструмента для анализа, который был выбран на основе сравнительного анализа различных CASE-средств. На основе этого анализа формируются требования к разрабатываемому приложению, и создаётся диаграмма вариантов использования, что помогает чётко зафиксировать интерфейсы взаимодействия пользователей с системой.

Во второй главе работы определяется основное программное обеспечение, которое будет использоваться для разработки, и выбирается язык программирования, необходимый для реализации проекта, проводится также проектирование мобильного приложения, включая создание диаграммы состояний и диаграммы классов, которые структурируют общую архитектуру приложения и позволяют обеспечить его функциональность и удобство использования.

Третья глава содержит подробное описание процесса создания приложения. В ней описываются этапы настройки проекта, реализация интерфейса и классов приложения, а также проводимое тестирование, этот этап является ключевым для обеспечения качественной и стабильной работы приложения и выявления возможных ошибок перед его запуском.

Глава 1 Анализ предметной области

1.1 Описание компании

Компания "ВорлдИнтерТех РУС" специализируется на создании программных продуктов и информационных систем, предназначенных для интеграции в сферу электронной коммерции. Основная деятельность организации заключается в обеспечении эффективного взаимодействия между бизнесом и потребителями, предлагая комплексные решения в области аудита, маркетинга и поддержки для розничной и оптовой торговли.

Внутри компании ключевая роль отводится разработчикам программного обеспечения, которые занимаются проектированием и реализацией решений на основе предоставленных технических требований. Именно они обеспечивают соответствие программных продуктов ожиданиям заказчиков и современным стандартам качества.

Выпускная квалификационная работа была выполнена на основе материалов компании, где основное внимание уделялось анализу потребностей клиентов, составлению подробных технических заданий и последующей разработке программного обеспечения, которое соответствует утвержденному плану. В процессе работы особое значение получали этапы тестирования и адаптации продукта под нужды пользователей, что позволило достигнуть высокого уровня функциональности и удобства пользования.

Организационная структура «ВорлдИнтерТех РУС» представлена на рисунке 1:

Рисунок 1 – Организационная структура «ВорлдИнтерТех РУС»

В подразделении отдела разработки — входят следующие сотрудники:

- менеджер проекта;
- senior-программист;
- middle-программист;
- junior-программист.

Функции сотрудников представлены на рисунке 2.

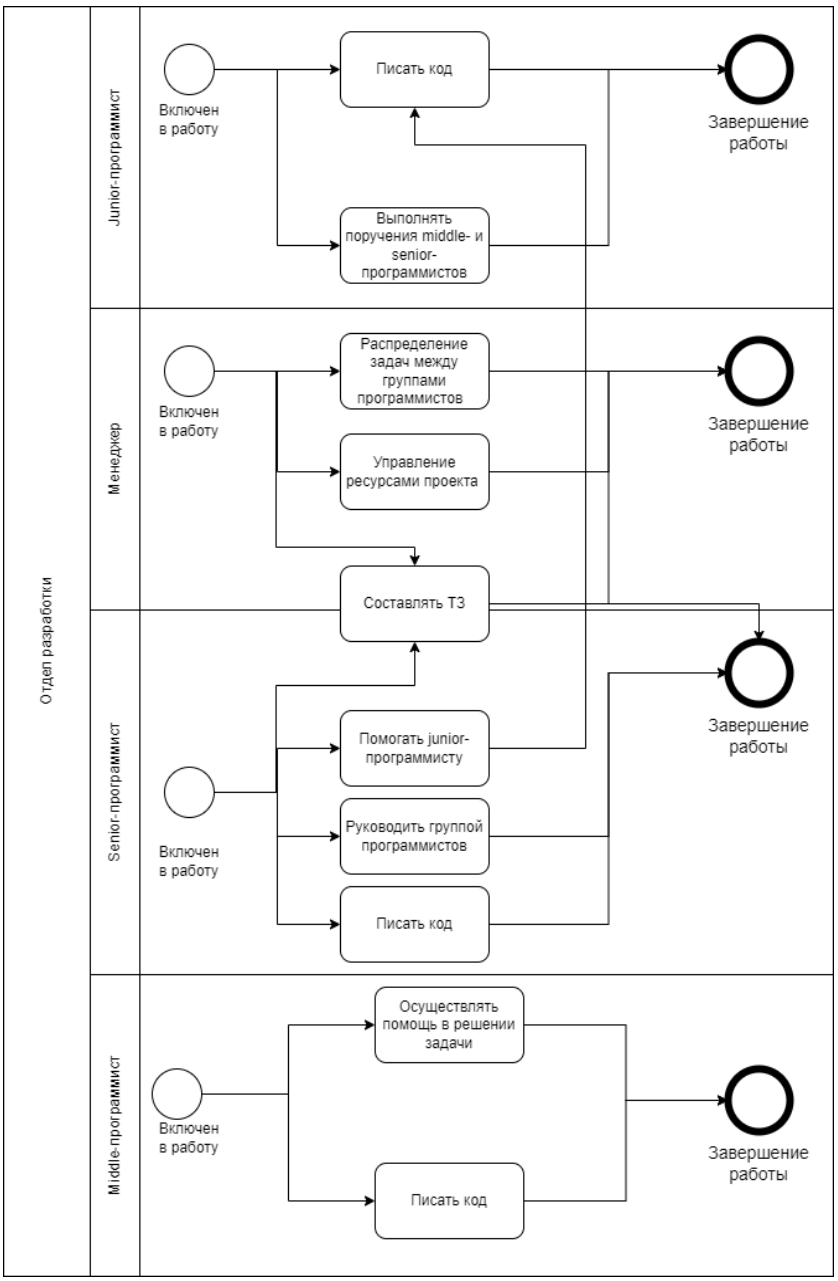

Рисунок 2 – Функции, выполняемые сотрудниками отдела разработок

Рассмотрим подробнее обязанности и особенности работы каждого из сотрудников, представленных на рисунке 2.

Junior-программист – это начинающий разработчик, который находится на этапе освоения профессиональных навыков в сфере создания программного обеспечения. Его основные обязанности включают выполнение простых задач, таких как поддержка существующих систем, внесение незначительных улучшений и написание тестов для проверки

корректности работы ПО. Время от времени ему могут поручать более сложные задания с целью развития его компетенций. Однако из-за недостатка опыта Junior-программисту часто требуется помощь и консультации со стороны более опытных коллег, таких как Middle или Senior-разработчики, чтобы успешно справляться с поставленными задачами.

Middle-программист – это специалист среднего уровня, обладающего достаточным опытом и знаниями для самостоятельного выполнения задач средней и повышенной сложности. Он способен эффективно решать технические проблемы, участвовать в разработке новых функциональностей и оптимизировать существующие решения. Кроме того, Middle-программист активно помогает младшим коллегам, делясь своим опытом, обучая их на практике и поддерживая в сложных ситуациях. Его роль заключается не только в написании качественного кода, но и в поддержании стабильности работы команды..

Senior-программист – это эксперт в области разработки программного обеспечения, чей высокий уровень квалификации позволяет решать наиболее сложные и комплексные задачи. Он специализируется на устранении критических ошибок, проектировании архитектуры новых систем и контроле качества кода, написанного другими членами команды. Senior также играет ключевую роль в планировании и оптимизации процессов разработки, предлагая стратегические решения для улучшения производительности проекта. Благодаря своему опыту, он часто выступает наставником для менее опытных коллег и консультантом для менеджеров проекта.

Менеджер проекта – это специалист, который отвечает за координацию всех этапов разработки и успешное достижение целей проекта, управляет ресурсами, распределяет задачи между участниками команды, определяет сроки выполнения работ и составляет технические задания, часто совместно с Senior-программистом. Менеджер выступает связующим звеном между заказчиком и разработчиками, уточняя требования клиента, согласовывая

промежуточные результаты и обеспечивая своевременную доставку продукта. Его работа требует отличных организационных навыков, умения находить баланс между интересами заказчика и возможностями команды, а также способности быстро реагировать на изменения в ходе проекта.

1.2 Выбор case-средств для описания бизнес-процессов

Бизнес-процесс — это последовательность взаимосвязанных действий, направленных на достижение определённой цели или результата. Для эффективного управления компанией необходимо чёткое понимание всех её процессов и их взаимодействия. Проектирование бизнес-процессов позволяет оптимизировать работу компании, устранить лишние этапы, повысить производительность и качество предоставляемых услуг.

Для описания бизнес-процессов существуют различные методы, включая текстовые, табличные и графические. «В последние годы наибольшую популярность получил графический метод, так как он наглядно представляет процессы, делает их более понятными для всех участников проекта и позволяет легко выявлять проблемы и точки улучшения» [7].

