МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное бюджетное образовательное учреждение высшего образования

«ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

Факультет вычислительной техники

Кафедра «Вычислительная техника»

Направление подготовки: 09.03.01 «Информатика и вычислительная техника»

Профиль подготовки: «Программное обеспечение средств вычислительной техники и автоматизированных систем»

БАКАЛАВРСКАЯ РАБОТА

на тему

МОБИЛЬНОЕ ПРИЛОЖЕНИЕ ДЛЯ ОБУЧЕНИЯ ПРОГРАММИРОВАНИЮ

Студент 03.06.2025 Пронин Илья Геннадьевич
(подпись, дата) (ФИО полностью)

Руководитель 04.06.25 Слепцов Н.В.
(подпись, дата) (фамилия, инициалы)

Нормоконтролёр 04.06.2025 Бычков А.С.
(подпись, дата) (фамилия, инициалы)

Работа допущена к защите (протокол зас. кафедры от 28.05.2025 г. № 16)

Заведующий кафедрой М.А. Митрохин
(подпись) (фамилия, инициалы)

Работа защищена с отметкой хорошо (протокол заседания ГЭК от 6 июня 2025г. № 15)

Секретарь ГЭК Е.И. Гурин
(подпись) (фамилия, инициалы)

Пенза 2025

Федеральное государственное бюджетное образовательное учреждение
высшего образования
«ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

Факультет
вычислительной техники

Кафедра
«Вычислительная техника»

«Утверждаю»
Заведующий кафедрой
М.А. Митрохин
«06» 05 2025 г.

ЗАДАНИЕ
НА БАКАЛАВРСКУЮ РАБОТУ

1. Студент Пронин Илья Геннадьевич гр. 21ВВП2 факультета ФВТ

2.Тема работы Мобильное приложение для обучения программированию

Тема утверждена приказом ПГУ ПГУ № 601/о от 06.05.2025 г.

3. Руководитель бакалаврской работы к.т.н., доцент кафедры ВТ Слепцов Н.В.,

4. Задание на работу (назначение разработки, исходные данные и т.п.) Разработать мобильное приложение для обучения программированию “УчимПрограммирование”. Функционал приложения – предоставление пользователям возможность изучать программирование: пользователь должен иметь возможность выбора языка программирования, изучение курсов и отдальных уроков по выбранному языку программирования, прохождение итоговых тестов после завершения курса и получение результата; управление профилем: пользователь должен иметь возможнось добавления в избранное понравившихся курсов, иметь возможность просматривать пройденные тесты с возможностью повторного прохождения; профиль администратор: пользователь должен иметь возможность добавлять свои языки программирования, курсы, уроки и тесты; регистрация и аутентификация: для доступа к персонализированным функциям клиенту необходима регистрация и вход в систему.

СУБД – SQLite, а все приложение должно быть реализовано на языке Kotlin в среде Android Studio.

5. Перечень подлежащих разработке вопросов:

Произвести анализ предметной области: выявить достоинства и недостатки сущетсвующих android приложений в сфере обучения программированию, сделать выводы, поставить итоговую задачу. Изучить работу с выбранной средой разработки. Подключить БД и программу администрирования СУБД. Выбрать и реализовать наиболее подходящую архитектуру, разработать и протестировать итоговое android приложение.

6. Календарный график выполнения работы

№ п/п	Наименование этапов работы	Объем работы	Срок выполнения	Подпись руководителя
1	Изучение предметной области	32	06.05.2025 – 08.05.2025	
2	Анализ требований	32	09.05.2025 – 11.05.2025	
3	Создание макета графического интерфейса	30	12.05.2025 – 14.05.2025	
4	Выбор технологий, применяемых при проектировании и разработке	40	15.05.2025 – 18.05.2025	
5	Реализация графического интерфейса	66	19.05.2025 – 22.05.2025	
6	Разработка основной части программы	70	23.05.2025 – 27.05.2025	
7	Оформление пояснительной записки	54	28.05.2025 – 30.05.2025	

Дата выдачи задания "06" мая 2025 г.

Руководитель бакалаврской работы _____ 06.05.2025 Слепцов Н.В.
(подпись, дата)

Задание к исполнению принял студент _____ 06.05.2025 Пронин И.Г.
(подпись, дата)

Работу к защите допустить
Декан факультета _____ 06.06.2025 Фионова Л.Р.
(подпись, дата)

Реферат

Бакалаврская работа содержит 121 листов, 38 рисунков, 9 таблиц, 7 источников и 3 приложений.

ANDROIDПРИЛОЖЕНИЕ, ЛОГИРОВАНИЕ, БАЗА ДАННЫХ, ПАРАЛЛЕЛЬНОЕ ПРОГРАММИРОВАНИЕ, ANDROIDSTUDIO, KOTLIN, SQLITE

Цель работы – разработка мобильное приложение для обучения программированию “УчимПрограммирование” и предоставления статистики о пройденный курсах, тестах.

Для достижения поставленной цели было разработано androidприложение, отвечающее следующим поставленным задачам: предоставление пользователю возможность выбора языка программирования для обучения, выбор курсов и уроков в выбранной области, решение тестов и получение результата, просмотр истории пройденного материала, добавление курсов в избранное, добавление своих курсов и тестов.

В результате проведенной работы разработано мобильное приложение “УчимПрограммирование” предоставляющее пользователю возможность изучать языки программирования, проходить тесты, просматривать статистику, добавлять понравившиеся курсы в избранное, добавлять свои курсы и тесты.

Содержание

Введение 7
1 Мобильное приложение для обучения программированию 8
1.1 Анализ предметной области 8
1.1.1 Актуальность разработки 8
1.1.2 Анализ существующих аналогов 8
1.1.3 Вывод 13
1.1.4 Постановка задачи 14
1.2 Выбор технологий и средств разработки приложения 21
1.2.1 Выбор среды разработки 22
1.2.2 Kotlin 23
1.2.3 SQLite 23
1.2.4 Gradle 24
1.3 Проектирование информационной системы 26
1.3.1 Архитектура приложения 27
1.3.2 Проектирование базы данных 28
1.4 Разработка приложения 31
1.4.1 Реализация БД 32
1.4.2 Описание программы 36
1.4.3 Руководство пользователя 51
1.4.4 Тестирование и отладка 64
2 Охрана труда 67
Заключение 71
Список использованных источников 72
Приложение А – UML-диаграммы 73
Приложение Б – Листинг программы 77
Приложение В – Презентация 116

Перечень сокращений и обозначений

В настоящей работе применяют следующие сокращения и обозначения

БД – база данных

СУБД – система управления базами данных

API (ApplicationProgrammingInterface) – описание способов, которыми одна компьютерная программа может взаимодействовать с другой программой

FK (Foreign key) – внешний ключ, понятие теории реляционных баз данных, относящееся к ограничениям целостности базы данных

ID (Identifier) – уникальный номер объекта

IDE (Integrateddevelopmentenvironment) – интегрированнаясредаразработки

JSON (JavaScriptObjectNotation) – текстовыйформатобменаданными

PK (Primary key) – первичный ключ, понятие теории реляционных баз данных, относящееся к ограничениям целостности базы данных

SQL (StructuredQueryLanguage) – язык структурированных запросов

UML (UnifiedModelingLanguage) – унифицированныйязыкмоделирования

Введение

С каждым годом сфера ITнабирает все большую популярность, люди все больше и больше интересуются информационными технологиями, в школах выделяется больше часов на изучение информатики, появляются школы с углубленным изучением информатики, но большинство людей всеровно предпочитают электроное обучение – очному. Это не удивительно, так как благодаря современным технологиям есть возможность обучаться дистанционно, в любое время, из любой точки мира.

В последние годы наблюдается стремительный ростпопулярности мобильныхприложений для обучения программированию. Пользователи предпочитают иметь возможность обучаться в свободное время, например в транспорте или в перерыве на работе. Так же пользователи предпочитаю сразу получать результаты пройденных тестов и в случае совершения ошибок сразу получать верные ответы. В последнее время учителя в школах тоже отдают предпочтения специальным приложениям для проведения тестирования, вместо бумажных тестов.

Данный дипломный проект посвящен разработке мобильного приложения для обучения программированию, которое позволит облегчить обучение, повысит комфорт пользователя и поможет учителям школ в проведении тестирования. Целью работы является создание функционального, удобного и интуитивно понятного приложения, отвечающего современным требованиям разработки androidприложений.

1 Мобильное приложения для обучения программированию

1.1 Анализ предметной области

1.1.1 Актуальность разработки

В настоящее время выдающееся число людей начало интересоваться программированием и большая их часть предпочитают электронное обучение - очному. Это не удивительно, так как благодаря современным информационным технологиям есть возможность обучаться дистанционно, в любое время и из любой точки мира.

Еще одним большим преимуществом приложений для обучения программированию перед очными курсами - стоимость и самостоятельный темп обучения. Вы можете выполнять задания в свободные минуты, например в транспорте или во время перерыва на работе/учебе.

Востребованность приложений для обучения программированию остается высокой и будет продолжать расти в будущем. Это обусловлено растущим спросом на IT-специалистов, доступностью и гибкостью онлайн-обучения, экономической привлекательностью, удобством использования и наличием поддержки и сообщества.

1.1.2 Анализ существующих аналогов

Приложения для обучения программированию — это программные инструменты, разработанные для помощи людям в изучении и освоении навыков программирования на различных платформах (мобильные устройства, веб-браузеры, десктопные компьютеры). Они предлагают интерактивные уроки, упражнения, задачи и проекты, чтобы сделать процесс обучения более доступным, интересным и эффективным. На рынке существует много различных приложений, которые позволяют изучать программирование в электронном виде. Эти приложения предоставляют доступ к изучению различных языков программирования, таких как: Java, C++, C#, Python и т.д.

Для более правильной реализации приложения, необходимо проанализировать аналоги, выявить их слабые и сильные стороны, а также определить основные требования к разрабатываемому приложению.

В этом пункте будут рассмотрены три самых популярных приложения для обучения программированию, а именно:

– мобильное приложение Sololearn;

– мобильное приложение Mimo;

– мобильное приложение CodeCademy Go.

Мобильное приложение Sololearn.

SoloLearn — это популярное приложение для начинающих и опытных программистов с широким выбором курсов по различным языкам программирования.

Интерфейс мобильного приложения Sololearn представлен на рисунках 1, 2.

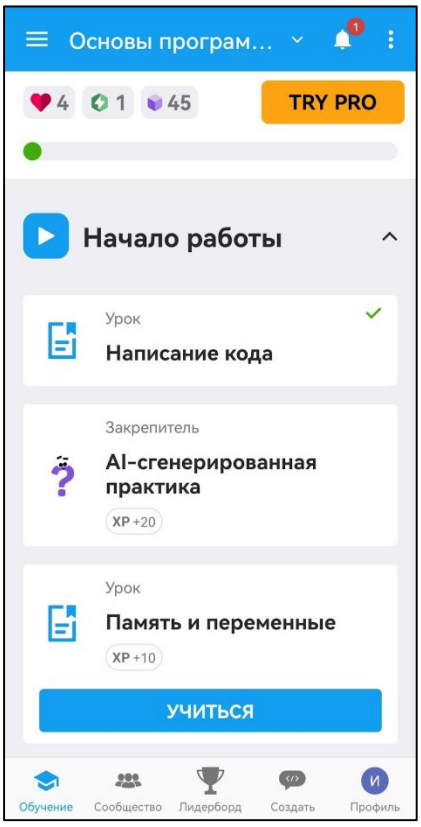


Рисунок 1 – Главная страница сервиса

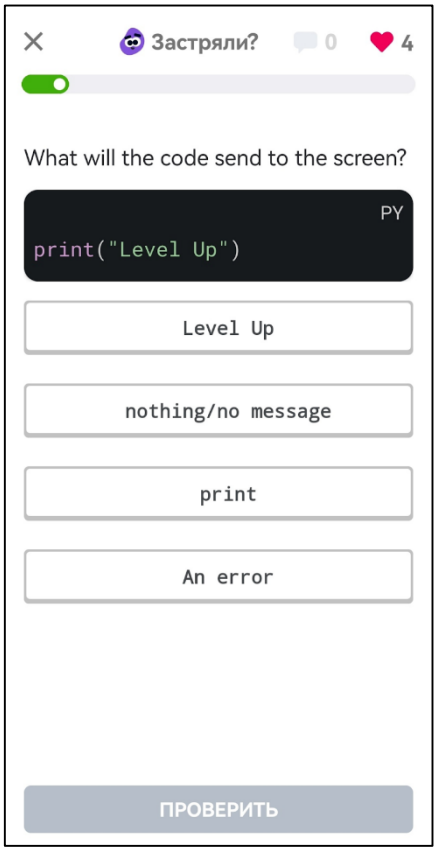


Рисунок 2 –Интерфейс тестовых заданий

Мобильное приложение Sololearn имеет большое колличество различных функций, предоставляемых пользователю. Ниже представлены основные из них:

– возможность обучения программированию. У пользователя есть выбор, какой язык программирования изучать;

– синхронизация. Приложение позволяет синхронизировать прогресс между разными устройствами;

– сообщество. В приложении вы можете общаться с другими учениками;

– лидерборд. За выполнение заданий вы получаете очки и соревнуетесь с другими учениками в специальных Лигах;

– разбор ошибок. Приложение разбирает ваши ошибки.

Преимущества приложения Sololearn:

– поддерживает русский язык;

– приложение подробно разбирает ваши ошибки;

– наличие смс-уведомлений, напоминающих о новых доступных заданиях;

– понятный интерфейс.

Недостатки приложения Sololearn:

– проблемы с производительностью;

– ограниченные функции при бесплатной версии;

– система “сердец”, где за каждую ошибку вы теряете одну из пяти “жизней” которые восстанавливаются со временем;

– при потере всех пяти “жизней”, вы временно не можете продолжить обучение.

Мобильное приложение Mimo.

Mimo – приложение с короткими уроками и проектами, идеально подходит для тех, кто хочет учиться в своем ритме и постепенно осваивать Python, HTML или JavaScript.

Интерфейс мобильного приложения Mimo представлен на рисунках 3 и 4.