CASE-средства (Computer-Aided Software Engineering) — это программные инструменты, предназначенные для автоматизации процессов проектирования, разработки и документирования систем, включая описание бизнес-процессов. Они позволяют создавать диаграммы, моделировать процессы и структурировать данные.

«Рассмотрим некоторые из CASE-средств: Lucidchart, Microsoft Visio, Draw.io» [2], [10].

«Lucidchart - Облачный сервис с удобным интерфейсом и поддержкой совместной работы, подходит для командной работы над проектами, а также имеет интеграции с популярными платформами, такими как Slack, Google

Workspace и Microsoft Teams» [2], [10].

Microsoft Visio - Профессиональный инструмент, входящий в экосистему Microsoft Office, «который имеет широкий набор шаблонов и библиотек для создания сложных диаграмм и предоставляет возможности для глубокого анализа бизнес-процессов» [8].

Draw.io - Бесплатный веб-инструмент с простым интерфейсом, который позволяет создавать блок-схемы, диаграммы BPMN, UML и другие типы диаграмм, которые можно экспортировать диаграммы в формате JPG, PNG, SVG. Работает как в браузере, так и в виде десктопного приложения и поддерживает интеграцию с Google Drive, OneDrive и другими облачными сервисами.

Для сравнения выбранных инструментов были использованы такие критерии, как стоимость, функциональность, наглядность, удобство использования и интеграция. Результаты анализа представлены в таблице 1.

Таблица 1 – Сравнительный анализ CASE-средств на основе выделенных критериев

Критерий	Lucidchart	Microsoft Visio	Draw.io
Стоимость	Платно	Платно	Бесплатно
Наглядность	Высокая	Высокая	Высокая
Функциональность	Широкий набор инструментов	Широкий набор инструментов	Основные типы диаграмм
Удобство использования	Простой интерфейс	Сложный для новичков	Простой интерфейс
Интеграция	Google Workspace, Slack, Teams	Microsoft Office	Google Drive, OneDrive

На основе проведённого анализа был выбран инструмент Draw.io. Этот выбор обусловлен несколькими ключевыми факторами: Draw.io является

полностью бесплатным решением, что делает его доступным. Инструмент предлагает достаточно широкий набор функций для создания различных типов диаграмм. Кроме того, простота интерфейса позволяет быстро освоить работу с ним даже новичкам, а поддержка интеграции с популярными облачными сервисами, такими как Google Drive и OneDrive, обеспечивает удобство хранения и совместного доступа к созданным диаграммам. Таким образом, Draw.io стал оптимальным выбором благодаря сочетанию простоты, функциональности и доступности.

1.3 Формирование требований для разрабатываемых приложений

Разработка любого программного продукта начинается с формирования требований, которые определяют функциональность и характеристики будущего приложения, требования служат основой для проектирования, разработки и тестирования системы, а также помогают обеспечить соответствие ожиданиям пользователей. Процесс формирования требований включает анализ потребностей целевой аудитории и формулировку задач, которые приложение должно решать.

Требования к разрабатываемому приложению можно разделить на два типа – функциональные и нефункциональные.

«Функциональные требования – это требования, описывающие конкретные действия или возможности, которые система должна выполнять.» [7].

«Для разрабатываемого мобильного приложения были выделены следующие ключевые функциональные требования» [6]:

- реализация системы регистрации и авторизации пользователя;
- создание индивидуальных тренировок;
- просмотр инструкции по работе приложения;
- просмотр видеоуроков;

- редактирование программ тренировок.

Также важно определить нефункциональные требования.

Нефункциональные требования характеризуют качество работы системы, такие как производительность, удобство использования, безопасность или совместимость.

Рассмотрим основные нефункциональные требования, которые предъявляются к данному приложению:

- минимальное время отклика интерфейса;
- стабильность работы приложения;
- минималистичный и интуитивно понятный дизайн;
- возможность расширения функционала за счёт добавления новых функций.

Следуя данным требованиям, можно создать приложение, которое будет соответствовать всем установленным стандартам и классификациям.

С помощью инструмента Draw.io была разработана диаграмма вариантов использования для наглядного представления взаимодействия пользователя с системой. Благодаря этой визуализации можно понять основные сценарии работы с приложением и возможные варианты его использования.

На диаграмме вариантов использования, показанной на рисунке 3, изображено предполагаемое взаимодействие пользователя с приложением.

Рисунок 3 – Диаграмма вариантов использования приложения

Пользователь должен иметь возможность создать профиль, создать и редактировать персональную программу тренировок, просматривать инструкцию по работе приложения, просматривать видеоуроки для правильного выполнения техники упражнения.

Выводы по главе 1

В первой главе выпускной квалификационной работы была рассмотрена деятельность компании «ВорлдИнтерТех РУС», включая анализ её структуры. Для описания и моделирования бизнес-процессов было выбрано соответствующее CASE-средство. Проведённый анализ разрабатываемого приложения позволил сформировать перечень функциональных и нефункциональных требований, на основе выявленных требований была составлена диаграмма вариантов использования, которая отражает взаимодействие пользователей с системой.

Глава 2 Проектирование мобильного приложения

2.1 Выбор средств разработки

«Android — это одна из самых популярных операционных систем в мире, которая занимает лидирующие позиции на мировом рынке мобильных операционных систем. Android является открытой платформой с открытым исходным кодом, что позволяет разработчикам свободно адаптировать и изменять систему под свои нужды, а также Android работает на различных устройствах от бюджетных смартфонов до флагманских моделей, что позволяет оптимизировать приложения под разные устройства, обеспечивая доступность для широкого круга пользователей.» [1]. «Также эта операционная система поддерживается Google Play — официальным магазином приложений для Android, который предоставляет разработчикам доступ к глобальной аудитории.»[4].

Для создания мобильных приложений под Android существует множество инструментов и сред разработки, каждая из которых имеет свои особенности, преимущества и ограничения.

«Flutter представлет собой современный фреймворк для кроссплатформенной разработки приложений», [3], [4] «который позволяет разработчикам создавать приложения для Android, iOS, веба и даже десктопных платформ, используя язык программирования Dart, отличающийся простотой и высокой производительностью» [4]. Flutter также известен своей активной документацией и сообществом, что делает его доступным даже для новичков.

«Xamarin предлагает разработчикам возможность использовать единый код для обеих платформ Android и iOS, что значительно сокращает время и ресурсы, необходимые для разработки. При этом предоставляет доступ к нативным API обеих платформ, что обеспечивает высокую скорость работы

приложений» [3].

NetBeans IDE изначально была создана для работы с Java, но со временем расширила свои возможности и стала поддерживать множество языков программирования, включая PHP, HTML, CSS, JavaScript и другие. Среда предоставляет мощные инструменты для создания пользовательских интерфейсов, управления проектами и отладки кода, а одним из ее главных преимуществ является простота и удобство использования, что делает ее привлекательным выбором для новичков. Хотя NetBeans не является специализированной средой для разработки мобильных приложений под Android, её гибкость позволяет адаптировать её для этой задачи, но с использованием дополнительных плагинов.

«Android Studio — это официальная интегрированная среда разработки (IDE), созданная компанией Google, считается стандартом для разработки мобильных приложений под Android и предоставляет широкий набор инструментов для создания, тестирования и оптимизации приложений» [1], [13]. Android Studio поддерживает различные языки программирования, такие как Java, Kotlin и C++, что делает её универсальной, предлагает инструменты для анализа производительности, а также «имеет встроенный эмулятор, который позволяет тестировать приложения на различных устройствах и версиях Android» [3].

Для более точного анализа сравнение будет происходить по шести параметрам: доступность, удобство среды, простота настройки, производительность, поддержка платформ, и интеграци с API. Все результаты анализа представлены в таблице 2, где указаны как достоинства, так и недостатки каждой IDE.

Таблица 2 – Сравнительный анализ средств разработки

Среда разработки	Доступность	Удобство среды	Простота настройки	Производи-тельность	Поддержка платформ	Интеграция с API
Flutter	+	+	+	+	-	+
Xamarin	-	+	-	+	+	+
NetBeans IDE	+	-	-	-	-	-
Android Studio	+	+	+/-	+	+	+

«По итогам сравнительного анализа, был выбран Android Studio» [13], так как данная среда специализируется на платформе Android и имеет постоянную поддержку со стороны Google, что делает её незаменимым выбором для создания нативных приложений, а также ей не требуется установка дополнительных плагинов, в отличие от других интегрированных сред разработки.