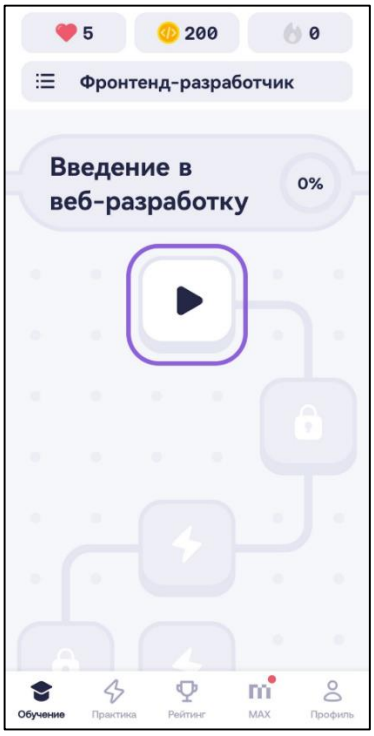


Рисунок 3 – Главная страница сервиса

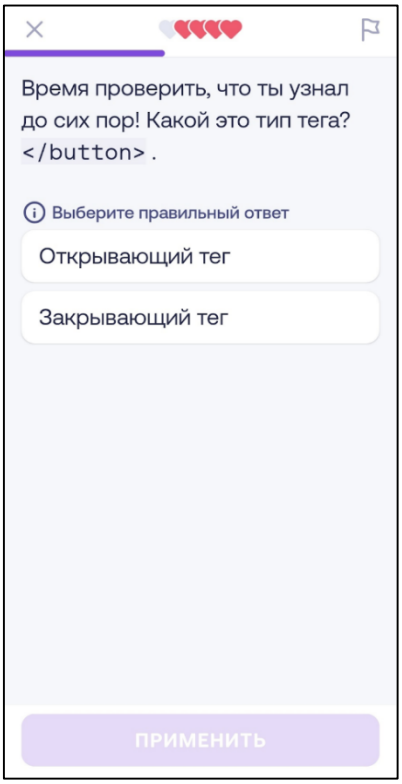


Рисунок 4 – Интерфейс заданий

Мобильное приложение Mimo имеет большое колличество различных функций, предоставляемых пользователю. Ниже представлены основные из них:

– возможность обучения программированию. У пользователя есть выбор, какой язык программирования изучать;

– особый подход. Перед обучением пользователь проходит опрос, где указывает свои пожелания и опыт в программировании;

– синхронизация. Приложение позволяет синхронизировать прогресс между разными устройствами;

– сообщество. В приложении вы можете общаться с другими учениками;

– разбор ошибок. Приложение разбирает ваши ошибки.

Плюсы мобильного приложения Mimo:

– поддерживает русский язык;

– особый подход к каждому пользователю;

– наличие полезных материалов по использованию приложения;

– приложение подробно разбирает ваши ошибки.

Минусы мобильного приложения Mimo:

– загруженность сервиса;

– необходимость платной подписки для полноценного обучения;

– не удобный интерфейс.

Мобильное приложение CodeCademy Go.

Codecademy Go — мобильное приложение, которое является сопутствующим ресурсом для онлайн-платформы Codecademy. Оно позволяет пользователям изучать концепции программирования в любом месте и в любое время.

Интерфейс мобильного приложения CodeCademy Go представлен на рисунках 5 и 6.

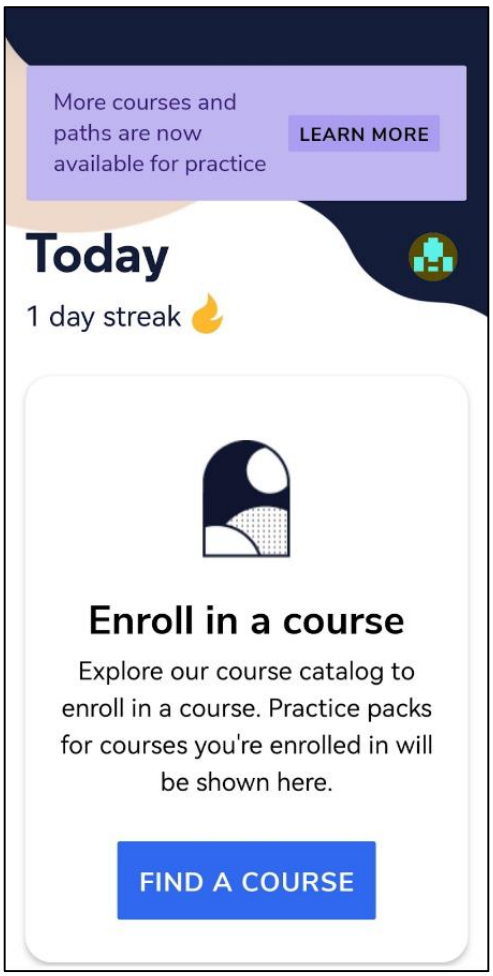


Рисунок 5 – Главная страница сервиса

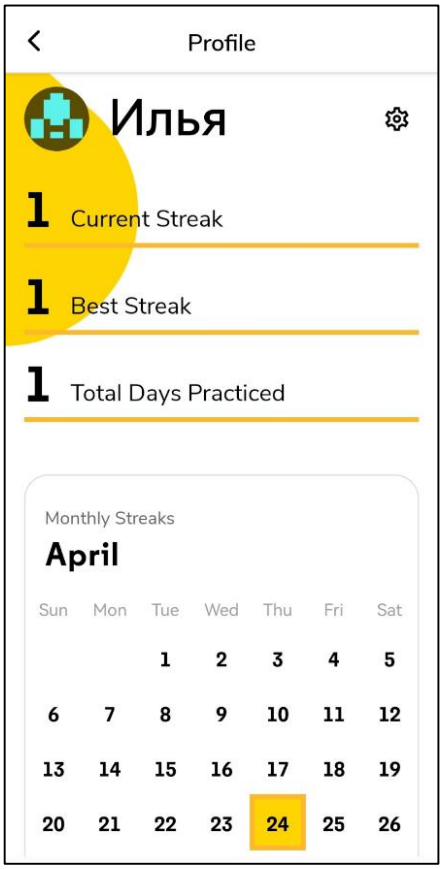


Рисунок 6 – Профиль пользователя

Функционал приложения Codecademy Go:

– выбор языка программирования для изучения;
– прохождение тестов на наиболее подходящий язык для изучения;
– синхронизация прогресса между мобильным приложением с сайтом;
– возможность отслеживать прогресс.

Преимущества Codecademy Go:

– система ежедневного входа;
– индивидуальных подход к выбору языка программирования;
– система оценок;
– возможность отслеживать прогресса.

Недостатки Codecademy Go:

– доступны только языки программирования купленные в браузерной версии;
– возможны проблемы с технической поддержкой;

– не поддерживает русский язык;

– ограничения бесплатной версии.

1.1.3 Вывод

Результаты анализа существующих аналогов можно представить в таблице 1, показывающие минусы и плюсы рассматриваемых сервисов.

Исходя из анализа аналогов следует отметить, что все они являются хорошими и конкурентоспособными приложениями для обучения программированию. У каждого есть свои плюсы и минусы, особые функции которые выделяются среди других похожих приложений.

Таблица 1 – Достоинства и недостатки мобильных приложений

Функционал	«SoloLearn»	«Mimo»	«CodeCademy Go»
Поддержка русского языка	+	+	-
Индивидуальный подход к пользователю	-	+	+
Простой и понятный интерфейс	+	+	-
Наличие бесплатных курсов	+	+	+
Расширенная статистика пользователя	-	-	+
Синхронизация прогресса между устройствами	+	+	+

Проанализировав данные сервисов, было решено разработать собственное приложение для обучения программированию, которое будет называться «УчимПрограммирование». Данное приложение будет соответствовать основным требованиям, которые предоставляет пользователь к сервису, а именно: удобный интерфейс; все основные функции, необходимые для комфортного обучения; возможность авторизованным пользователям просматривать доступные курсы.

Помимо базового функционала, будут реализованы функции, которые

отсутствуют у аналогов и которые будут привлекательны конечному пользователю. А именно будут реализованы системы: “Избранное” позволяющая пользователю сохранять понравившиеся темы для повторения в будущем и “Режим администратора” позволяющий добавлять свои темы, уроки и тесты.

Данный режим будет полезен например учителям в школах, во дистанционного обучения или во время подготовки итоговых тестов. Преимуществами «УчимПрограммирование» перед другими приложениями для обучения программированию будет заключаться в том, что пользователю будет доступна возможность обучаться на различных языках в своем темпе, а также данный функционал будет расширен своими интересными «фишками», которых нет в аналогах.

1.1.4 Постановка задачи

Приложение должно реализовывать следующий функционал:

– со стороны клиента:

1) регистрация и авторизация;
2) профиль пользователя;
3) отслеживание прогресса обучения;
4) доступ к разным языкам, курсам и урокам;
5) разнообразие задач;
6) объяснение решения;
7) избранное;
8) объяснение решения;
9) история пройденных тестов;

– минимальные системные требования:

1) ОС: Android 5.0 и выше;
2) минимальная скорость интернета: 1 МБит/Сек;
3) минимальный объем оперативной памяти: 2 ГБ;
4) минимальный объем памяти: 50 МБ.

Данное приложение имеет функционал, который разделен на три категории, а именно:

– для неавторизованного пользователя;

– для авторизованного пользователя;

– для авторизованного администратора.

Диаграмма вариантов использования приложения для неавторизованного пользователя представлена на рисунке 7.

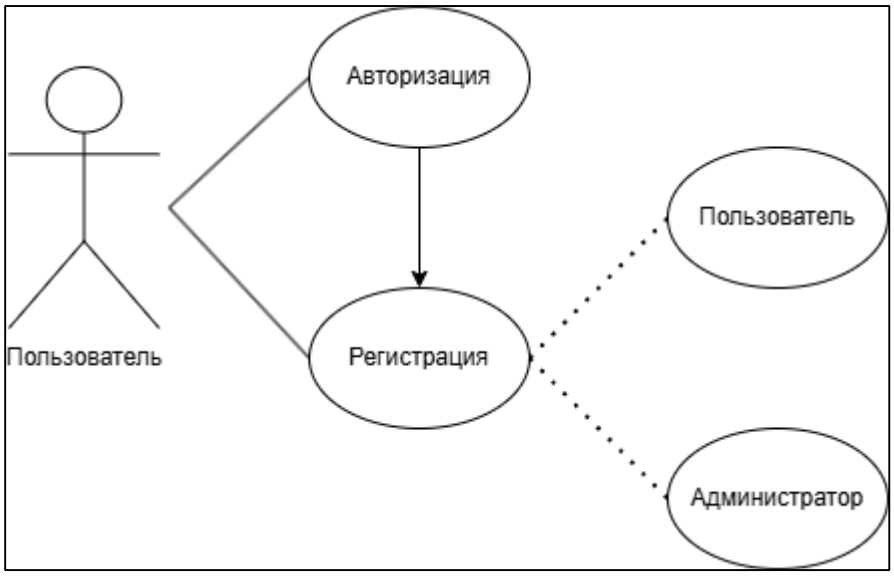


Рисунок 7 – UML диаграмма вариантов использования неавторизованного пользователя

Неавторизованный пользователь имеет возможность пользоваться ограниченным набором функций, именно:

– авторизация в приложении;

– регистрация профиля пользователя;

– регистрация профиля администратора.

Авторизованному пользователю будет доступен расширенный функционал работы программы. UML диаграмма представлена на рисунке 8.

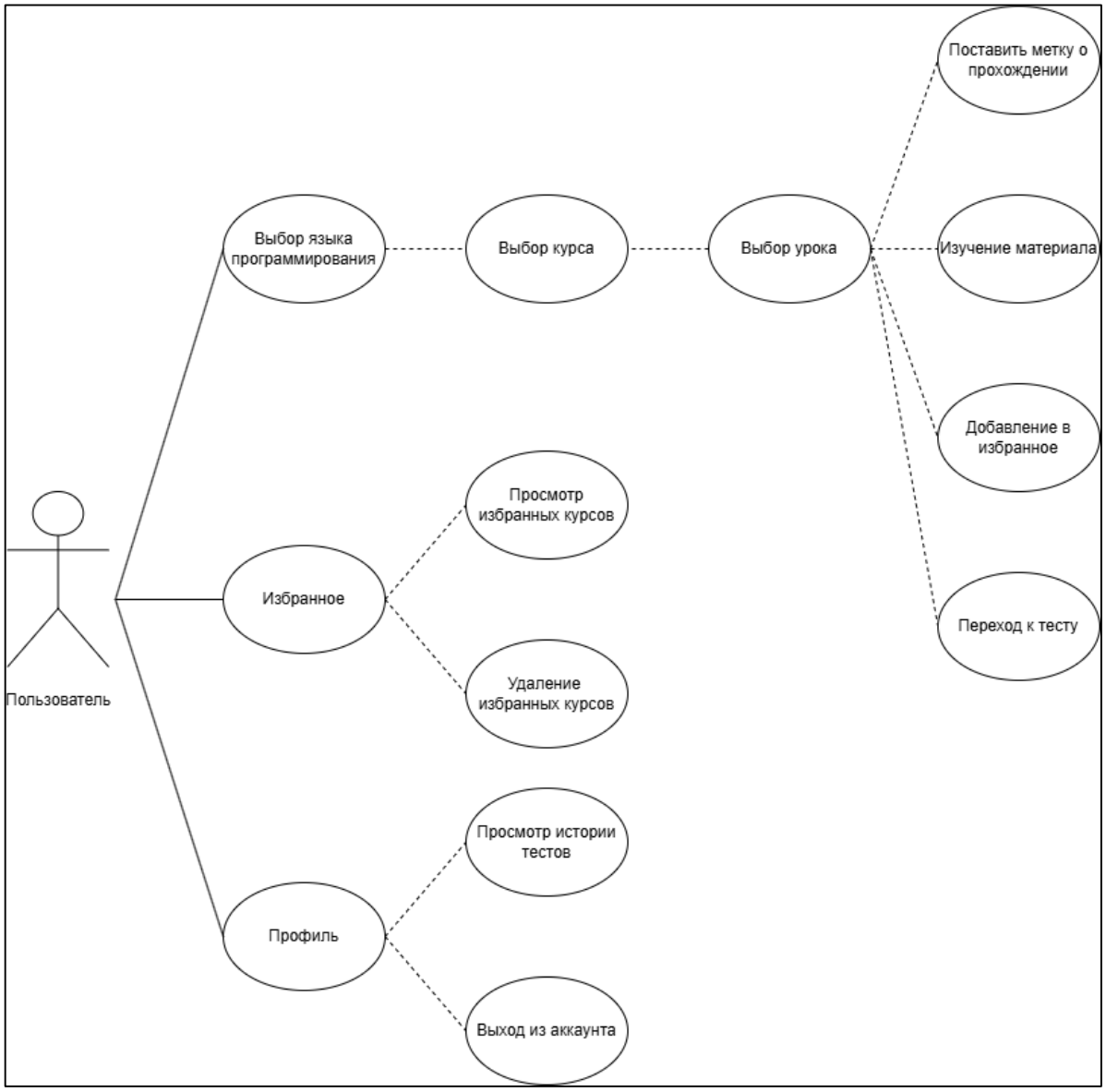


Рисунок 8 –UML диаграмма вариантов использования авторизованного пользователя

Функциональные возможности, доступные авторизованному пользователю:

- просмотр профиля;
- просмотр истории пройденных тестов;
- выход из аккаунта;
- добавление курсов в избранное;
- просмотр курсов в избранном;
- удаление курсов из избранного;
- выбор языка программирования;

– выбор курса;

– выбор урока;

– прохождение тестов.

Авторизованному администратору будет доступен расширенный функционал работы программы. UML диаграмма представлена на рисунке 9.