2.2 Выбор языка программирования

«Как упоминалось ранее, Android Studio» [13] «поддерживает несколько языков программирования: Kotlin, Java и C++» [19], [12].

Kotlin — это современный язык программирования, созданный компанией JetBrains и официально поддерживаемый Google как основной язык для разработки Android-приложений. «Этот язык был разработан специально для устранения недостатков Java и предоставления более удобного и безопасного инструмента для разработчиков, одним из ключевых преимуществ Kotlin является его совместимость с Java, что позволяет использовать существующий код и библиотеки без необходимости полного переписывания, это делает переход на Kotlin максимально простым для команд, которые ранее работали с Java» [19]. Kotlin отличается лаконичным и читаемым синтаксисом, что значительно уменьшает объем кода и снижает вероятность ошибок. Особого внимания заслуживает поддержка корутин в Kotlin, которая значительно упрощает работу с асинхронными задачами, такими как сетевые запросы или работа с базами данных [12].

«Java — это классический язык программирования, который долгое время был основным языком для разработки Android-приложений, он широко используется не только в мобильной разработке, но и в других областях, таких как веб-разработка, серверное программирование и создание корпоративных решений» [14]. Одним из главных преимуществ Java является её универсальность, она поддерживается множеством платформ и имеет огромное сообщество разработчиков, это может упростить поиск решений и помощи в случае возникновения проблем, но Java имеет и свои недостатки, например, её синтаксис требует написания большого объема кода, что может замедлить процесс разработки и увеличить вероятность ошибок. Тем не менее, «Java остается надежным выбором для проектов, где требуется стабильность и совместимость с существующими системами» [15], [19].

C++ — это мощный язык программирования низкого уровня, который используется для создания высокопроизводительных приложений. В Android он применяется для разработки нативных модулей, таких как игры, графические приложения или компоненты, требующие интенсивных вычислений, а также C++ обеспечивает максимальную производительность за счет прямого управления памятью и аппаратными ресурсами, но несмотря на все плюсы, он также имеет свои недостатки. Разработка на нем требует глубоких знаний и внимательности, а синтаксис C++ сложнее, чем у Kotlin или Java, что замедляет процесс разработки и увеличивает риск ошибок.

Для удобства проведем сравнение языков программирования, обращая внимание на такие ключевые аспекты, как быстродействие, лаконичность кода, распространенность на рынке и доступность документации. Подробные результаты этого анализа представлены в таблице 3.

Таблица 3 – Сравнение языков программирования

Язык программирования	Простота использования	Производительность	Безопасность кода	Поддержка и актуальность
Java	+/-	+	-	+
Kotlin	+	+	+	+
C++	-	+	-	+/-

На основе проведенного анализа можно сделать вывод, что Kotlin является оптимальным выбором для разработки мобильных приложений под Android, предлагая идеальный баланс между простотой использования, производительностью и безопасностью кода. Благодаря своей лаконичности и совместимости с Java, Kotlin позволяет разработчикам быстрее создавать качественные приложения, избегая распространенных ошибок.

2.3 Проектирование интерфейса пользователя

Проектирование пользовательского интерфейса является одним из ключевых этапов разработки мобильного приложения, так как именно через интерфейс пользователи взаимодействуют с системой. На этом этапе важно обеспечить удобство, интуитивную понятность и функциональность всех экранов приложения. Для структурирования процесса проектирования был использован инструмент Draw.io с применением нотации UML (Unified Modeling Language) для создания диаграмм состояний, которые помогают визуализировать переходы между экранами и логику взаимодействия пользователя с приложением: авторизация, главная страница, профиль, создание программ тренировок, просмотр статистики и видеоуроков.

Для описания поведения приложения была составлена диаграмма состояний (State Machine Diagram) [11], отражающая основные экраны и переходы между ними. Диаграмма состояний позволяет четко представить, как пользователь будет перемещаться по приложению, какие действия он может выполнять на каждом экране и как система реагирует на его действия. Детали диаграммы состояний можно увидеть на рисунке 4.

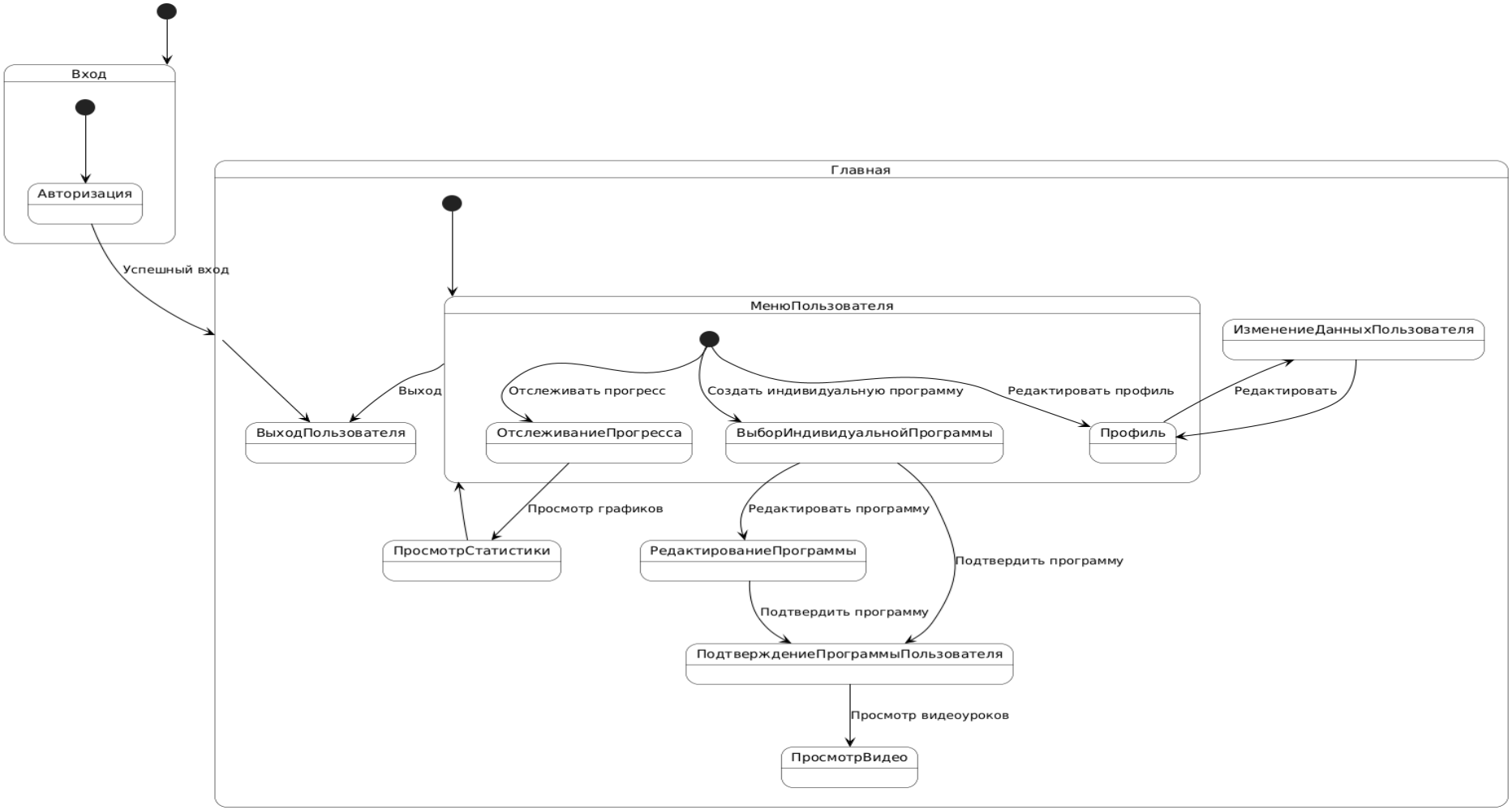

Рисунок 4 – Диаграмма состояний

Первый экран, с которого начинается взаимодействие пользователя с приложением это экран авторизации, здесь пользователь может войти в систему, используя электронную почту и пароль, или зарегистрироваться, если у него еще нет аккаунта. Интерфейс выполнен минималистично, чтобы не перегружать пользователя лишними элементами.

После успешной авторизации пользователь попадает на главную страницу, которая содержит приветственное сообщение, через эту страницу можно попасть в меню пользователя, которое предоставляет доступ ко всем основным разделам приложения, таким как переход к экрану отслеживания прогресса, выбор индивидуальной программы тренировок и редактирование профиля.