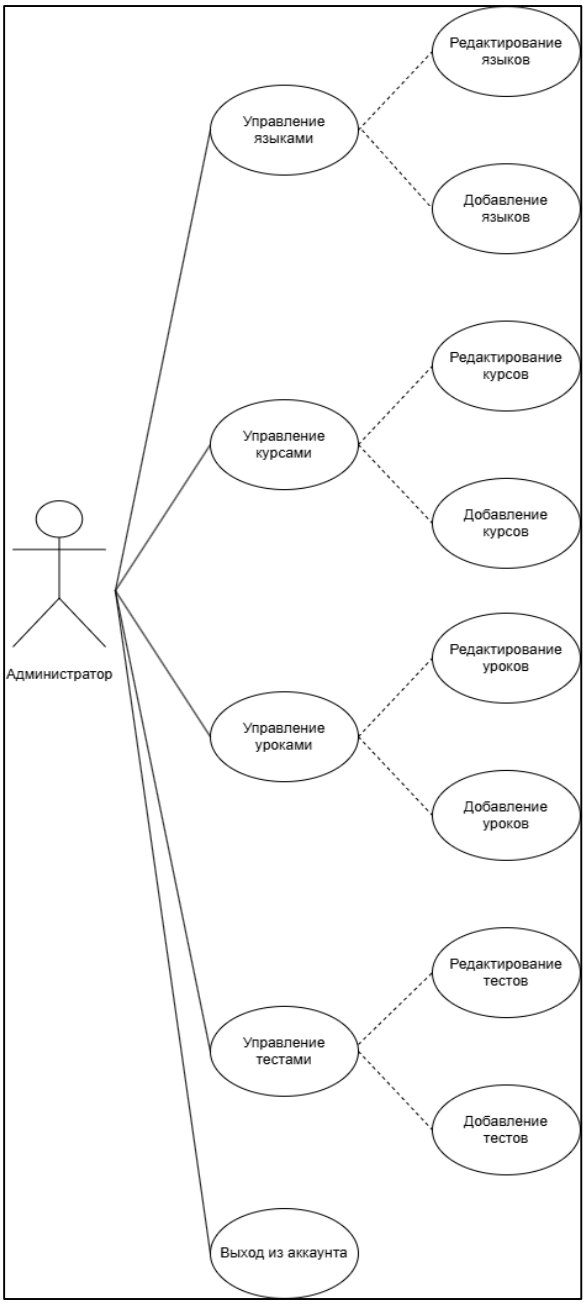


Рисунок 9 –UML диаграмма вариантов использования авторизованного администратора

Функциональные возможности, доступные авторизованному администратору:

- редактирование языков;
- добавление языка;
- редактирование курсов;
- добавление курса;
- редактирование уроков;
- добавление урока;
- редактирование тестов;
- добавление теста.

1.2 Выбор технологий и средств разработки приложения

В данном разделе будут рассмотрены технологии и средства разработки, которые были использованы для реализации мобильного приложения для обучения программированию «УчимПрограммирование».

Ниже приведены технологии и средства разработки, которые были выбраны:

- Android Studio;
- Kotlin;
- SQLite;
- Gradle;
- androidx.core:core-ktx;
- androidx.appcompat:appcompat;
- androidx.constraintlayout;
- Material Design Components.

1.2.1 Выбор среды разработки

Существует огромное колличество сред разработки (IDE) которые по своему хороши. Некоторые предоставляют более удобный интерфес для написания, тесторования и отладки кода, другие включают в себя более продвинутые инструменты для управления проектами, симуляторы устройств и

инструменты для автоматизированного тестирования. НапримерXcode - официальная IDE для разработки iOS-приложений. Поддерживает языки Swift и Objective-C, включает симулятор iOS-устройств и инструменты для тестирования. Xcode также предоставляет инструменты для дизайна интерфейсов, написания автоматизированных тестов и анализа производительности приложений. Так же существует популярна IDE от Microsoft – VisualStudio, которая поддерживает разработку кроссплатформенных приложений с использованием Xamarin. Visual Studio предоставляет множество инструментов для разработки, тестирования и отладки приложений, а также интеграцию с облачными сервисами и системами контроля версий. Но так как я разрабатываю Androidприложение я остановлю свой выбор на AndroidStudio. Использование Android Studio для разработки приложений под Android имеет множество преимуществ.

Android Studio — это официальный IDE для Android, поддерживаемый Google. Это гарантирует своевременные обновления, поддержку новых функций платформы и совместимость с последними версиями Android. Android Studio включает в себя все необходимые инструменты для разработки Android-приложений: Android SDK, инструменты сборки, эмуляторы и отладочные средства. Это упрощает настройку и ускоряет начало работы. Визуальный редактор интерфейсов (Layout Editor), позволяет создавать и редактировать UI с помощью drag-and-drop, а также сразу видеть предпросмотр на разных устройствах и версиях Android.Редактор поддерживает Java, Kotlin и XML, предоставляет интеллектуальное автодополнение, рефакторинг кода и мгновенный анализ ошибок, что повышает производительность и снижает количество ошибок [1,2].

1.2.2 Kotlin

Kotlin – это современный статически типизированный язык программирования, разработанный компанией JetBrains. Он полностью совместим с платформой Java и используется преимущественно для разработки приложений под Android, а также для серверной разработки, веб-приложений и многого другого. Kotlin предлагает более краткий и выразительный синтаксис по

сравнению с Java, что снижает объем шаблонного кода и упрощает чтение и поддержку программ. Так же Android Studio полностью поддерживает язык Kotlin. Это облегчает написание более краткого и безопасного кода [3,4].

Преимущества Kotlin:

– Kotlin полностью совместим с Java — можно использовать существующие Java-библиотеки и код прямо в Kotlin-проектах без дополнительных усилий;

– Kotlin поддерживает лямбда-выражения, функции высшего порядка, неизменяемые коллекции и другие функциональные концепции, что упрощает написание выразительного и гибкого кода;

– встроенная система типов предотвращает ошибки, связанные с null (например, NullPointerException), обеспечивая большую надежность и безопасность кода;

– Kotlin поддерживается JetBrains и Google, имеет большое сообщество и постоянно развивается, получая новые возможности и улучшения [5,6].

1.2.3 SQLite

В качестве СУБД была выбрана SQLite – это легковесная встраиваемая реляционная база данных, которая широко используется для хранения данных в приложениях, особенно в мобильных и встраиваемых системах.SQLite не требует отдельного серверного процесса или отдельной установки. Она представляет собой библиотеку, которую можно встроить прямо в приложение. Все данные хранятся в одном файле на диске.

Преимущества SQLite:

– SQLite является встроенной базой данных в Android, поэтому не требует дополнительной установки или настройки. Это упрощает процесс разработки и распространения приложений;

– SQLite занимает очень мало места и не требует запуска отдельного серверного процесса, что экономит ресурсы устройства и повышает производительность;

– SQLite позволяет хранить данные непосредственно на устройстве, что обеспечивает быстрый доступ к информации и работу приложения в оффлайн-режиме;

– работа с SQLite происходит локально, без необходимости подключения к серверу, что важно для мобильных приложений с ограниченным или отсутствующим интернетом.

1.2.4 Gradle

Gradle — это современная система автоматизации сборки проектов, широко используемая в разработке программного обеспечения, включая Android-приложения. Она позволяет автоматизировать процесс компиляции, тестирования, упаковки и деплоя программ.

Преимущества Gradle:

– Gradle – официальная система сборки для Android-проектов. Android Studio генерирует и использует Gradle-скрипты для управления сборкой приложений;

– Gradle управляет процессом сборки проекта — от компиляции исходного кода и обработки ресурсов до создания итогового файла приложения (APK, JAR и др.);

– Gradle использует свой собственный язык сценариев на основе Groovy или Kotlin DSL, что позволяет создавать сложные и настраиваемые процессы сборки под любые нужды;

– позволяет создавать разные варианты сборки (например, debug, release), а также конфигурации для разных устройств и целей.

1.2.5 Библиотека androidx.core:core-ktx

androidx.core:core-ktx – это библиотека расширений (KTX) для Android, которая предоставляет удобные и идиоматичные функции на языке Kotlin для стандартной библиотеки androidx.core. Она создана, чтобы сделать разработку Android-приложений на Kotlin более простой, лаконичной и выразительной.

1.2.6 Библиотека androidx.appcompat:appcompat

androidx.appcompat:appcompat- это одна из ключевых библиотек Android Jetpack, которая обеспечивает обратную совместимость новых функций пользовательского интерфейса Android с более старыми версиями операционной системы.

1.2.7 Библиотека androidx.constraintlayout

androidx.constraintlayout – это библиотека для Android, предоставляющая мощный и гибкий менеджер компоновки (Layout Manager) под названием ConstraintLayout. Она позволяет создавать сложные и адаптивные пользовательские интерфейсы с помощью системы ограничений (constraints) между элементами интерфейса.

1.2.8 MaterialDesignComponents

MaterialDesignComponents – это набор официальных библиотек и компонентов пользовательского интерфейса от Google, реализующих принципы и стили Material Design — дизайн-системы, разработанной Google для создания красивых, удобных и последовательных интерфейсов на разных платформах.

Преимущества MaterialDesignComponents:

– последовательность интерфейса. Компоненты следуют единым стандартам дизайна, что повышает удобство и узнаваемость приложений;

– адаптивность. Компоненты автоматически подстраиваются под разные размеры экранов и платформы;

– поддержка темизации. Легко изменять цвета, шрифты и стили, создавая уникальный внешний вид приложения;

– встроенные анимации и взаимодействия. Компоненты имеют плавные переходы и отклики на действия пользователя.

1.3. Проектирование информационной системы

1.3.1. Архитектура android приложения «УчимПрограммирование»

На рисунке 11 представлена архитектура создаваемого мобильного приложения «УчимПрограммирование».

Приложение построено на основе архитектурного шаблона Model-View-Controller (MVC) (рисунок 10):

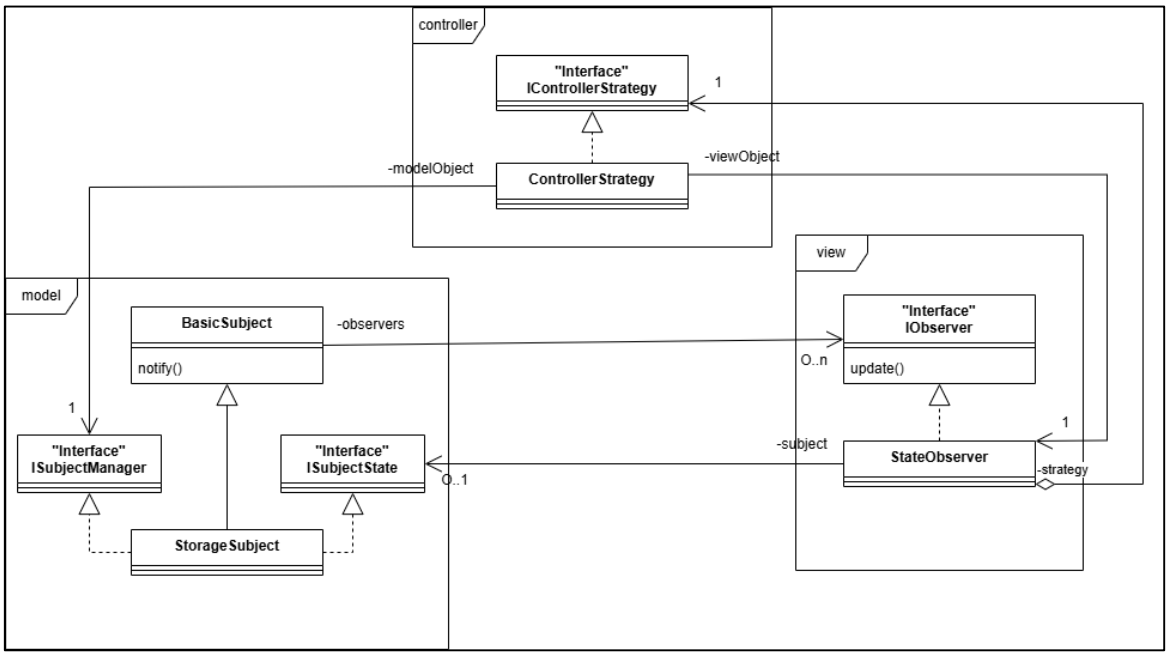


Рисунок 10 – Архитектура приложения

Model (Модель) - отвечает за управление данными и бизнес-логикой приложения. Хранит состояние приложения. Обрабатывает правила и операции над данными. Не зависит от пользовательского интерфейса и не знает о его реализации. Представлен классами данных и DatabaseHelper.

View (Представление) - отвечает за отображение данных пользователю. Показывает информацию, полученную из модели. Обеспечивает пользовательский интерфейс. Activity, Fragment и XML-макеты.

Controller (Контроллер) - посредник между Model и View. Обрабатывает пользовательский ввод (клики, ввод текста и т.п.). Обновляет модель на основе действий пользователя. Обновляет представление, чтобы отразить изменения в модели. Логика обработки пользовательского ввода в Activity/Fragment. Singleton Pattern - используется в AuthManager для управления единственным экземпляром менеджера аутентификации. Adapter Pattern - реализован в адаптерах RecyclerView для связывания данных с представлениями.

Пользователь взаимодействует с интерфейсом (View) - например, нажимает кнопку или вводит данные.View передаёт событие в Controller. Controller обрабатывает событие, обновляет данные в Model (например, сохраняет изменения). Model изменяет своё состояние и уведомляет View об обновлении данных. View обновляет отображение, показывая актуальные данные пользователю.

Преимущества такого разделения:

– чёткое разделение ответственности: каждый компонент отвечает за свою область;

– повышенная модульность: можно менять реализацию одного компонента без затрагивания других;

– упрощение поддержки и тестирования: легко тестировать бизнес-логику отдельно от UI;

– параллельная разработка: разработчики могут работать над моделью, дизайнерами — над представлением, а контроллеры обеспечивают связь между ними.

Топология системы и распределение ее компонентов по физическим узлам представлены на рисунке 11.

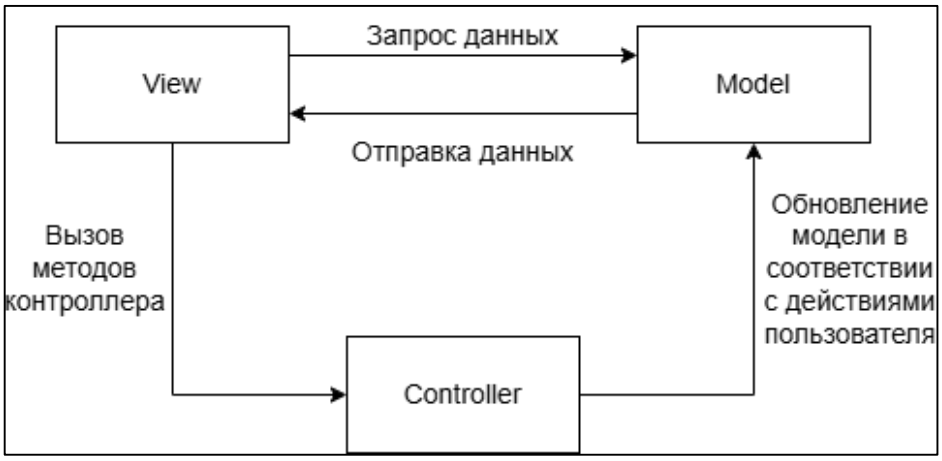


Рисунок 11 –диаграмма развертывания системы

1.3.3 Проектирование базы данных

База данных «УчимПрограммирование» спроектирована для хранения информации о пользователях, языках программирования, курсах, уроках, вопросов для тестирования, истории прохождения тестов и избранного.