Проектирование интерфейса пользователя было выполнено с приоритетом на удобство использования и функциональность. Применение диаграммы состояний в нотации UML позволило точно выстроить логику переходов между экранами, обеспечивая четкую последовательность действий пользователя, такой подход гарантирует, что пользователи смогут легко освоить приложение и эффективно использовать его [16].

2.4 Описание средств для решения поставленных задач

Для создания мобильного приложения для занятий фитнесом была выбрана платформа Android Studio с языком Kotlin, что обеспечивает современный и эффективный инструментарий для разработки. В проекте используются разнообразные библиотеки и компоненты, которые обеспечивают необходимую функциональность и делают взаимодействие пользователя с приложением максимально удобным.

Вместо традиционной SQL-базы данных для хранения пользовательских данных было принято решение использовать DataStore с JSON-сериализацией. Такой подход позволяет легко и быстро сохранять и загружать данные, а также обеспечивает простоту интеграции и масштабирования приложения. Это решение отлично подходит для хранения персональных упражнений и параметров пользователя, а также соответствует требованиям к быстродействию и надежности.

Данные приложения хранятся в DataStore — современном и удобном механизме для работы с парами «ключ-значение». Пользовательские упражнения, представленные в виде объектов класса `UserExercise`, сериализуются в JSON с помощью библиотеки Gson и сохраняются в отдельном пространстве DataStore под определённым ключом. Такой метод позволяет эффективно управлять различными параметрами упражнений, включая название, описание, количество подходов, повторов и время выполнения. Параметры профиля пользователя, такие как имя, рост и вес, хранятся аналогичным образом. Этот подход облегчает разработку и ускоряет работу приложения.

При создании мобильного приложения были использованы ключевые компоненты Android SDK, которые обеспечивают эффективную реализацию функциональности, высокую производительность и удобство разработки.

Для создания пользовательского интерфейса был выбран современный

декларативный подход с использованием Jetpack Compose, который позволяет описывать интерфейс непосредственно в коде Kotlin. Одним из ключевых компонентов Jetpack Compose является Scaffold, который предоставляет готовый шаблон для создания структуры экрана, он включает базовые элементы интерфейса, такие как AppBar для заголовков и кнопок действий, Floating Action Button (FAB) для выполнения основных операций, Drawer для боковой панели навигации и BottomBar для быстрого доступа к ключевым функциям. Использование Scaffold значительно упрощает разработку интерфейса, так как он уже содержит все необходимые компоненты для создания стандартного экрана, что экономит время и обеспечивает единообразие дизайна.

Управление состоянием пользовательского интерфейса в Jetpack Compose осуществляется через механизм State, в отличие от традиционного подхода, где состояние хранится в ViewModel или других компонентах, Jetpack Compose использует реактивные переменные состояния, которые автоматически обновляют интерфейс при изменении данных. State представлен через объекты `mutableStateOf` и `State`, и когда значение переменной изменяется, Compose перерисовывает только те части интерфейса, которые зависят от этого значения. Это делает управление данными простым и эффективным, снижая вероятность ошибок и ускоряя разработку.

Для работы с асинхронными задачами был выбран Kotlin Coroutines, который представляет собой современный инструмент для выполнения долгих операций без блокировки основного потока. Корутины — это легковесные потоки, которые могут быть приостановлены и возобновлены, что делает их идеальным выбором для задач, таких как загрузка данных с сервера, чтение и запись данных в DataStore или выполнение фоновых вычислений. Корутины значительно упрощают работу с асинхронными задачами по сравнению с традиционными подходами, такими как Callbacks

или RxJava, делая код более читаемым и менее подверженным ошибкам, а также корутины легко интегрируются с другими компонентами, такими как LiveData и ViewModel.

Таким образом, использование компонентов Android SDK, таких как RecyclerView, Scaffold, State и Kotlin Coroutines, позволило создать эффективное и удобное мобильное приложение.

Компоненты, использованные в разработке мобильного приложения для занятий фитнесом, играют ключевую роль в создании удобных интерфейсов, управления взаимодействием с пользователями и обеспечения эффективной передачи данных между различными экранами приложения, они обеспечивают не только функциональность, но и структурированность системы, что критически важно для её масштабируемости и поддержки.

Архитектура мобильного приложения была тщательно спроектирована с учетом требований к производительности, удобству использования и надежности. Для наглядного представления структуры приложения была составлена «диаграмма классов, которая демонстрирует ключевые классы и их взаимосвязи» [17]. Эта диаграмма помогает лучше понять, как организованы основные компоненты системы, и упрощает восприятие всей архитектуры. Диаграмма классов для мобильного приложения представлена на рисунке 5.

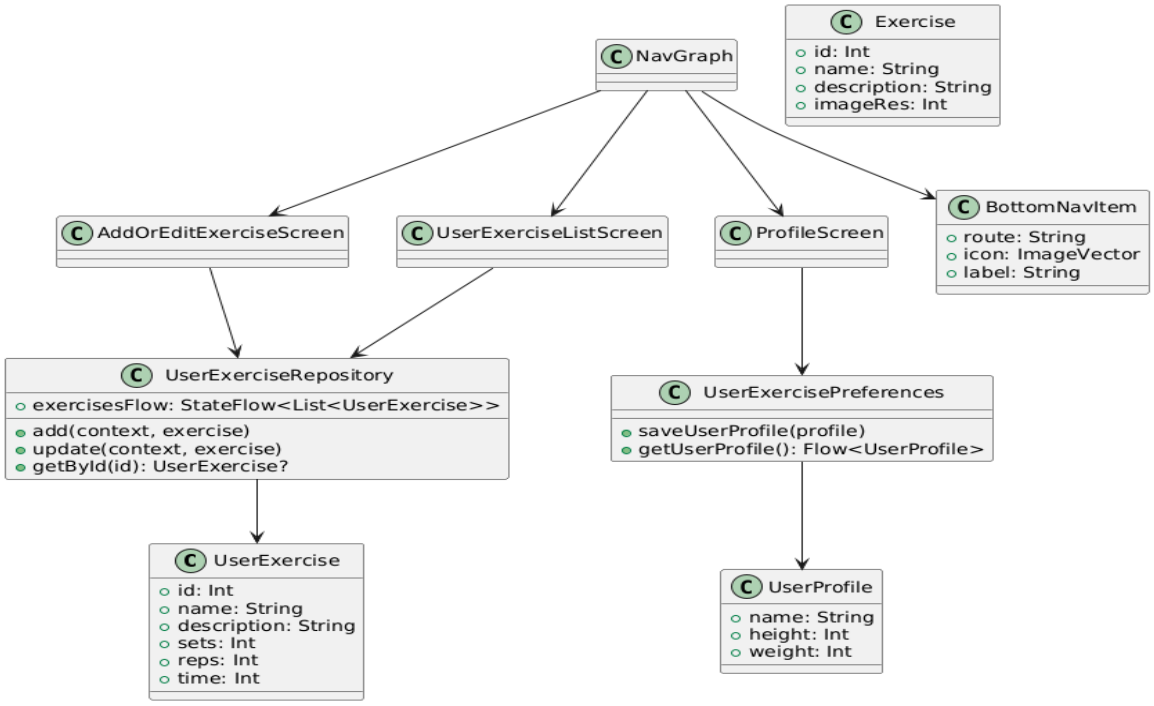

Рисунок 5 – Диаграмма классов мобильного приложения

Выводы по главе 2

Во второй главе было обоснованно использование инструментов для разработки приложения. В качестве основной среды была выбрана Android Studio, а для написания кода использован язык Kotlin, для описания взаимодействия пользователя с системой была составлена диаграмма состояний, отражающая последовательность действий и переходов между экранами и были разработаны макеты интерфейса, определяющие ключевые элементы дизайна и архитектурные компоненты. Особое внимание уделялось способам хранения данных: для сохранения информации о пользователе и его тренировках применен DataStore с использованием JSON-сериализации через библиотеку Gson, это решение обеспечивает удобство работы с данными и их быстрый доступ. А для лучшего понимания структуры приложения была создана диаграмма классов, демонстрирующая основные компоненты системы и их взаимодействие, что позволяет упростить процесс разработки и обеспечить четкую организацию кода.

Глава 3 Реализация мобильного приложения

3.1 Разработка мобильного приложения

Для начала был создан проект в Android Studio. После установки среды разработки мы переходим в окно создания проекта, выбираем шаблон "Empty Activity" (рисунок 6), заполняем данные нашего приложения (название, пакет, язык Kotlin) и нажаем "Finish" (рисунок 7). После этого создается базовый проект с поддержкой Jetpack Compose.

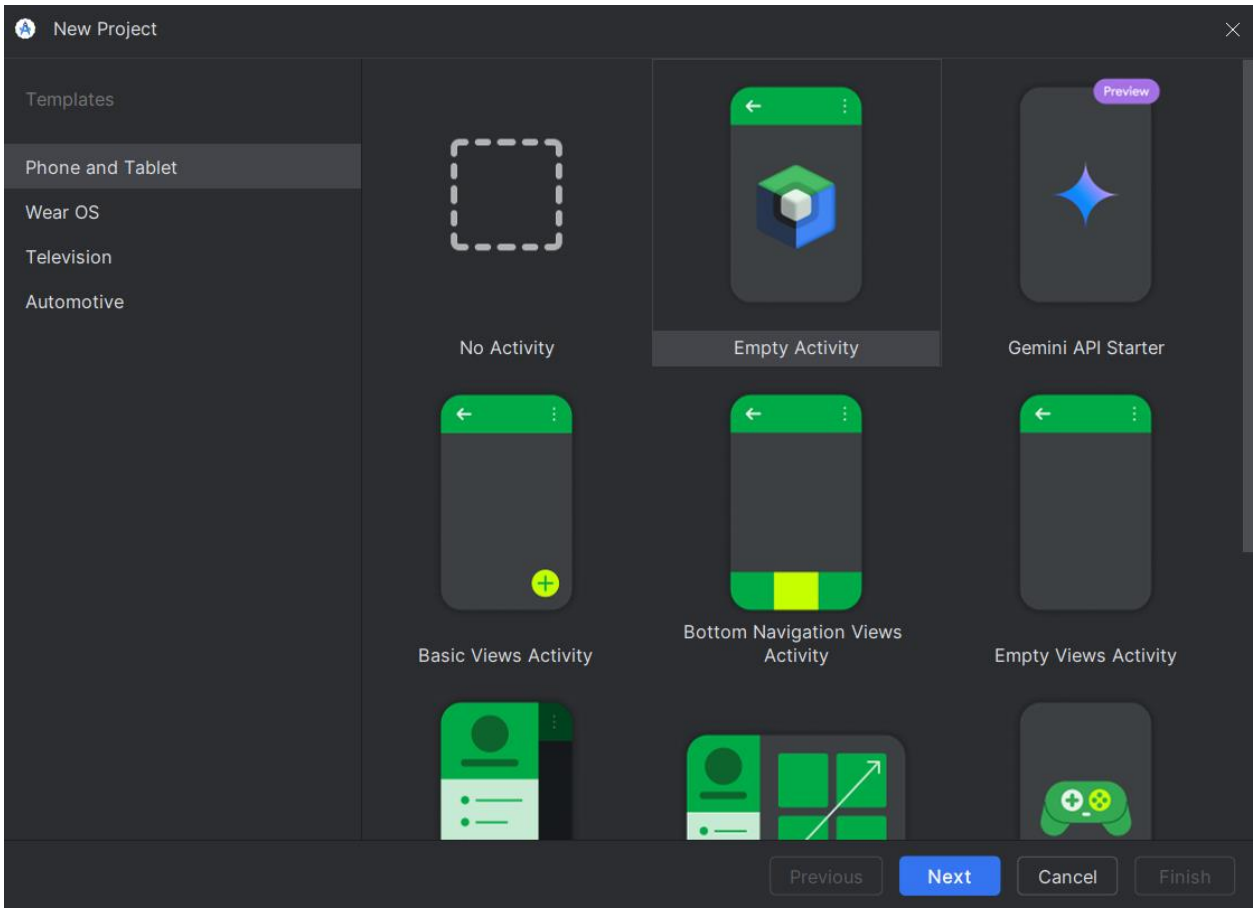

Рисунок 6 – Выбор главного экрана для приложения

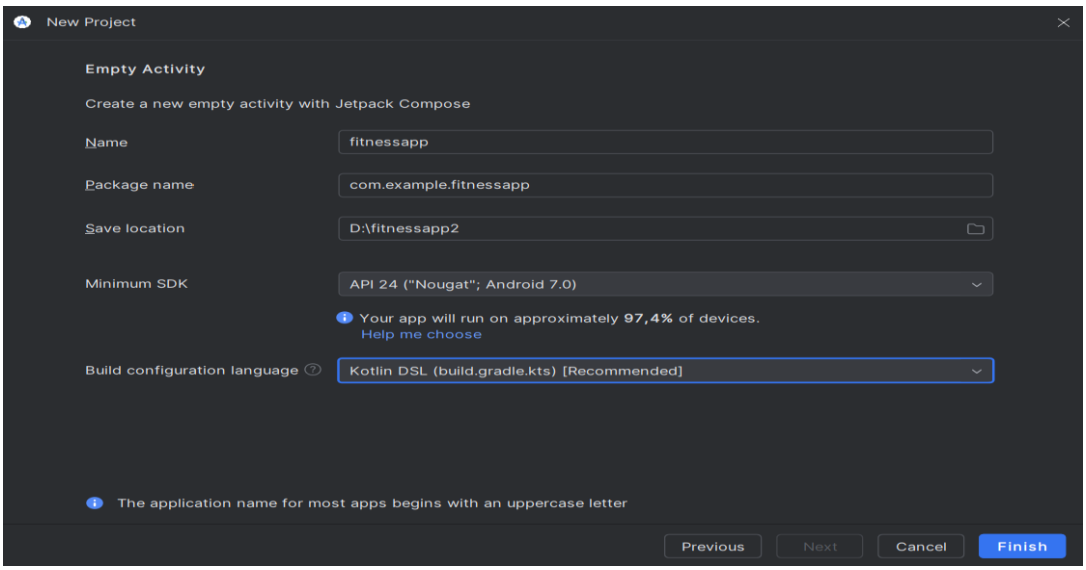

Рисунок 7 – Заполнение данных приложения

Для корректной работы над проектом необходимо подключить требуемые библиотеки, добавив их в список зависимостей. Все необходимые зависимости прописываются в конфигурационном файле проекта “build.gradle.kts” , который используется в Android Studio. Этот файл является ключевым элементом системы сборки Gradle, предоставляя разработчикам возможность настраивать различные аспекты проекта.

С помощью данного файла можно выполнять следующие задачи:

- Подключать необходимые библиотеки и плагины;
- Указывать параметры сборки, такие как версии компилятора и целевые платформы;
- Создавать пользовательские задачи для автоматизации процессов;
- Определять репозитории, из которых будут загружаться зависимости;
- Задавать метаданные проекта, такие как его название, версия и другие параметры (Рисунок 8).

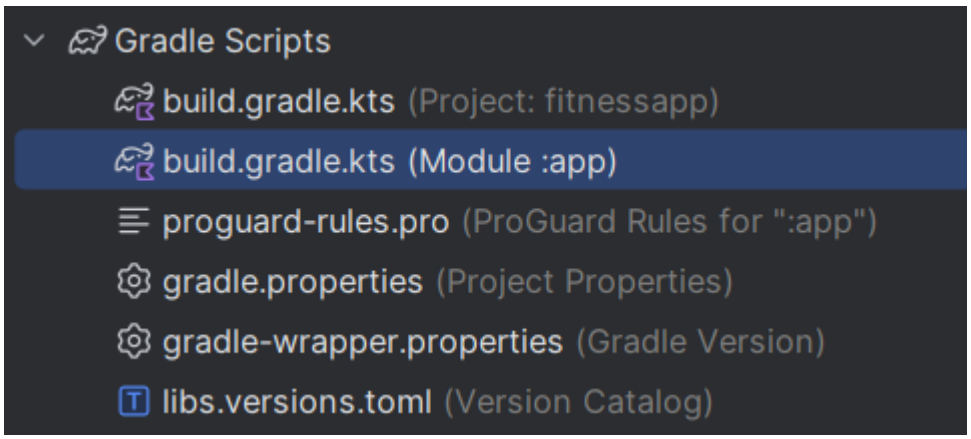

Рисунок 8 – Конфигурационный файл Gradle

Основные зависимости будут составлять библиотеки для работы с таймером, управления асинхронными задачами (Рисунок 9).

```
dependencies {
    implementation(libs.androidx.core.ktx)
    implementation(libs.androidx.lifecycle.runtime.ktx)
    implementation(libs.androidx.activity.compose)
    implementation(platform(libs.androidx.compose.bom))
    implementation(libs.androidx.ui)
    implementation(libs.androidx.ui.graphics)
    implementation(libs.androidx.ui.tooling.preview)
    implementation(libs.androidx.material3)
    testImplementation(libs.junit)
    androidTestImplementation(libs.androidx.junit)
    androidTestImplementation(libs.androidx.espresso.core)
    androidTestImplementation(platform(libs.androidx.compose.bom))
    androidTestImplementation(libs.androidx.ui.test.junit4)
    debugImplementation(libs.androidx.ui.tooling)
    debugImplementation(libs.androidx.ui.test.manifest)
    implementation ("androidx.navigation:navigation-compose:2.5.3")
    implementation ("androidx.lifecycle:lifecycle-runtime-compose:2.6.0")
    implementation ("androidx.media3:media3-exoplayer:1.2.0")
    implementation ("androidx.media3:media3-ui:1.2.0")
```

Рисунок 9 – Зависимости приложения

Для проекта мне понадобится картинка для фона приложения, а также

GIF-изображения для визуализации правильной техники выполнения упражнений. Для этого я скачал их в форматах jpeg и GIF соотвественно, далее задал им названия которые будут использоваться в проекте приложения. Все они будут хранится в папке drawable, которая используется для хранения графических ресурсов, таких как изображения, иконки, фигуры, градиенты, а также других визуальных элементов. Это показано на рисунке 10.