Основные компоненты показаны в таблицах 2-8:

Таблица 2 – Атрибуты таблицы users(Пользователи системы):

№	Поле	Тип данных	Ограничения	Индексы	Описание
1	id	INTEGER	PRIMARY KEY, AUTOINCREMENT	PRIMARY	Уникальный идентификатор пользователя
2	username	TEXT	NOT NULL, UNIQUE	UNIQUE	Имя пользователя для входа в систему
3	email	TEXT	NOT NULL, UNIQUE	UNIQUE	Электронная почта пользователя
4	password	TEXT	NOT NULL	-	Хеш пароля пользователя
5	role	TEXT	DEFAULT 'user'	INDEX	Роль пользователя в системе

Таблица 3 – Атрибуты таблицы languager (Языки программирования)

№	Поле	Тип данных	Ограничения	Индексы	Описание
1	id	INTEGER	PRIMARY KEY, AUTOINCREMENT	PRIMARY	Уникальный идентификатор языка
2	name	TEXT	NOT NULL, UNIQUE	UNIQUE	Название языка программирования
3	description	TEXT	NOT NULL	-	Подробное описание языка
4	icon_res_id	TEXT	-	-	Идентификатор ресурса иконки в приложении

Таблица 4 – Атрибуты таблицы courses (образовательные курсы)

№	Поле	Тип данных	Ограничения	Индексы	Описание	Примеры значений
1	id	INTEGER	PRIMARY KEY	PRIMARY	Уникальный идентификатор курса	1, 2, 3...
2	language	INTEGER	FOREIGN KEY, NOT NULL	INDEX	Ссылка на язык программирования	1, 2, 3...
3	title	TEXT	NOT NULL	INDEX	Название курса	"Основы Python", "Java для начинающих"
4	descriptio	TEXT	NOT NULL	-	Детальное описание курса	"Изучите основы Python с нуля"
5	difficulty	TEXT	NOT NULL	INDEX	Уровень сложности курса	"Beginner", "Intermediate", "Advanced"

Таблица 5 – Атрибуты таблицы lessons (учебные уроки)

№	Поле	Тип данных	Ограничения	Индексы	Описание
1	id	INTEGER	PRIMARY KEY, AUTOINCREMENT	PRIMARY	Уникальный идентификатор урока
2	course_id	INTEGER	FOREIGN KEY, NOT NULL	INDEX	Ссылка на курс
3	name	TEXT	NOT NULL	INDEX	Название урока
4	content	TEXT	NOT NULL	-	Текстовое содержимое урока
5	order_number	INTEGER	NOT NULL	INDEX	Порядковый номер урока в курсе

№	Поле	Тип данных	Ограничения	Индексы	Описание
6	is_completed	INTEGER	DEFAULT 0	INDEX	Статус завершения урока (boolean)
7	youtube_video_id	TEXT	-	-	Идентификатор видео на YouTube

Таблица 6 – Атрибуты таблицы quiz_question (вопросы для тестирования)

№	Поле	Тип данных	Ограничения	Индексы	Описание
1	id	INTEGER	PRIMARY KEY, AUTOINCREMENT	PRIMARY	Уникальный идентификатор вопроса
2	lesson_id	INTEGER	FOREIGN KEY, NOT NULL	INDEX	Ссылка на урок
3	question_text	TEXT	NOT NULL	-	Текст вопроса
4	option_a	TEXT	NOT NULL	-	Первый вариант ответа
5	option_b	TEXT	NOT NULL	-	Второй вариант ответа
6	option_c	TEXT	NOT NULL	-	Третий вариант ответа
7	option_d	TEXT	NOT NULL	-	Четвертый вариант ответа
8	correct_answer	TEXT	NOT NULL	-	Правильный ответ (A, B, C, D)
9	explanation	TEXT	-	-	Объяснение правильного ответа

Таблица 7 – Атрибуты таблицы user_quiz_attempts

№	Поле	Тип данных	Ограничения	Индексы	Описание
1	id	INTEGER	PRIMARY KEY, AUTOINCREMENT	PRIMARY	Уникальный идентификатор попытки
2	user_id	INTEGER	FOREIGN KEY, NOT NULL	INDEX	Ссылка на пользователя
3	lesson_id	INTEGER	FOREIGN KEY, NOT NULL	INDEX	Ссылка на урок
4	score	INTEGER	NOT NULL	-	Количество правильных ответов
5	total_questions	INTEGER	NOT NULL	-	Общее количество вопросов в тесте
6	date_completed	TEXT	NOT NULL	INDEX	Дата и время завершения теста

Таблица 8 – Атрибуты таблицы favorites (избранные материалы)

№	Поле	Тип данных	Ограничения	Индексы	Описание
1	id	INTEGER	PRIMARY KEY, AUTOINCREMENT	PRIMARY	Уникальный идентификатор записи
2	user_id	INTEGER	FOREIGN KEY, NOT NULL	INDEX	Ссылка на пользователя
3	lesson_id	INTEGER	FOREIGN KEY	INDEX	Ссылка на урок
4	youtube_video_id	TEXT	-	-	ID видео YouTube
5	item_type	TEXT	NOT NULL	INDEX	Тип элемента в избранном
6	date_added	TEXT	NOT NULL	INDEX	Дата добавления в избранное

Связи между таблицами реализованы с помощью внешних ключей для обеспечения целостности данных.

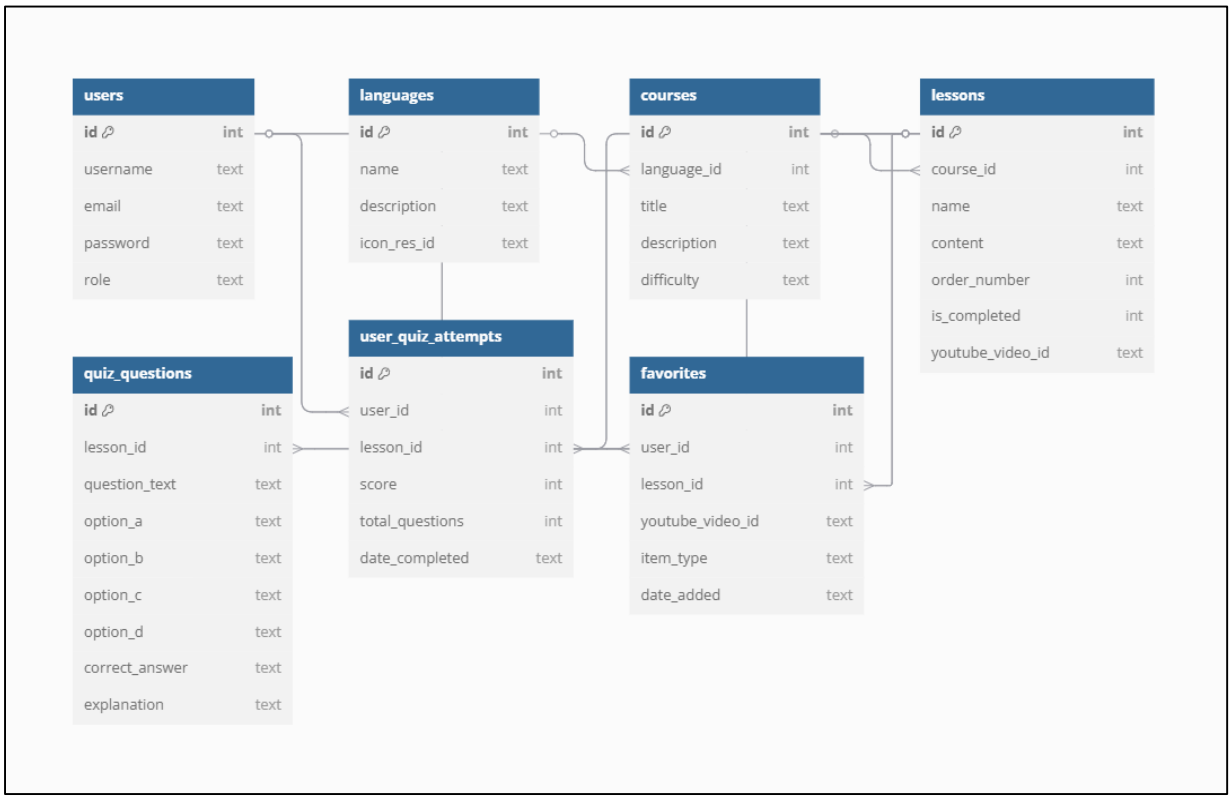


Рисунок12 – Физическая модель БД

1.4 Разработка приложения

1.4.1 Реализация БД

DatabaseHelper - центральный компонент доступа к данным.

DatabaseHelper наследуется от SQLiteOpenHelper и реализует паттерн DAO (Data Access Object) для каждой сущности:

```
class DatabaseHelper(context: Context) : SQLiteOpenHelper(context,
DATABASE_NAME, null, DATABASE_VERSION)
```

Далее мы объявляем статические константы, которые содержат: названия таблиц и названия полей в таблицах.

```
companion object {
    private const val DATABASE_VERSION = 5
    private const val DATABASE_NAME = "LearnProgramming.db"

    // Таблица Users
    private const val TABLE_USERS = "users"
    private const val KEY_USER_ID = "id"
```

```
    private const val KEY_USER_USERNAME = "username"
    private const val KEY_USER_EMAIL = "email"
    private const val KEY_USER_PASSWORD = "password"
    private const val KEY_USER_ROLE = "role"

    // Таблица ProgrammingLanguages
    private const val TABLE_LANGUAGES = "languages"
    private const val KEY_LANG_ID = "id"
    private const val KEY_LANG_NAME = "name"
    private const val KEY_LANG_DESCRIPTION = "description"
    private const val KEY_LANG_ICON_RES_ID = "icon_res_id"

    // Таблица Courses
    private const val TABLE_COURSES = "courses"
    private const val KEY_COURSE_ID = "id"
    private const val KEY_COURSE_LANG_ID = "language_id"
    private const val KEY_COURSE_TITLE = "title"
    private const val KEY_COURSE_DESCRIPTION = "description"
    private const val KEY_COURSE_DIFFICULTY = "difficulty"
```

Далее мы создаем таблицы баз данных и определяем внешние ключи.

```
override fun onCreate(db: SQLiteDatabase?) {
    val createUserTable = ("CREATE TABLE " + TABLE_USERS + "("
            + KEY_USER_ID + " INTEGER PRIMARY KEY AUTOINCREMENT,"
            + KEY_USER_USERNAME + " TEXT UNIQUE,"
            + KEY_USER_EMAIL + " TEXT UNIQUE,"
            + KEY_USER_PASSWORD + " TEXT,"
            + KEY_USER_ROLE + " TEXT DEFAULT 'user'" + ")")
    db?.execSQL(createUserTable)

    val createLanguageTable = ("CREATE TABLE " + TABLE_LANGUAGES + "("
            + KEY_LANG_ID + " INTEGER PRIMARY KEY AUTOINCREMENT,"
            + KEY_LANG_NAME + " TEXT UNIQUE,"
            + KEY_LANG_DESCRIPTION + " TEXT,"
            + KEY_LANG_ICON_RES_ID + " TEXT" + ")")
    db?.execSQL(createLanguageTable)

    val createCourseTable = ("CREATE TABLE " + TABLE_COURSES + "("
            + KEY_COURSE_ID + " INTEGER PRIMARY KEY AUTOINCREMENT,"
            + KEY_COURSE_LANG_ID + " INTEGER,"
            + KEY_COURSE_TITLE + " TEXT,"
            + KEY_COURSE_DESCRIPTION + " TEXT,"
            + KEY_COURSE_DIFFICULTY + " TEXT,"
            + "FOREIGN KEY(" + KEY_COURSE_LANG_ID + ") REFERENCES " +
TABLE_LANGUAGES + "(" + KEY_LANG_ID + ")" + ")")
    db?.execSQL(createCourseTable)
```

Добавление нового пользователя в базу данных.

```
fun addUser(user: User): Long {
    val db = this.writableDatabase
val values = ContentValues()
    values.put(KEY_USER_USERNAME, user.username)
```

```
    values.put(KEY_USER_EMAIL, user.email)
    values.put(KEY_USER_PASSWORD, user.password)
    values.put(KEY_USER_ROLE, user.role)

    val id = db.insert(TABLE_USERS, null, values)
    db.close()
    return id
}

fun getUserByUsername(username: String): User? {
val db = this.readableDatabase
val cursor = db.query(
        TABLE_USERS, arrayOf(KEY_USER_ID, KEY_USER_USERNAME,
KEY_USER_EMAIL, KEY_USER_PASSWORD, KEY_USER_ROLE),
        KEY_USER_USERNAME + "=?", arrayOf(username), null, null, null,
null
    )
    var user: User? = null
    if (cursor != null && cursor.moveToFirst()) {
        user = User(
            id =
cursor.getLong(cursor.getColumnIndexOrThrow(KEY_USER_ID)),
            username =
cursor.getString(cursor.getColumnIndexOrThrow(KEY_USER_USERNAME)),
            email =
cursor.getString(cursor.getColumnIndexOrThrow(KEY_USER_EMAIL)),
            password =
cursor.getString(cursor.getColumnIndexOrThrow(KEY_USER_PASSWORD)),
            role =
cursor.getString(cursor.getColumnIndexOrThrow(KEY_USER_ROLE))
        )
        cursor.close()
    }
    db.close()
    return user
}
```

Создаем функции для добавления данных в соответствующие таблицы.

```
private fun addLanguage(db: SQLiteDatabase?, language:
ProgrammingLanguage): Long {
    val values = ContentValues()
    values.put(KEY_LANG_NAME, language.name)
    values.put(KEY_LANG_DESCRIPTION, language.description)
    values.put(KEY_LANG_ICON_RES_ID, language.iconResId)
    return db?.insert(TABLE_LANGUAGES, null, values) ?: -1L
}

private fun addCourse(db: SQLiteDatabase?, course: Course): Long {
    val values = ContentValues()
    values.put(KEY_COURSE_LANG_ID, course.languageId)
    values.put(KEY_COURSE_TITLE, course.title)
    values.put(KEY_COURSE_DESCRIPTION, course.description)
    values.put(KEY_COURSE_DIFFICULTY, course.difficulty)
```

```
    return db?.insert(TABLE_COURSES, null, values) ?: -1L
}
```