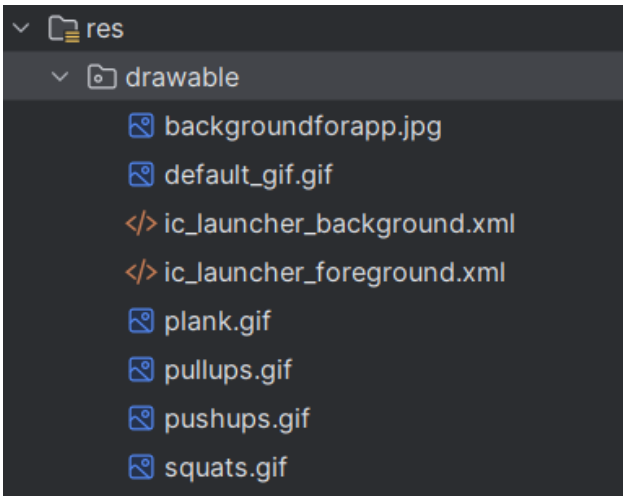

Рисунок 10 – Ресурсы проекта

После создал экран главного меню, на котором нас встречает приветственный текст и расположена кнопка навигации к списку упражнений. Это можно увидеть на рисунке 11.

```
fun HomeScreen(navController: NavHostController) {
    Scaffold(
        topBar = { TopAppBar(title = { Text( text: "Фитнес Приложение") }) },
        containerColor = Color.Transparent
    ) { padding ->
        Column(
            modifier = Modifier
                .fillMaxSize()
                .padding(padding)
                .padding(16.dp)
        ) {
            Text( text: "Добро пожаловать в фитнес-приложение!", style = MaterialTheme.typography.headlineMedium)
            Spacer(modifier = Modifier.height(20.dp))
            Button(onClick = { navController.navigate( route: "exercises") }) {
                Text( text: "Список упражнений")
            }
        }
    }
}
```

Рисунок 11 – Экран главного меню

Далее определил список упражнений, и их подробное описание. Данный список представлен на рисунке 12.

```
data class Exercise(val name: String, val description: String)

val exercises = listOf(
    Exercise( name: "Отжимания",  description: "Укрепляет грудные мышцы, плечи и трицепсы."),
    Exercise( name: "Приседания",  description: "Развивает ноги и ягодицы."),
    Exercise( name: "Планка",  description: "Укрепляет корпус и спину."),
    Exercise( name: "Подтягивания",  description: "Развивает силу рук и спины."),
)
```

Рисунок 12 – Список упражнений

Далее создал карточку с кликабельными элементами, которая будет перемещать пользователя на экран деталей упражнений. Карточка представлена на рисунке 13.

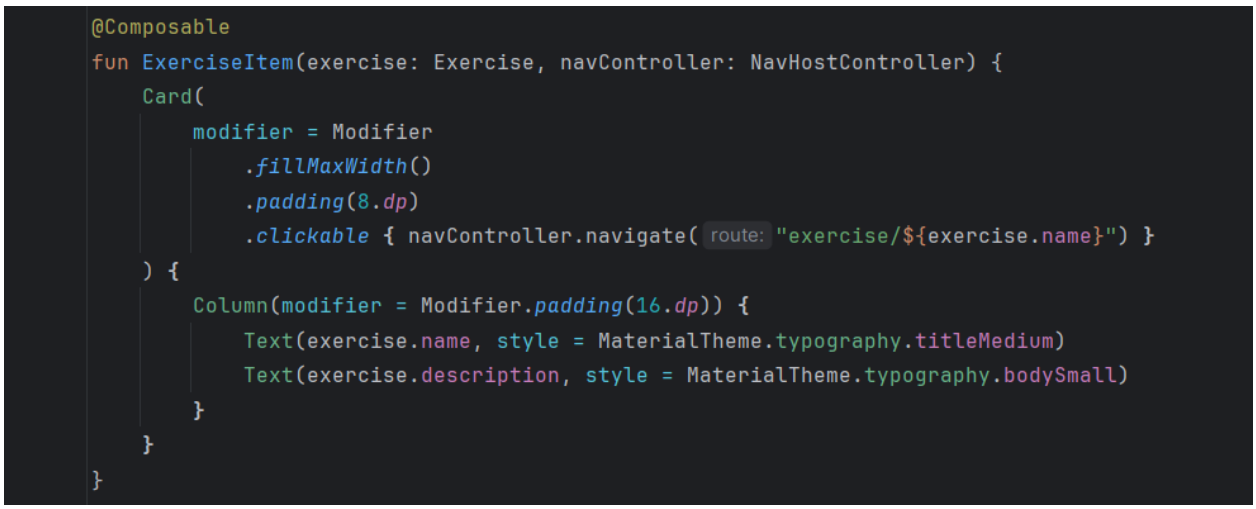

Рисунок 13 – Карточка упражнения

После необходимо было создать экран деталей упражнения, на котором будет описываться упражнение (рисунок 14), показываться правильная техника выполнения упражнения с помощью GIF-анимации (рисунок 15), а также будет находиться таймер для отслеживания времени во время выполнения упражнения, который представлен на рисунке 16.

```
fun ExerciseDetailScreen(exercise: Exercise) {
    var timeLeft by remember { mutableStateOf( value: 30) }
    var isRunning by remember { mutableStateOf( value: false) }
    val coroutineScope = rememberCoroutineScope()

    // Получаем GIF по названию упражнения
    val gifResId = when (exercise.name) {
        "Отжимания" -> R.drawable.pushups
        "Приседания" -> R.drawable.squats
        "Планка" -> R.drawable.plank
        "Подтягивания" -> R.drawable.pullups
        else -> R.drawable.default_gif
    }

    Scaffold(
        topBar = { TopAppBar(title = { Text(exercise.name) }) },
        containerColor = Color.Transparent
    ) { padding ->
        Column(
            modifier = Modifier
                .fillMaxSize()
                .padding(padding)
                .padding(16.dp),
            verticalArrangement = Arrangement.Center
        ) {
            Text(exercise.description, style = MaterialTheme.typography.bodyLarge)
            Spacer(modifier = Modifier.height(20.dp))
```

Рисунок 14 – Детали и описание упражнения

```
        // Используем Coil для загрузки GIF
        AsyncImage(
            model = gifResId,
            contentDescription = exercise.name,
            modifier = Modifier
                .fillMaxWidth()
                .height(200.dp)
        )

        Spacer(modifier = Modifier.height(20.dp))
```

Рисунок 15 – Вывод GIF-анимации

```
    // Таймер
    Text(
        text = if (timeLeft > 0) "Осталось: $timeLeft сек" else "Время вышло!",
        style = MaterialTheme.typography.headlineMedium
    )
    Spacer(modifier = Modifier.height(20.dp))

    Row {
        Button(onClick = {
            if (!isRunning) {
                isRunning = true
                coroutineScope.launch {
                    while (isRunning && timeLeft > 0) {
                        delay( timeMillis: 1000L)
                        timeLeft -= 1
                    }
                }
            }
        }) {
            Text( text: "Старт")
        }
        Spacer(modifier = Modifier.width(10.dp))
        Button(onClick = { isRunning = false }) {
            Text( text: "Пауза")
        }
    }
}
```

Рисунок 16 – Вывод таймера

3.2 Тестирование приложения

«Цель тестирования заключается в проверке корректности работы всех функций приложения» [9], выявлении возможных ошибок и обеспечении удобства взаимодействия пользователя с системой. Особое внимание уделяется ключевым действиям, таким как добавление и удаление упражнений, навигация между экранами и работа с профилем пользователя. Каждое действие тестируется на соответствие ожидаемому результату, чтобы гарантировать стабильность и надежность приложения.