Создаем функции для добавления, извлечения и обновления данных в соответствующих таблицах.

```
fun getQuizQuestionById(questionId: Int): QuizQuestion? {
val db = this.readableDatabase
var question: QuizQuestion? = null
    val cursor = db.query(
        TABLE_QUIZ_QUESTIONS, null, "$KEY_QUESTION_ID = ?",
arrayOf(questionId.toString()), null, null, null
    )
    if (cursor.moveToFirst()) {
        question = QuizQuestion(
            id =
cursor.getInt(cursor.getColumnIndexOrThrow(KEY_QUESTION_ID)),
            lessonId =
cursor.getInt(cursor.getColumnIndexOrThrow(KEY_QUESTION_LESSON_ID)),
            questionText =
cursor.getString(cursor.getColumnIndexOrThrow(KEY_QUESTION_TEXT)),
            optionA =
cursor.getString(cursor.getColumnIndexOrThrow(KEY_QUESTION_OPTION_A)),
            optionB =
cursor.getString(cursor.getColumnIndexOrThrow(KEY_QUESTION_OPTION_B)),
            optionC =
cursor.getString(cursor.getColumnIndexOrThrow(KEY_QUESTION_OPTION_C)),
            optionD =
cursor.getString(cursor.getColumnIndexOrThrow(KEY_QUESTION_OPTION_D)),
            correctAnswer =
cursor.getString(cursor.getColumnIndexOrThrow(KEY_QUESTION_CORRECT_ANS
WER)),
            explanation =
cursor.getString(cursor.getColumnIndexOrThrow(KEY_QUESTION_EXPLANATION
))
        )
    }
    cursor.close()
    db.close()
    return question
}

fun updateQuizQuestion(question: QuizQuestion): Int {
    val db = this.writableDatabase
val values = ContentValues().apply {
put(KEY_QUESTION_LESSON_ID, question.lessonId)
        put(KEY_QUESTION_TEXT, question.questionText)
        put(KEY_QUESTION_OPTION_A, question.optionA)
        put(KEY_QUESTION_OPTION_B, question.optionB)
        put(KEY_QUESTION_OPTION_C, question.optionC)
        put(KEY_QUESTION_OPTION_D, question.optionD)
        put(KEY_QUESTION_CORRECT_ANSWER, question.correctAnswer)
```

```
        put(KEY_QUESTION_EXPLANATION, question.explanation)
}
val result = db.update(TABLE_QUIZ_QUESTIONS, values, "$KEY_QUESTION_ID
= ?", arrayOf(question.id.toString()))
    db.close()
    return result
}
```

1.4.2 Описание программы

Создание всез Activityкомпонентов.

Стартовый экран. Файл SplashScreenActivity реализует инициализацию приложения, проверку аунтификации, анимированный переход.

```
Packagecom.learnprogramming.app

import android.annotation.SuppressLint
import android.content.Intent
import android.os.Bundle
import android.os.Handler
import android.os.Looper
import android.view.animation.AnimationUtils
import android.widget.ImageView
import android.widget.TextView
import androidx.appcompat.app.AppCompatActivity
import com.learnprogramming.app.auth.AuthManager
import com.learnprogramming.app.auth.LoginActivity

@SuppressLint("CustomSplashScreen") // Стандартный SplashScreen API
доступен с API 31, делаем кастомный для совместимости
class SplashScreenActivity : AppCompatActivity() {

    private val SPLASH_TIME_OUT: Long = 2500 // Увеличил время для
анимации

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_splash)

        val logoImageView: ImageView =
findViewById(R.id.imageViewLogo)
        val appNameTextView: TextView =
findViewById(R.id.textViewAppName)

        val fadeInAnimation = AnimationUtils.loadAnimation(this,
R.anim.fade_in)
        logoImageView.startAnimation(fadeInAnimation)
        appNameTextView.startAnimation(fadeInAnimation)

        Handler(Looper.getMainLooper()).postDelayed({
if (AuthManager.isLoggedIn(this)) {
```

```
                startActivity(Intent(this, MainActivity::class.java))
            } else {
                startActivity(Intent(this, LoginActivity::class.java))
            }
            finish() // Закрываем SplashScreenActivity,
чтобыпользовательнемогвернутьсянанееекнопкой "назад"
}, SPLASH_TIME_OUT)
    }
}
```

Основной экран. Файл MainActivity реализует Bottom Navigation, управление фрагментами, центральную навигацию.

```
package com.learnprogramming.app

import android.os.Bundle
import androidx.appcompat.app.AppCompatActivity
import androidx.fragment.app.Fragment
import com.learnprogramming.app.databinding.ActivityMainBinding
import com.learnprogramming.app.fragment.FavoritesFragment
import com.learnprogramming.app.ui.fragments.HomeFragment
import com.learnprogramming.app.ui.fragments.ProfileFragment
import com.learnprogramming.app.auth.AuthManager
import android.content.Intent
import com.learnprogramming.app.ui.AdminDashboardActivity

class MainActivity : AppCompatActivity() {

    private lateinit var binding: ActivityMainBinding

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)

        // Проверкаролипользователя
        val userRole = AuthManager.getLoggedInUserRole(this)
        if (userRole == "admin") {
            // Еслиадминистратор, перенаправляемв
AdminDashboardActivity
            startActivity(Intent(this,
AdminDashboardActivity::class.java))
            finish() // Завершаем MainActivity,
чтобыпользовательнемогвернутьсякнейкнопкой "Назад"
            return // Прерываемвыполнение onCreate для MainActivity
        }

        binding = ActivityMainBinding.inflate(layoutInflater)
        setContentView(binding.root)

        // Устанавливаем HomeFragment поумолчанию
        if (savedInstanceState == null) {
            replaceFragment(HomeFragment())
            binding.bottomNavigation.selectedItemId = R.id.nav_home
```

```
}

        binding.bottomNavigation.setOnItemSelectedListener { item ->
when (item.itemId) {
                R.id.nav_home -> replaceFragment(HomeFragment())
                R.id.nav_favorites ->
replaceFragment(FavoritesFragment())
                R.id.nav_profile -> replaceFragment(ProfileFragment())
                // Добавьтедругиепунктыменюздесь, еслионипоявятся
            }
            true
}
}

    private fun replaceFragment(fragment: Fragment) {
supportFragmentManager.beginTransaction()
            .replace(R.id.fragment_container, fragment)
            .commit()
    }
```

Аунтификация. Файл LoginActivityреализует вход пользователя, валидацию данных и управление сессией.

```
package com.learnprogramming.app.auth

import android.content.Intent
import android.os.Bundle
import android.widget.Toast
import androidx.appcompat.app.AppCompatActivity
import com.learnprogramming.app.MainActivity // Предполагается, что
MainActivity - этоглавныйэкранпослевхода
import com.learnprogramming.app.R
import com.learnprogramming.app.databinding.ActivityLoginBinding
import com.learnprogramming.app.db.DatabaseHelper

class LoginActivity : AppCompatActivity() {

    private lateinit var binding: ActivityLoginBinding
    private lateinit var dbHelper: DatabaseHelper

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        binding = ActivityLoginBinding.inflate(layoutInflater)
        setContentView(binding.root)

        dbHelper = DatabaseHelper(this)

        // Проверка, еслипользовательужевошел, перенаправляемна
MainActivity
        if (AuthManager.isLoggedIn(this)) {
            startActivity(Intent(this, MainActivity::class.java))
            finish()
```

```
            return
        }

        binding.buttonLogin.setOnClickListener {
val username = binding.editTextLoginUsername.text.toString().trim()
            val password =
binding.editTextLoginPassword.text.toString().trim()

            if (username.isEmpty() || password.isEmpty()) {
                Toast.makeText(this, "Пожалуйста, заполнитевсеполя",
Toast.LENGTH_SHORT).show()
                return@setOnClickListener
            }

            val user = dbHelper.getUserByUsername(username)

            if (user != null && user.password == password) { //
Вреальномприложениисравнениехэшейпаролей
                AuthManager.setLoggedIn(
                    this,
                    true,
                    user.id,
                    user.username,
                    user.role,
                    user.email
                )
                Toast.makeText(this, "Входвыполненуспешно!",
Toast.LENGTH_SHORT).show()
                val intent = Intent(this, MainActivity::class.java)
                startActivity(intent)
                finish()
            } else {
                Toast.makeText(this,
"Неверноеимяпользователяилипароль", Toast.LENGTH_SHORT).show()
            }
}

binding.textViewGoToRegister.setOnClickListener {
startActivity(Intent(this, RegisterActivity::class.java))
}
}
}
```

Каталог курсов. Файл CoursesActivityреализует отображение курсов по языку и фильтрацию по сложности.

```
class CoursesActivity : AppCompatActivity() {

    private lateinit var binding: ActivityCoursesBinding
    private lateinit var dbHelper: DatabaseHelper
    private lateinit var courseAdapter: CourseAdapter
    private var coursesList = mutableListOf<Course>()
    private var languageId: Long = -1L
```

```
    private var languageName: String? = null

    companion object {
        const val EXTRA_LANGUAGE_ID = "LANGUAGE_ID"
        const val EXTRA_LANGUAGE_NAME = "LANGUAGE_NAME"
    }

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        binding = ActivityCoursesBinding.inflate(layoutInflater)
        setContentView(binding.root)

        dbHelper = DatabaseHelper(this)

        languageId = intent.getLongExtra(EXTRA_LANGUAGE_ID, -1L)
        languageName = intent.getStringExtra(EXTRA_LANGUAGE_NAME)

        if (languageId == -1L) {
            Toast.makeText(this, "Ошибка:
неуказанязыкпрограммирования", Toast.LENGTH_LONG).show()
            finish()
            return
        }

supportActionBar?.title = "Курсы: ${languageName ?: ""}"
supportActionBar?.setDisplayHomeAsUpEnabled(true)
        binding.textViewCoursesTitle.text = "Курсыпо ${languageName ?:
"выбранномуязыку"}"

        setupRecyclerView()
        loadCourses()
    }

    private fun setupRecyclerView() {
        // courseAdapter = CourseAdapter(coursesList) { course ->
        //     // Этотобработчиктеперьненужен, переходв
LessonsActivity происходитвсамомадаптере
        //     Toast.makeText(this, "Выбранкурс: ${course.title}",
Toast.LENGTH_SHORT).show()
        // }
        courseAdapter = CourseAdapter(this, coursesList) //
Обновленныйконструктор
        binding.recyclerViewCourses.apply {
layoutManager = LinearLayoutManager(this@CoursesActivity)
adapter = courseAdapter
}
}

    private fun loadCourses() {
        val courses = dbHelper.getCoursesForLanguage(languageId)
        coursesList.clear()
        coursesList.addAll(courses)
        if (coursesList.isEmpty()){
```

```
            Toast.makeText(this, "Дляэтогоязыкапоканеткурсов",
Toast.LENGTH_SHORT).show()
        }
        courseAdapter.notifyDataSetChanged()
    }

    override fun onOptionsItemSelected(item: android.view.MenuItem):
Boolean {
        return when (item.itemId) {
            android.R.id.home -> {
                finish()
                true
            }
            else -> super.onOptionsItemSelected(item)
        }
    }
}
```

Список уроков. Файл LessonsActivityреализует отображение уроков курса и отслеживание прогресса.

```
class LessonsActivity : AppCompatActivity() {

    private lateinit var binding: ActivityLessonsBinding
    private lateinit var dbHelper: DatabaseHelper
    private lateinit var lessonAdapter: LessonAdapter
    private var lessonsList = mutableListOf<Lesson>()
    private var courseId: Long = -1L
    private var courseName: String? = null

    companion object {
        const val EXTRA_COURSE_ID = "COURSE_ID"
        const val EXTRA_COURSE_NAME = "COURSE_NAME"
    }

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        binding = ActivityLessonsBinding.inflate(layoutInflater)
        setContentView(binding.root)

        dbHelper = DatabaseHelper(this)

        courseId = intent.getLongExtra(EXTRA_COURSE_ID, -1L)
        courseName = intent.getStringExtra(EXTRA_COURSE_NAME)

        if (courseId == -1L) {
            Toast.makeText(this, "Ошибка: неуказанкурс",
Toast.LENGTH_LONG).show()
            finish()
            return
        }

supportActionBar?.title = "Уроки: ${courseName ?: ""}"
```

```
supportActionBar?.setDisplayHomeAsUpEnabled(true)
        binding.textViewLessonsTitle.text = "Урокикурса \"${courseName
?: "Выбранныйкурс"}\""

        setupRecyclerView()
        loadLessons()
    }

    private fun setupRecyclerView() {
        lessonAdapter = LessonAdapter(lessonsList) { lesson ->
val intent = LessonDetailActivity.newIntent(this, lesson.id,
lesson.name)
            startActivity(intent)
}
binding.recyclerViewLessons.apply {
layoutManager = LinearLayoutManager(this@LessonsActivity)
adapter = lessonAdapter
}
}

    private fun loadLessons() {
        val lessons = dbHelper.getLessonsForCourse(courseId)
        lessonsList.clear()
        lessonsList.addAll(lessons)
        if (lessonsList.isEmpty()){
            Toast.makeText(this, "Дляэтогокурсапоканетуроков",
Toast.LENGTH_SHORT).show()
        }
        lessonAdapter.notifyDataSetChanged()
    }

    override fun onOptionsItemSelected(item: android.view.MenuItem):
Boolean {
        return when (item.itemId) {
            android.R.id.home -> {
                finish()
                true
            }
            else -> super.onOptionsItemSelected(item)
        }
    }
}
```