При добавлении новых упражнений важно убедиться, что все введенные пользователем данные (название, описание, количество подходов, повторений и время выполнения) сохраняются корректно и отображаются в интерфейсе без искажений и необходимо проверить, что система корректно обрабатывает ошибки ввода (например, пустые поля).

При удалении упражнения необходимо проверить, что оно полностью исчезает из списка тренировок и не вызывает ошибок при последующем использовании приложения, а также следует убедиться, что изменения сохраняются после перезапуска приложения.

Навигация является ключевым аспектом удобства использования приложения, нужно проверить, что переходы между экранами выполняются быстро и без сбоев, а интерфейс обновляется корректно.

Изменения данных профиля должны сохраняться мгновенно и отображаться в интерфейсе без необходимости перезагрузки приложения и проверить, что система корректно обрабатывает ввод недопустимых значений (например, отрицательный вес или рост).

Результаты функционального тестирования приведены в таблице 4.

Таблица 4 – Результаты тестирования

Действие	Ожидаемый результат	Действительный результат
Добавление новых упражнений	Упражнение успешно добавляется в список тренировок, отображается в интерфейсе с корректными данными (название, описание, параметры)	Соответствует
Удаление созданных упражнений	Выбранное упражнение удаляется из списка тренировок, изменения сохраняются, упражнение больше не отображается	Соответствует
Навигация между экранами	Пользователь может легко переходить между экранами (главная страница, профиль, тренировки) без задержек или сбоев	Соответствует
Редактирование профиля пользователя	Измененные данные (имя, рост, вес) сохраняются и отображаются в профиле после обновления	Соответствует
Сохранение данных	Информация сохраняется при перезапуске приложения	Соответствует

Тестирование показало, что приложение работает стабильно, а все ключевые функции выполняются корректно. Ожидаемые результаты совпадают с действительными, что подтверждает готовность приложения к выпуску.

Приложение работает корректно и реализовывает весь функционал (Рисунки 17 - 26).

Рисунок 17 – Страница входа

Рисунок 18 – Профиль

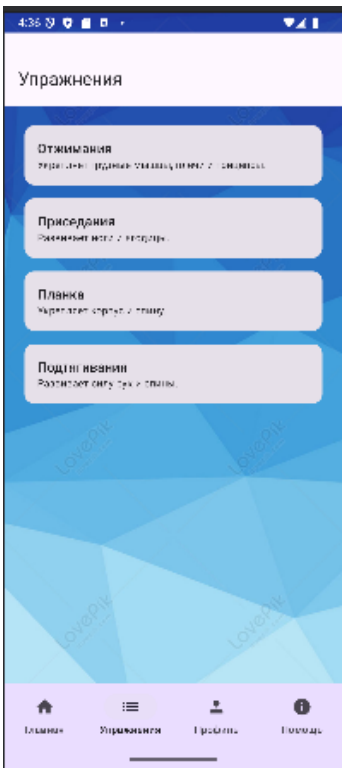

Рисунок 19 – Главная страница с упражнениями

Рисунок 20 – Страница с помощью

Рисунок 21 – Страница с первым упражнением

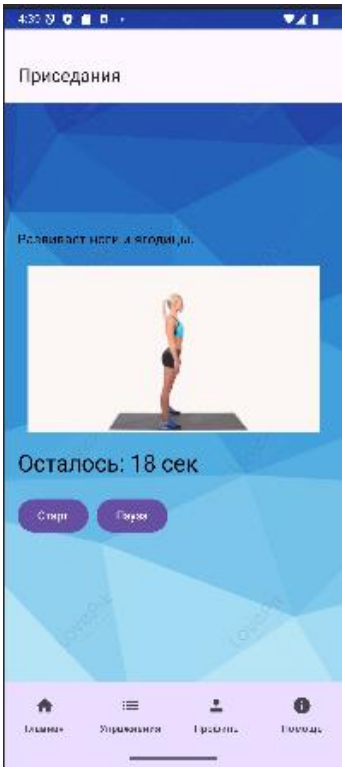

Рисунок 22 – Страница со вторым упражнением

Рисунок 23 – Страница с третьим упражнением

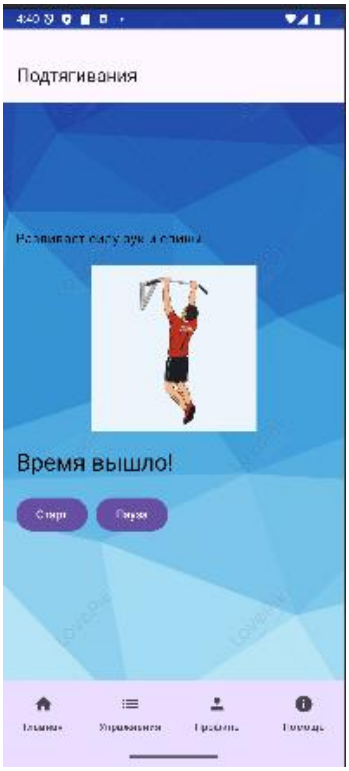

Рисунок 24 – Страница четвертым упражнением

Рисунок 25 – Страница с добавлением и удалением пользовательских тренировок

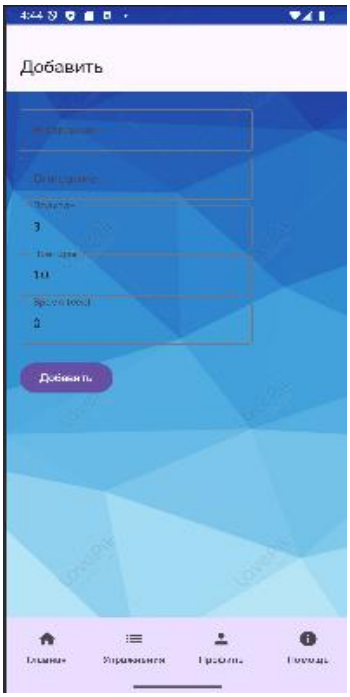

Рисунок 26 – Страница с добавлением, редактированием пользовательских упражнений

После завершения разработки приложения был сформирован APK-файл для тестирования его функциональности на физических устройствах.

APK представляет собой формат файлов, используемый для распространения и установки приложений на платформе Android. Данный файл содержит весь программный код, необходимые библиотеки, ресурсы и метаданные, которые требуются для корректной работы программы. Это универсальный способ упаковки, позволяющий пользователям легко устанавливать и запускать приложения на своих устройствах. Процесс компиляции можно наблюдать на рисунке 27.

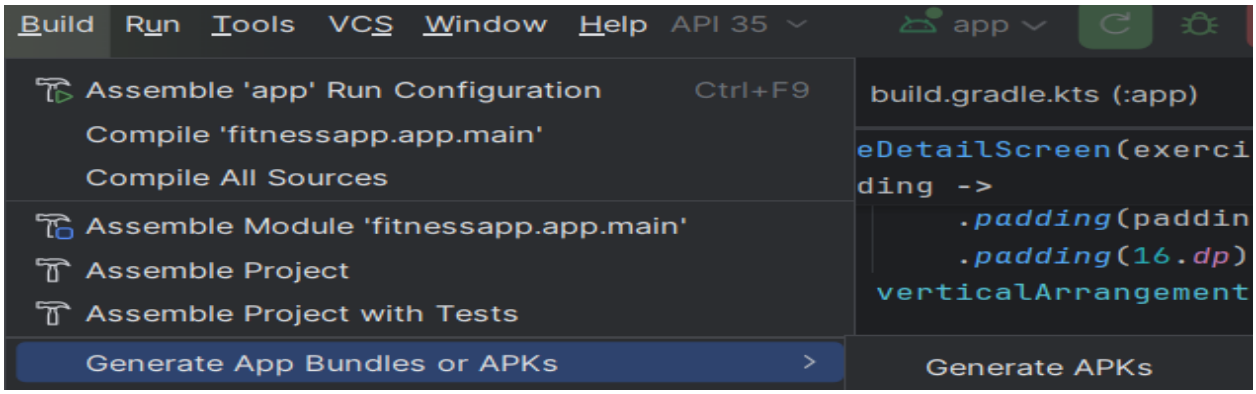

Рисунок 27 – Создание APK

После завершения сборки проекта в его директории будет сгенерирован файл, готовый для установки на физические устройства. Этот файл также можно использовать для публикации приложения в онлайн-магазинах, таких как Google Play. (Рисунок 28)

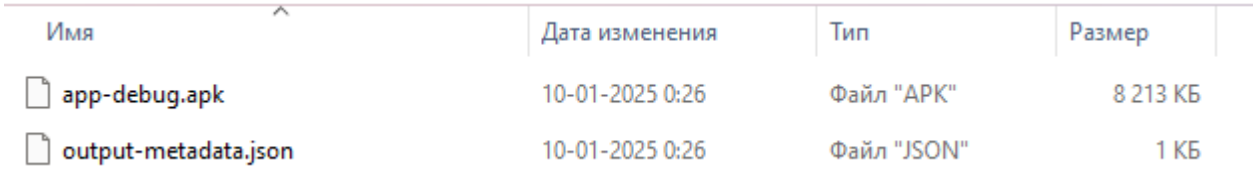

Рисунок 28 – Сгенерированный APK-файл

Приложение протестировано на различных устройствах, и во всех случаях оно функционировало корректно, обеспечивая выполнение всех заложенных функций.

Выводы по третьей главе. Результатом работы, выполненной в третьей главе данного исследования, было успешно разработано мобильное приложение для занятий фитнесом с использованием Android Studio и языка программирования Kotlin. Приложение включает в себя все ключевые функции, необходимые для эффективного управления тренировками: экран со списком упражнений, возможность добавления и редактирования упражнений, а также персонализированный профиль пользователя. Для хранения данных, таких как параметры тренировок и информация о пользователе, был выбран современный подход с использованием DataStore, дополненный механизмом JSON-сериализации. Это решение обеспечивает надежное и быстрое сохранение информации, а также удобство её обработки и извлечения.

Чтобы гарантировать стабильность и корректность работы приложения, были проведены комплексные тесты всех его функций. Тестирование подтвердило, что основные операции выполняются без ошибок, а приложение демонстрирует устойчивую работу в различных сценариях использования. Особое внимание уделялось проверке взаимодействия с хранилищем данных, навигации между экранами и сохранении списка упражнений после перезапуска приложения.

Завершающим этапом разработки стала успешная сборка APK-файла, который был установлен и протестирован на реальных мобильных устройствах. Результаты тестирования показали, что приложение работает без сбоев, а его интерфейс остается интуитивно понятным и удобным для пользователей. Это подтверждает готовность программного продукта к практическому применению в повседневной жизни.

Заключение

В рамках выполнения данной бакалаврской работы было спроектировано и разработано мобильное приложение для занятий фитнесом. Основной задачей проекта стало создание удобного инструмента, который поможет пользователям эффективно планировать тренировки, создавать свои собственные и достигать, таким образом, своих спортивных целей, при этом приложение адаптировано для интеграции в платформу, что расширяет возможности взаимодействия пользователей.

На начальном этапе работы была проведена детальная проработка предметной области, а также проанализированы особенности деятельности компании, для которой разрабатывался данный продукт. Для наглядного представления архитектуры и логики работы системы было выбрано подходящее CASE-средство для создания UML-диаграмм. Эти диаграммы позволили четко структурировать требования к будущему приложению и заложить основу для его дальнейшей разработки.

Процесс проектирования включал разработку архитектуры приложения, выбор технологий и создание пользовательского интерфейса. В качестве основной среды разработки была использована Android Studio, а для написания кода выбран язык программирования Kotlin. Интерфейс приложения был спроектирован с учетом современных принципов дизайна, таких как Material Design, чтобы обеспечить удобство использования и привлекательный внешний вид.

Реализация функционала потребовала решения ряда технических задач. Были разработаны механизмы для добавления, редактирования и удаления упражнений, а также система хранения данных о тренировках и параметрах пользователя. Для хранения информации использовался современный подход с применением DataStore и JSON-сериализации, что обеспечило надежность и быстродействие при работе с данными. Особое внимание уделялось

стабильности работы приложения и корректности всех вычислений, связанных с тренировками.

Проведенным тестированием была подтверждена работоспособность всех модулей приложения. Проверялась не только корректность выполнения операций, но и удобство взаимодействия с интерфейсом, тестирование проводилось на реальных устройствах, что позволило убедиться в высокой производительности и надежности приложения. Результаты показали, что оно успешно справляется с поставленными задачами и готово к практическому использованию.

Разработанное мобильное приложение представляет собой законченный продукт, который может быть использован для эффективного управления тренировками. При создании особое внимание уделялось удобству использования и простоте интерфейса, чтобы сделать приложение доступным не только для профессионалов, но и для новичков. В перспективе возможно дальнейшее развитие продукта за счет добавления новых функций, таких как интеграция с носимыми устройствами (например, смарт-часами), расширенная аналитика прогресса или возможность устраивать челленджи с другими пользователями. Но уже в текущем виде приложение полностью соответствует поставленным целям и может стать полезным инструментом для пользователей, стремящихся улучшить свою физическую форму.

Список используемой литературы и используемых источников

1. Ахметов А. К. Операционная система Android: история создания и развития. Разработка приложений для платформы Android [Электронный ресурс]. URL: https://cyberleninka.ru/article/n/operatsionnaya-sistema-android-istoriya-sozdaniya-i-razvitiya-razrabotka-prilozheniy-dlya-platformy-android (дата обращения: 15.03.2025).
2. Анализ качества case-средств для функционального моделирования систем //Информационно- аналитические и интеллектуальные системы для производства и социальной сферы / А. В. Гузилов, К. С. Мышенков, М. Ф. Симонов. – 2021. – 29 с.
3. Анализ современных платформ для разработки мобильных приложений / А. Р. Иванова, Д. К. Петров. – 2020. – 124 с.
4. Бурнет Э. Привет, Android! Разработка мобильных приложений. 2- е издание. – СПб.: Издательство «Питер». – 2016. – 256 с.
5. Вендров А. М. CASE-технологии: практическая работа в Rational Rose. – 1998. – 176 с.
6. Методологии и стандарты разработки требований к программным системам / М. В. Сидоров, Л. А. Кузнецова. – 2018. – 168 с.
7. Моделирование бизнес-процессов: учебное пособие / А.Н. Байдаков, О.С. Звягинцева, А.В. Назаренко. – 2017. – 179с.
8. Моделирование бизнес-процессов: учебник и практикум для вузов / О. И. Долганова, Е. В. Виноградова, А. М. Лобанова. – 2023. – 322 с.
9. Тестирование программного обеспечения / С. С. Куликов, Г. В. Данилова, О. Г. Смолякова, М. М. Меженная. – 2019. – 277 с.
10. Федоров Н. В. Проектирование информационных систем на основе современных CASE-технологий. – 2008. – 278 с.
11. UML 2.0: объектно-ориентированное моделирование и

разработка / Г. Буч, Д. Рамбо , И. Якобсон. – 2007. – 544 с.

12. A Comparative Study: Java Vs Kotlin Programming In Android Application Development / Subham Bose, Aditi, Madhuleena Mukherjee, Madhurima Banerjee – 2018. – 5 с.

13. Android Studio [Электронный ресурс]. URL: https://developer.android.com/studio (дата обращения: 05.04.2025).

14. Are you still smelling it? A comparative study between Java and Kotlin language / Matheus Flauzino, Júlio Veríssimo, Ricardo Terra, Elder Cirilo, Vinicius H. S. Durelli, Rafael S. Durelli. – 2018. – 10 с.

15. Comparative Analysis of Python and Java for Beginners / Mrs. Selina Khoirom, Moirangthem Sonia, Borishphia Laikhuram, Jaeson Laishram, Tekcham Davidson Singh – 2020. – 24 с.

16. Craig Larman Applying UML and Patterns: An Introduction to Object-Oriented Analysis and Design and Iterative Development – 2004. – 736 с.

17. Database Systems: Design, Implementation, & Management / Carlos Coronel, Steven Morris – 2018. – 816 с.

18. Insights into Layout Patterns of Mobile User Interfaces by an Automatic Analysis of Android Apps / Alireza Sahami Shirazi, Niels Henze, Albrecht Schmidt, Robin Goldberg, Benjamin Schmidt, Hansjorg Schmauder – 2013. – 10с.

19. Java and Kotlin, a performance comparison / Niclas Everlönn and Stylianos Gakis – 2020. – 70 с.

20. Tips and Tools for Killer GUIs / Joshua Marinacci, Chris Adamson, Swing Hacks. – 2017. – 542 с.