Детали урока. Файл LessonDetailActivityреализует отображение контента, интеграцию видео и управление избранным.

```
class LessonDetailActivity : AppCompatActivity() {

    private lateinit var binding: ActivityLessonDetailBinding
    private lateinit var dbHelper: DatabaseHelper
    private var lessonId: Int = -1
    private var userId: Long = -1
    private var currentLesson: Lesson? = null
```

```
    private var isFavorite: Boolean = false

    companion object {
        const val EXTRA_LESSON_ID = "extra_lesson_id"
        const val EXTRA_LESSON_NAME = "extra_lesson_name"

        fun newIntent(context: Context, lessonId: Int, lessonName:
String): Intent {
            val intent = Intent(context,
LessonDetailActivity::class.java)
            intent.putExtra(EXTRA_LESSON_ID, lessonId)
            intent.putExtra(EXTRA_LESSON_NAME, lessonName)
            return intent
        }
    }

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        binding = ActivityLessonDetailBinding.inflate(layoutInflater)
        setContentView(binding.root)

        dbHelper = DatabaseHelper(this)
        lessonId = intent.getIntExtra(EXTRA_LESSON_ID, -1)
        val lessonName = intent.getStringExtra(EXTRA_LESSON_NAME)

supportActionBar?.title = lessonName ?: "Урок"
supportActionBar?.setDisplayHomeAsUpEnabled(true)

        if (lessonId == -1) {
            Toast.makeText(this, "Ошибка: Урокненайден",
Toast.LENGTH_LONG).show()
            finish()
            return
        }

        userId = AuthManager.getLoggedInUserId(this)

        loadLessonDetails()
        setupListeners()
        updateFavoriteButtonState()
    }

    private fun loadLessonDetails() {
        currentLesson = dbHelper.getLessonById(lessonId)
        currentLesson?.let {
binding.textViewLessonTitle.text = it.name
            binding.textViewLessonContent.text = it.content // Make
sure content is plain text or use WebView for HTML
            updateMarkAsCompleteButton(it.isCompleted)

            if (userId != -1L) {
                isFavorite = dbHelper.isLessonInFavorites(userId,
it.id.toLong())
```

```
            }
            updateFavoriteButtonState()

            // Показываемилискрываемкнопку YouTube
            if (!it.youtubeVideoId.isNullOrEmpty()) {
                binding.buttonWatchOnYouTube.visibility = View.VISIBLE
} else {
                binding.buttonWatchOnYouTube.visibility = View.GONE
}

            // Check for quiz questions and update button visibility
            val quizQuestions =
dbHelper.getQuizQuestionsForLesson(it.id)
            if (quizQuestions.isNotEmpty()) {
                binding.buttonGoToQuiz.visibility = View.VISIBLE
} else {
                binding.buttonGoToQuiz.visibility = View.GONE
}
} ?: run {
Toast.makeText(this, "Неудалосьзагрузитьдеталиурока",
Toast.LENGTH_LONG).show()
            finish()
}
}

    private fun setupListeners() {
        binding.buttonMarkAsComplete.setOnClickListener {
currentLesson?.let {
val newCompletionStatus = !it.isCompleted
                val rowsAffected =
dbHelper.updateLessonCompletionStatus(it.id, newCompletionStatus)
                if (rowsAffected > 0) {
it.isCompleted = newCompletionStatus
                    updateMarkAsCompleteButton(newCompletionStatus)
                    val message = if (newCompletionStatus)
"Урокотмеченкакпройденный" else "Отметкаопрохожденииурокаснята"
                    Toast.makeText(this, message,
Toast.LENGTH_SHORT).show()
                } else {
                    Toast.makeText(this,
"Неудалосьобновитьстатусурока", Toast.LENGTH_SHORT).show()
```

Тестирование. ФайлQuizActivityреализуетпроведениетестов, подсчетрезультатаисохранениепопыток.

```
class QuizActivity : AppCompatActivity() {

    private lateinit var binding: ActivityQuizBinding
    private lateinit var dbHelper: DatabaseHelper
    private var lessonId: Int = -1
    private lateinit var questionsList: List<QuizQuestion>
    private var currentQuestionIndex: Int = 0
```

```
    private var score: Int = 0

    companion object {
        const val EXTRA_LESSON_ID = "extra_lesson_id"
        const val EXTRA_LESSON_NAME = "extra_lesson_name"

        fun newIntent(context: Context, lessonId: Int, lessonName:
String): Intent {
            val intent = Intent(context, QuizActivity::class.java)
            intent.putExtra(EXTRA_LESSON_ID, lessonId)
            intent.putExtra(EXTRA_LESSON_NAME, lessonName)
            return intent
        }
    }

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        binding = ActivityQuizBinding.inflate(layoutInflater)
        setContentView(binding.root)

        dbHelper = DatabaseHelper(this)
        lessonId = intent.getIntExtra(EXTRA_LESSON_ID, -1)
        val lessonName = intent.getStringExtra(EXTRA_LESSON_NAME)

supportActionBar?.title = "Тест: ${lessonName ?: "Урок"}"
supportActionBar?.setDisplayHomeAsUpEnabled(true)

        if (lessonId == -1) {
            Toast.makeText(this, "Ошибка: Тестненайден",
Toast.LENGTH_LONG).show()
            finish()
            return
        }
```

Админ-панель. Файл AdminDashboard Activity реализует управление контентом, статистикой и системными функциями.

```
class AdminDashboardActivity : AppCompatActivity() {

    private lateinit var binding: ActivityAdminDashboardBinding

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        binding =
ActivityAdminDashboardBinding.inflate(layoutInflater)
        setContentView(binding.root)

supportActionBar?.title = "ПанельАдминистратора"

        binding.buttonManageLessons.setOnClickListener {
startActivity(Intent(this, ManageLessonsActivity::class.java))
}
```

```
binding.buttonManageQuizzes.setOnClickListener {
startActivity(Intent(this, ManageQuizzesActivity::class.java))
}

binding.buttonAdminLogout.setOnClickListener {
AuthManager.setLoggedIn(this, false)
            val intent = Intent(this, LoginActivity::class.java)
            intent.flags = Intent.FLAG_ACTIVITY_NEW_TASK or
Intent.FLAG_ACTIVITY_CLEAR_TASK
startActivity(intent)
            finish()
}
}
}
```

Реализация вкладки “Избранное”.

Файл FavoritesFragmentреализует группировку по типам контента с возможностью добавления и удаления.

```
class FavoritesFragment : Fragment() {

    private lateinit var recyclerViewFavorites: RecyclerView
    private lateinit var textViewNoFavorites: TextView
    private lateinit var favoriteAdapter: FavoriteAdapter
    private lateinit var dbHelper: DatabaseHelper
    private var userId: Long = -1
    private var favoriteItems: List<FavoriteItem> = emptyList()

    override fun onCreateView(
        inflater: LayoutInflater, container: ViewGroup?,
        savedInstanceState: Bundle?
    ): View? {
        val view = inflater.inflate(R.layout.fragment_favorites,
container, false)
        dbHelper = DatabaseHelper(requireContext())
        userId = AuthManager.getLoggedInUserId(requireContext())

        recyclerViewFavorites =
view.findViewById(R.id.recyclerViewFavorites)
        textViewNoFavorites =
view.findViewById(R.id.textViewNoFavorites)

        recyclerViewFavorites.layoutManager =
LinearLayoutManager(context)
        return view
    }

    override fun onResume() {
        super.onResume()
        loadFavorites()
    }
```

```
    private fun loadFavorites() {
        if (userId != -1L) {
            favoriteItems = dbHelper.getUserFavorites(userId) //
favoriteItems теперь List
            if (favoriteItems.isEmpty()) {
                recyclerViewFavorites.visibility = View.GONE
textViewNoFavorites.visibility = View.VISIBLE
textViewNoFavorites.text = "Визбранномпоканичегонет."
```

Реализация пользовательского интерфейса.

Файл LanguageAdapterреализует отображение языков с иконками и обработку кликов.

```
class LanguageAdapter(
    private val context: Context,
    private val languages: List<ProgrammingLanguage>
) : RecyclerView.Adapter<LanguageAdapter.LanguageViewHolder>() {

    override fun onCreateViewHolder(parent: ViewGroup, viewType: Int):
LanguageViewHolder {
        val binding =
ItemLanguageBinding.inflate(LayoutInflater.from(parent.context),
parent, false)
        return LanguageViewHolder(binding)
    }

    override fun onBindViewHolder(holder: LanguageViewHolder,
position: Int) {
        val language = languages[position]
        holder.bind(language)
    }

    override fun getItemCount(): Int = languages.size

    inner class LanguageViewHolder(private val binding:
ItemLanguageBinding) : RecyclerView.ViewHolder(binding.root) {
        fun bind(language: ProgrammingLanguage) {
            binding.textViewLanguageName.text = language.name
            binding.textViewLanguageDescription.text =
language.description

            language.iconResId?.let {
val resourceId = context.resources.getIdentifier(it, "drawable",
context.packageName)
                if (resourceId != 0) {

binding.imageViewLanguageIcon.setImageResource(resourceId)
                } else {

binding.imageViewLanguageIcon.setImageResource(R.mipmap.ic_launcher)
```

```
                }
} ?:
binding.imageViewLanguageIcon.setImageResource(R.mipmap.ic_launcher)

            binding.root.setOnClickListener {
val intent = Intent(context, CoursesActivity::class.java).apply {
putExtra(CoursesActivity.EXTRA_LANGUAGE_ID, language.id)
                    putExtra(CoursesActivity.EXTRA_LANGUAGE_NAME,
language.name)
}
context.startActivity(intent)
}
}
    }
}
```

Файл CourseAdapterреализует курсы с индикатором сложности и прогресс-барами.

```
class CourseAdapter(
    private val context: Context,
    private val courses: List<Course>
) : RecyclerView.Adapter<CourseAdapter.CourseViewHolder>() {

    override fun onCreateViewHolder(parent: ViewGroup, viewType: Int):
CourseViewHolder {
        val binding =
ItemCourseBinding.inflate(LayoutInflater.from(parent.context), parent,
false)
        return CourseViewHolder(binding)
    }

    override fun onBindViewHolder(holder: CourseViewHolder, position:
Int) {
        val course = courses[position]
        holder.bind(course)
    }

    override fun getItemCount(): Int = courses.size

    inner class CourseViewHolder(private val binding:
ItemCourseBinding) : RecyclerView.ViewHolder(binding.root) {
        fun bind(course: Course) {
            binding.textViewCourseTitle.text = course.title
            binding.textViewCourseDescription.text =
course.description
            binding.textViewCourseDifficulty.text = "Сложность:
${course.difficulty}"

            binding.root.setOnClickListener {
val intent = Intent(context, LessonsActivity::class.java).apply {
putExtra(LessonsActivity.EXTRA_COURSE_ID, course.id)
                    putExtra(LessonsActivity.EXTRA_COURSE_NAME,
```

```
course.title)
}
context.startActivity(intent)
}
}
    }
}
```

Файл LessonAdapterреализует уроки с статусом прохождения и временными метками.

```
class LessonAdapter(
    private val lessons: List<Lesson>,
    private val onItemClick: (Lesson) -> Unit
) : RecyclerView.Adapter<LessonAdapter.LessonViewHolder>() {

    override fun onCreateViewHolder(parent: ViewGroup, viewType: Int):
LessonViewHolder {
        val binding =
ItemLessonBinding.inflate(LayoutInflater.from(parent.context), parent,
false)
        return LessonViewHolder(binding)
    }

    override fun onBindViewHolder(holder: LessonViewHolder, position:
Int) {
        val lesson = lessons[position]
        holder.bind(lesson)
    }

    override fun getItemCount(): Int = lessons.size

    inner class LessonViewHolder(private val binding:
ItemLessonBinding) : RecyclerView.ViewHolder(binding.root) {
        fun bind(lesson: Lesson) {
            binding.textViewLessonOrder.text = "${lesson.order}."
            binding.textViewLessonTitle.text = lesson.name
            // Тутможнодобавитьлогикудляотображенияиконки (например,
еслиурокпройден)
            //
binding.imageViewLessonAction.setImageResource(R.drawable.ic_check_cir
cle)

            binding.root.setOnClickListener {
onItemClick(lesson)
}
}
    }
}
```

Файл AdminLessonAdapterреализует управление элементами, редактирование, удаление.

```
class AdminLessonAdapter(
    private val lessons: MutableList<Lesson>,
    private val onEditClick: (Lesson) -> Unit,
    private val onDeleteClick: (Lesson) -> Unit
) : RecyclerView.Adapter<AdminLessonAdapter.LessonViewHolder>() {

    override fun onCreateViewHolder(parent: ViewGroup, viewType: Int):
LessonViewHolder {
        val view =
LayoutInflater.from(parent.context).inflate(R.layout.item_admin_lesson
, parent, false)
        return LessonViewHolder(view)
    }

    override fun onBindViewHolder(holder: LessonViewHolder, position:
Int) {
        val lesson = lessons[position]
        holder.bind(lesson)
    }

    override fun getItemCount(): Int = lessons.size

    fun updateLessons(newLessons: List<Lesson>) {
        lessons.clear()
        lessons.addAll(newLessons)
        notifyDataSetChanged()
    }

    inner class LessonViewHolder(itemView: View) :
RecyclerView.ViewHolder(itemView) {
        private val lessonNameTextView: TextView =
itemView.findViewById(R.id.textViewAdminLessonName)
        private val editButton: ImageButton =
itemView.findViewById(R.id.buttonEditLesson)
        private val deleteButton: ImageButton =
itemView.findViewById(R.id.buttonDeleteLesson)

        fun bind(lesson: Lesson) {
            lessonNameTextView.text = lesson.name
            // Можнодобавитьотображение ID курсаилиязыка,
еслиэтонеобходимо

            editButton.setOnClickListener { onEditClick(lesson) }
deleteButton.setOnClickListener { onDeleteClick(lesson) }
}
    }
}
```

ФайлFavoriteAdapterреализуетгруппировкупотипам, swipe-to-delete.

```
class FavoriteAdapter(
    private val favoriteItems: MutableList<FavoriteItem>,
    private val dbHelper: DatabaseHelper,
```

```
    private val userId: Long,
    private val onItemAction: (FavoriteItem, String) -> Unit
) : RecyclerView.Adapter<FavoriteAdapter.FavoriteViewHolder>() {

    override fun onCreateViewHolder(parent: ViewGroup, viewType: Int):
FavoriteViewHolder {
        val view =
LayoutInflater.from(parent.context).inflate(R.layout.item_favorite,
parent, false)
        return FavoriteViewHolder(view)
    }

    override fun onBindViewHolder(holder: FavoriteViewHolder,
position: Int) {
        val item = favoriteItems[position]
        holder.bind(item)
    }

    override fun getItemCount(): Int = favoriteItems.size

    inner class FavoriteViewHolder(itemView: View) :
RecyclerView.ViewHolder(itemView) {
        private val imageViewIcon: ImageView =
itemView.findViewById(R.id.imageViewFavoriteIcon)
        private val textViewTitle: TextView =
itemView.findViewById(R.id.textViewFavoriteTitle)
        private val textViewType: TextView =
itemView.findViewById(R.id.textViewFavoriteType)
        private val textViewDateAdded: TextView =
itemView.findViewById(R.id.textViewFavoriteDateAdded)
        private val buttonRemove: ImageButton =
itemView.findViewById(R.id.buttonRemoveFromFavorites)

        fun bind(favoriteItem: FavoriteItem) {
            textViewTitle.text = favoriteItem.fullTitle
            textViewDateAdded.text = "Добавлено:
${formatDate(favoriteItem.dateAdded)}"

            when (favoriteItem.itemType) {
                "lesson" -> {
                    textViewType.text = "Тип: Урок"

imageViewIcon.setImageResource(R.drawable.ic_lesson)
                }
                "youtube_video" -> {
                    textViewType.text = "Тип: YouTube Видео"

imageViewIcon.setImageResource(R.drawable.ic_youtube)
                }
                else -> {
                    textViewType.text = "Тип: Неизвестно"

imageViewIcon.setImageResource(R.drawable.ic_help_outline)
```

```
                }
            }

            itemView.setOnClickListener {
onItemAction(favoriteItem, "open")
}

buttonRemove.setOnClickListener {
val position = adapterPosition
if (position != RecyclerView.NO_POSITION) {
                    val itemToRemove = favoriteItems[position]
                    dbHelper.removeFromFavorites(itemToRemove.id)
                    favoriteItems.removeAt(position)
                    notifyItemRemoved(position)
                    notifyItemRangeChanged(position,
favoriteItems.size)
                    onItemAction(itemToRemove, "delete")
                }
}
}

        private fun formatDate(dateString: String): String {
            return try {
                val inputFormat = SimpleDateFormat("yyyy-MM-
dd'T'HH:mm:ss'Z'", Locale.getDefault())
                inputFormat.timeZone = TimeZone.getTimeZone("UTC")
                val date = inputFormat.parse(dateString)
                val outputFormat = SimpleDateFormat("dd.MM.yyyy
HH:mm", Locale.getDefault())
                outputFormat.timeZone = TimeZone.getDefault()
                date?.let { outputFormat.format(it) } ?: dateString
            } catch (e: ParseException) {
                dateString
```

1.4.3 Руководствопользователя

Чтобы начать работу в приложении, необходимо зарегистрироваться или авторизоваться в системе (рисунок 13). Для этого в поля «Имя пользователя» и «Пароль» введем учетные данные пользователя, после чего нажмем кнопку «Войти». Для регистрации нажмем кнопку «Зарегистрироваться».

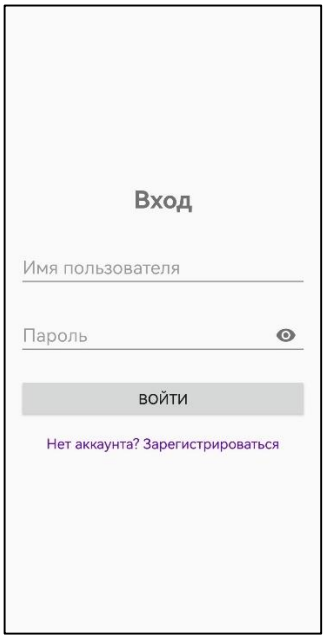


Рисунок 13 – Окно авторизации в системе

При попытке входа с хотя бы одним незаполненным полем на экране появится предупреждающее уведомление (рисунок 14).

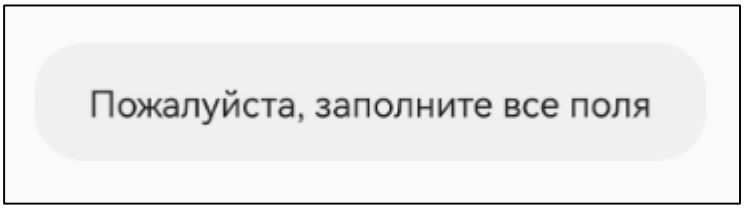


Рисунок 14 – Попытка входа с незаполненными полями

Если попытаться войти в профиль пользователя с несуществующим логином, появится соответствующее окно (рисунок 15).

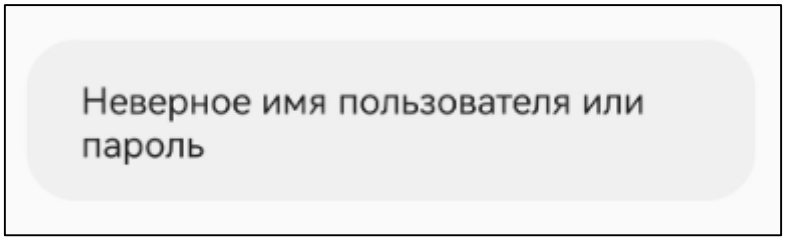


Рисунок 15 – Ошибкавхода

При попытке регистрации нового аккаунта с хотя бы одним незаполненным полем на экране появится предупреждение (рисунок 16).

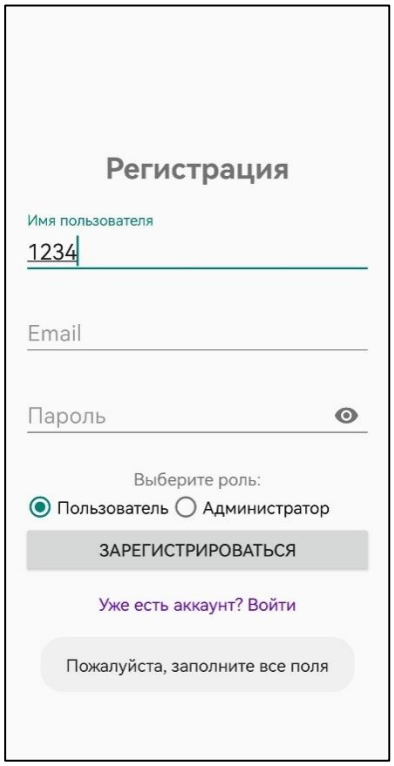


Рисунок 16 – Попытка регистрации с незаполненными полями

Если регистрация успешно пройдена, на экране появится соответствующее уведомление (рисунок 17).

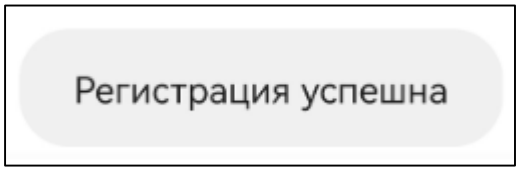


Рисунок 17 – Успешная регистрация

Если авторицация успешно пройдена, на экране появится соответствующее уведомление (рисунок 18).

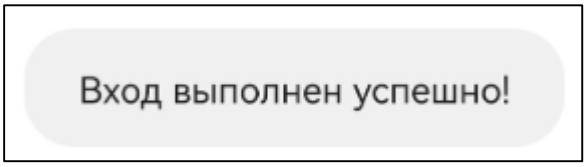


Рисунок 18 – Успешная авторизация

После успешной авторизации откроется главное меню программы (рисунок 19). Что бы выполнить нужное действие, необходимо щелкнуть по той или иной кнопке действия.

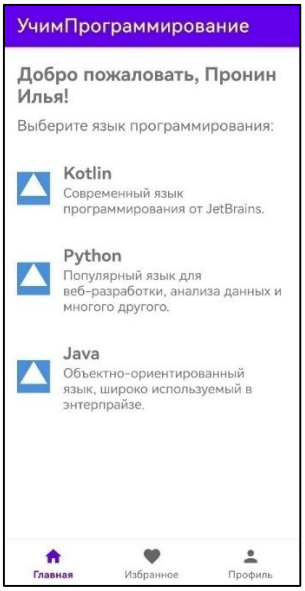


Рисунок 19 – Главное меню программы

Во вкладке “Главная” пользователю доступен выбор между доступными языками программирования. Для выбора необходимо нажать на желаемый язык программирования (рисунок 20).

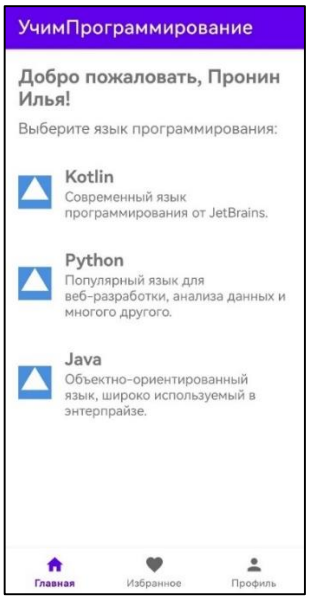


Рисунок 20 – Доступные языки программирования

После выбора языка программирования пользователю доступен выбор курса. Для выбора необходимо нажать на желаемый курс (рисунок 21).

Рисунок 21 – Доступные курсы

После выбора курса пользователю доступен выбор урока. Для выбора необходимо нажать на желаемый урок (рисунок 22).

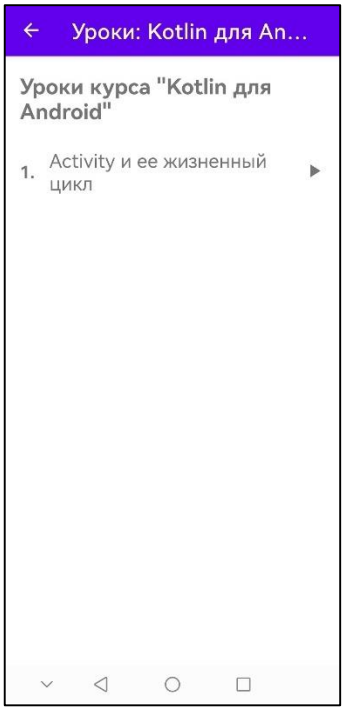


Рисунок 22 – Доступные уроки

После выбора урока пользователю доступен урок и ряд других функций (рисунок 23):

– избранное;
– метка о прохождении;
– перейти к тесту;
– вернуться к другим урокам.

Рисунок 23 – Функции урока

Для добавления урока в избранное нужно нажать на “Сердечко” в правом верхнем углу экрана. После добавления в избранное, на экране появится соответствующее уведомление (рисунок 24).

Рисунок 24 – Добавление в избранное

Для удаления урока из избранного нужно повторно нажать на “Сердечко” в правом верхнем углу экрана. После удаления из избранного, на экране появится соответствующее уведомление (рисунок 25).

Рисунок 25 – Удаление из избранного

Чтобы установить метку о прохождении, нужно нажать на синий прямоугольник по центру. После установления метки о прохождении, на экране появится соответствующее уведомление (рисунок 26).

Рисунок 26 – Установление метки о прохождении

Чтобы убрать метку о прохождении, нужно повторно нажать на синий прямоугольник по центру. После этого, на экране появится соответствующее уведомление (рисунок 27).

Рисунок 27 – Удаление метки о прохождении

Чтобы перейти к тесту, надо нажать на кнопку “Перейти к тесту” (рисунок 28).

Рисунок 28 – Перейти к тесту

После того как пользователь перешел к тесту, ему доступно задание с выбором ответа (рисунок 29).

Рисунок 29 – Задание с выбором ответа

В случае правильного ответа, на экране появляется соответствующее уведомление (рисунок 30).

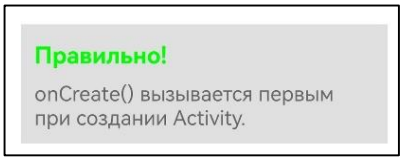


Рисунок 30 – Уведомление о правильном ответе

В случае неправильного ответа, на экране появляется соответствующее уведомление (рисунок 31).

Рисунок 31 – Уведомление о неправильном ответе

После ответа, пользователь может нажать на “Завешить тест” (рисунок 32).

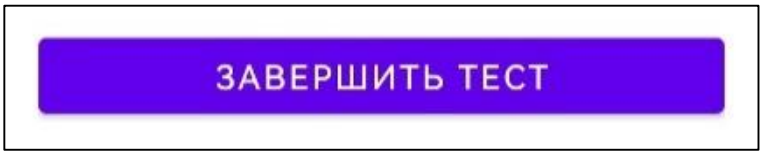


Рисунок 32 – Завершить тест

После завершения теста, пользователь получает специальное уведомление, в котором написан результат. Так же пользователь может перейти к урокам или пройти тест заново (рисунок 33).

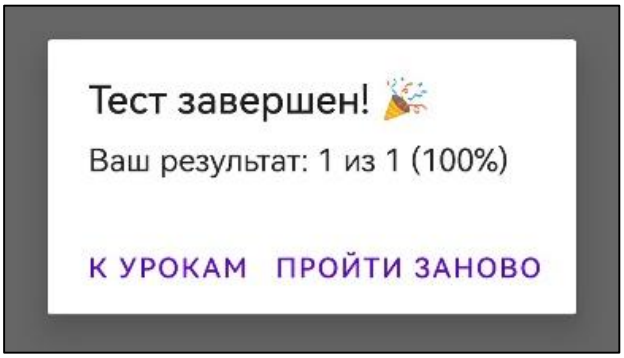


Рисунок 33 – Окно с результатом

Нажав на раздел “Избранное” в главном меню, мы видим все ранее добавленные в “избранное ” уроки (рисунок 34).

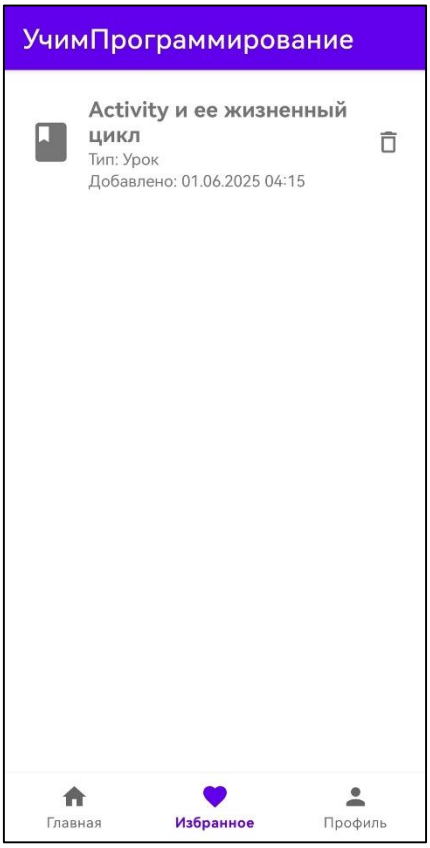


Рисунок 34 – Раздел “Избранное”

Нажав на раздел “Профиль” в главном меню, мы поподаем в свой профиль (рисунок 35).

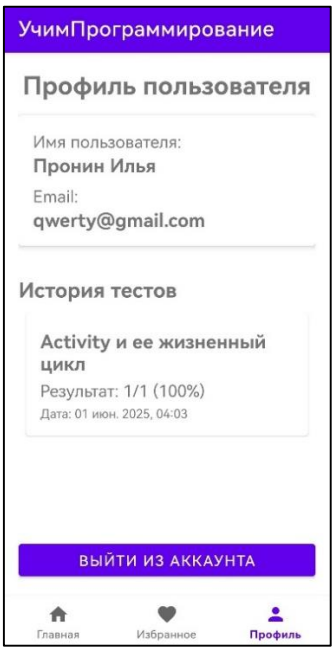


Рисунок 35 – Раздел “Профиль”

В разделе “Профиль” мы можем видеть свой данные от аккаунта (рисунок 36).

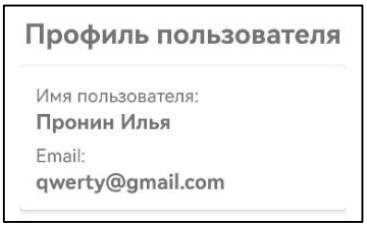


Рисунок 36 – Профиль пользователя

В разделе “Профиль” мы можем видеть свою историю тестов, где отображаются ранее пройденные тесты, их результаты и дата прохождения (рисунок 37).

Рисунок 37 – История тестов

Для выхода из аккаунта пользователь должен нажать кнопку “Выйти из аккаунта” в разделе “Профиль” (рисунок 38).

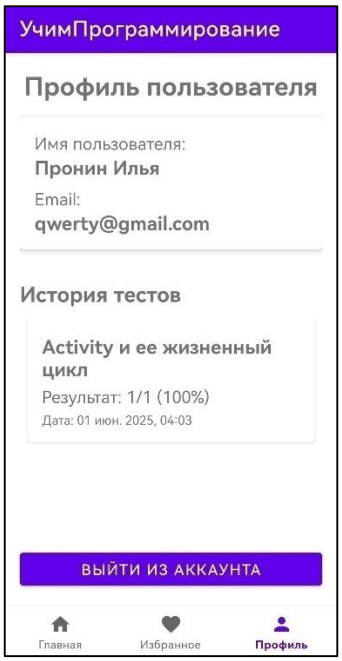


Рисунок 38 – Выход из аккаунта

1.4.4 Тестирование и отладка

В процессе работы с программой пользователь может допустить некоторые ошибки, которые могут привести к её некорректной работе. Например, пользователь может случайно забыть заполнить некоторые обязательные поля в момент регистрации аккаунта, либо неправильно ввести его электронный адрес, превысив его общую длину или нарушив формат ввода. Для таких случаев в программе предусмотрены обработчики исключительных ситуаций, работу которых требуется проверить путем составления тест-кейсов (таблица 9).

Таблица 9 – Тестирование и отладка

Наименование	Описание	Шаги для воспроизведения	Ожидаемый результат
1. Проверка наличия пустых полей во время авторизации в системе	Во время входа в систему пользователю необходимо заполнить поля «Логин» и «Пароль»	1. Запустить программу; 2. Не заполнив хотя бы одно из полей («Имя пользователя» или «Пароль»), нажать на кнопку «Войти»	На экране отобразилось уведомление: «Пожалуйста, заполните все поля»
2. Ввод несуществующего имени пользователя	Во время авторизации пользователь должен указать корректный логин и пароль для входа в систему	1. Запустить программу; 2. В поле «Имя пользователя» ввести несуществующее имя пользователя; 3. Нажать кнопку «Войти».	На экране появилось уведомление: «Неверное имя пользователя или пароль»
3. Ввод неправильного пароля для входа в систему	Необходимо ввести корректный логин и пароль для успешной авторизации в системе.	1. Запустить программу; 2. В поле «Имя пользователя» или «Пароль» ввести некорректные данные; 3. Нажать кнопку «Войти».	На экране появилось уведомление: «Неверное имя пользователя или пароль»

Наименование	Описание	Шаги для воспроизведения	Ожидаемый результат
4. Попытка зарегистрировать новый аккаунт с уже использующимся именем пользователя	Для регистрации нового аккаунта требуется указать имя пользователя, которое не используется другим пользователем	1. Запустить программу; 2. нажать кнопку «Регистрация клиента»; 3. В поле «имя пользователя» ввести уже занятое имя пользователя.	На экране появилось уведомление: «Пользователь с таким именем уже существует»
5. Регистрация нового аккаунта при наличии хотя бы одного незаполненного поля	Для регистрация нового аккаунта с указанием имени пользователя требуется заполнить все поля, после чего нажать кнопку «Зарегистрировать»	1. Запустить программу; 2. нажать кнопку «Регистрация клиента»; 3. Оставить хотя бы одно поле пустым.	На экране появилось уведомление: «Пожалуйста щаполните все поля»
6. Попытка добавления уже существующего урока	Для добавления урока требуется зайти в аккаунт администратора и заполнить все поля в меню добавления урока	1. Запустить программу; 2. Осуществить вход в систему с аккаунта администратора; 3. В главном меню нажать кнопку «Добавить урок»; 4. В поле ввести название уже существующего урока.	На экране появилось уведомление: «Урок с таким названием уже существует»
7. Попытка добавления уже существующего курса	Для добавления курса требуется зайти в аккаунт администратора и заполнить все поля в меню добавления курса»	1. Запустить программу; 2. Осуществить вход в систему с аккаунта администратора; 3. В главном меню нажать кнопку «Добавить курс»; 4. В поле ввести название уже существующего курса.	На экране появилось уведомление: «Курс с таким названием уже существует»

Наименование	Описание	Шаги для воспроизведения	Ожидаемый результат
8. Попытка добавления уже существующего теста	Для добавления теста требуется зайти в аккаунт администратора и заполнить все поля в меню добавления теста»	1. Запустить программу; 2. Осуществить вход в систему с аккаунта администратора; 3. В главном меню нажать кнопку «Добавить тест»; 4. В поле ввести название уже существующего теста.	На экране появилось уведомление: «Тест с таким названием уже существует»
9. Попытка добавления уже существующего языка программирования	Для добавления языка программирования требуется зайти в аккаунт администратора и заполнить все поля в меню добавления языка программирования»	1. Запустить программу; 2. Осуществить вход в систему с аккаунта администратора; 3. В главном меню нажать кнопку «Добавить язык программирования»; 4. В поле ввести название уже существующего языка программирования.	На экране появилось уведомление: «Языке программирования с таким названием уже существует»
10. Попытка ответа на вопрос без выбранного варианта ответа	Для попытки ответа на вопрос требуется начать тест по одному из выбранных уроков и нажать кнопку “ответить” не выбрав вариант ответа	1. Запустить программу; 2. Выполнить вход в систему; 3. В главном меню выбрать язык программирования; 4. Выбрать курс по выбранному языку программирования; 5. Выбрать урок; 6. Нажать на кнопку «Перейти к тесту»; 7. Нажать на кнопку «Ответить» не выбрав вариант ответа.	На экране появилось уведомление: «Пожалуйста выберите вариант ответа»

Наименование	Описание	Шаги для воспроизведения	Ожидаемый результат
11. Попытка выбора нескольких ответив в тесте с одним вариантом ответа	Для попытки выбора нескольких вариантов ответа в вопросе с одним вариантом ответа требуется начать тест по одному из выбранных уроков и выбрать несколько вариантов ответа	1. Запустить программу; 2. Выполнить вход в систему; 3. В главном меню выбрать язык программирования; 4. Выбрать курс по выбранному языку программирования; 5. Выбрать урок; 6. Нажать на кнопку «Перейти к тесту»; 7. Нажать на несколько вариантов ответа.	Предыдущий выбранный вариант ответа перестанет быть активным.

2 Охрана труда

Работа за компьютером требует соблюдения мер безопасности для сохранения здоровья, особенно для разработчиков и других специалистов, проводящих за экраном большую часть дня. Игнорирование этих правил может привести к проблемам со зрением, нарушениям осанки, развитию туннельного синдрома и другим заболеваниям. Рассмотрим ключевые аспекты безопасной организации рабочего процесса.

2.1 Организация рабочего пространства

Правильно оборудованное рабочее место — основа профилактики профессиональных заболеваний.

Требования к помещению:

– расположение. Предпочтительны помещения с окнами на север или северо-восток;

– площадь. Не менее 4,5 м² на одно рабочее место для обеспечения свободного движения и комфорта;

– освещение. Обязательно сочетание естественного и искусственного света. Для защиты от бликов используйте жалюзи или плотные шторы;

– вентиляция. Регулярное проветривание для снижения концентрации вредных веществ (например, формальдегида) и поддержания уровня кислорода;

– шум. Уровень не должен превышать 50 дБ — это минимизирует стресс и повышает концентрацию;

– микроклимат;

1) температура: 20–24°C (оптимально — 22°C);
2) влажность: 40–60% (максимум — 75%);
3) скорость движения воздуха: до 0,1 м/с.

Электробезопасность:

– заземление рабочего места для нейтрализации электромагнитного излучения от монитора и периферии;

– использование современных устройств с низким уровнем излучения.

2.2 Освещение: баланс между комфортом и продуктивностью

Неправильное освещение вызывает усталость глаз, головные боли и снижает эффективность работы.

Рекомендации:

– естественный свет: Рабочий стол располагайте так, чтобы свет из окна падал слева;

– искусственное освещение;

1) общая освещенность: 300–500 лк;
2) равномерное распределение света. Избегайте бликов на экране;
3) используйте лампы теплого спектра (4000–6500 K);
4) люминесцентные лампы должны быть оснащены рассеивателями.

2.3 Режим работы и отдыха

Регулярные перерывы и физическая активность — залог сохранения здоровья.

Физические требования:

– правильная поза: спина прямая, экран на уровне глаз, расстояние до монитора — 50–70 см;

– паждые 60–90 минут делайте перерыв на 10–15 минут: разминка для глаз, шеи и спины.

Психологические аспекты:

– используйте техники тайм-менеджмента (например, Pomodoro) для снижения стресса;

– избегайте переутомления: нормируйте рабочие часы.

Организационные меры:

– планируйте задачи так, чтобы чередовать работу за компьютером с другими видами активности;

– отдыхайте в выходные дни без использования гаджетов.

Соблюдение норм безопасности при работе за компьютером не только сохраняет здоровье, но и повышает продуктивность. Организуйте эргономичное пространство, следите за освещением, делайте перерывы и используйте специализированные программы. Помните: профилактика всегда проще, чем лечение последствий [7].

Заключение

В ходе выполнения данной выпускной квалификационной работы было разработано Android приложение “УчимПрограммирование”, реализующее основной функционал мобильного приложения для изучения программирования. Приложение включает модули регистрации и аутентификации пользователей, каталог языков программирования, курсов, уроков, систему тестов с выводом результата, системы “Избранное” и “История тестов”, профиль пользователя со статистикой.

Проект демонстрирует применение технологий AndroidStudio, SQLite, Kotlin и Gradle для создания Android приложения. Реализованный дизайн с белым фоном и контрастными блоками придает приложению современный вид.

Проект имеет значительный потенциал для дальнейшего развития, включая создание полноценной системы тестирования, интеграцию ежедневных заданий с системой уведомлений, расширение функционала личного кабинетас добавлением системы друзей и общедоступной статисткиой. Дальнейшая работа также может быть направлена на повышение безопасности и оптимизацию производительности.

Опыт, полученный в ходе решения поставленных задач и разработкемобильного приложения, однозначно поможет в дальнейшем трудоустройстве и развитии себя.

Список использованных источников

1. Билл Филлипс «Android. Программирование для профессионалов на Kotlin». – 407 с.

2. Денис Колисниченко. Программирование для Android. –СПб.: Питер, 2014. – 285 с.

3. Хорстманн, Кей С., Корнелл, Гари. Java – Библиотека проффесионала, том 1. Основы. 9-е изд.: Пер. с англ. – М: ООО «И.Д. Вильямс», 2014. – 864 с.

4. Лафоре Р. Структуры данных и алгоритмы в Java. 2-е изд. – СПб: Питер, 2013. – 704 с.

5. Официальная документация JavaFX. [Электронный ресурс]/ Режим доступа: https://www.oracle.com/java/technologies/javase/javafx-docs.html/, свободный (дата обращения 15.05.2024).

6. Описание шаблона проектирования MVC. [Электронный ресурс]/ Режим доступа: https://pub.dev/documentation/mvc_pattern/latest/, свободный (дата обращения 11.05.2024).

7. СанПиН 2.2.2/2.4.1340-03 «Гигиенические требования к персональным жлектронно-вычислительным машинам и организации работы».

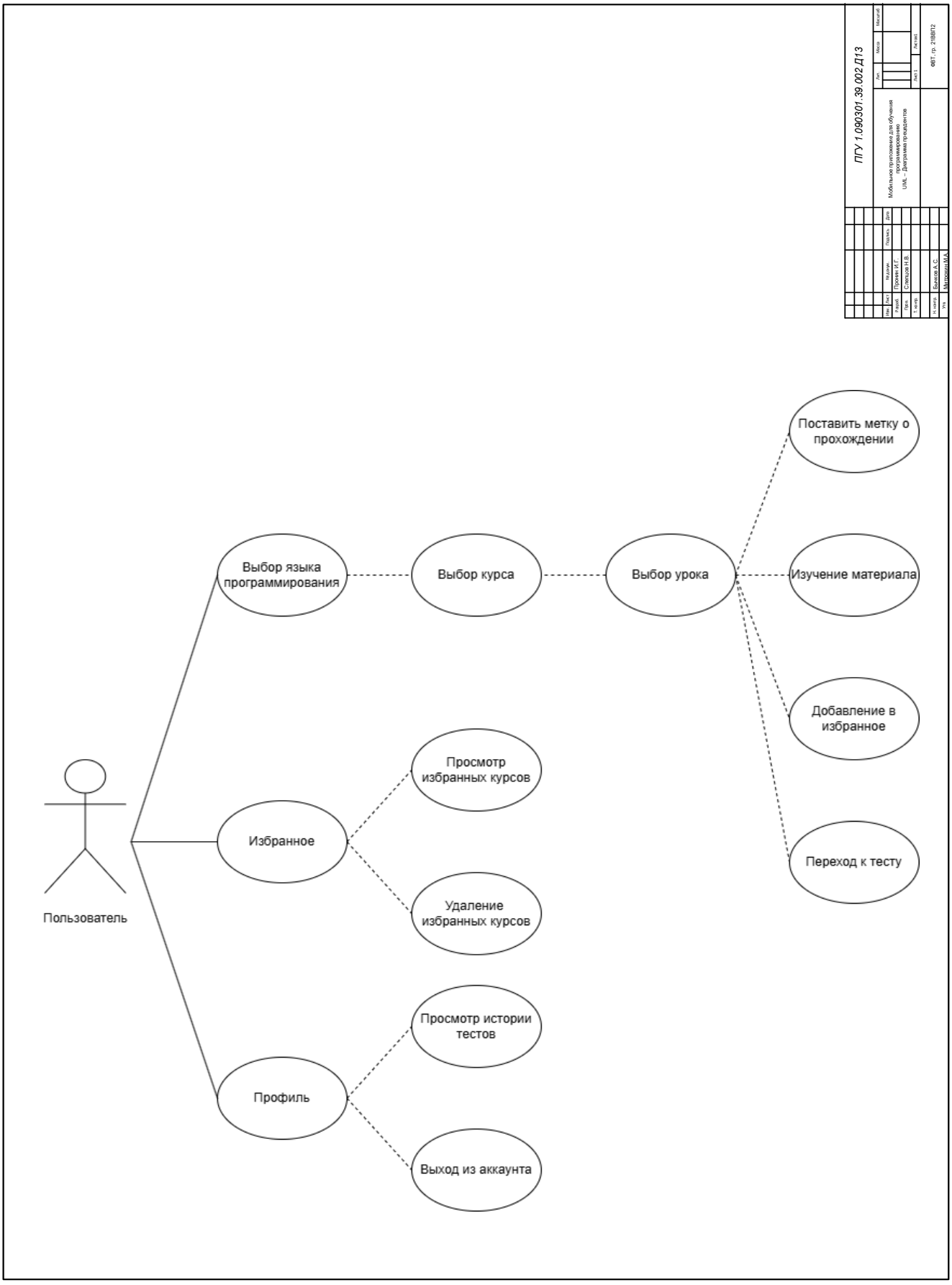
Приложение А

(Обязательное)

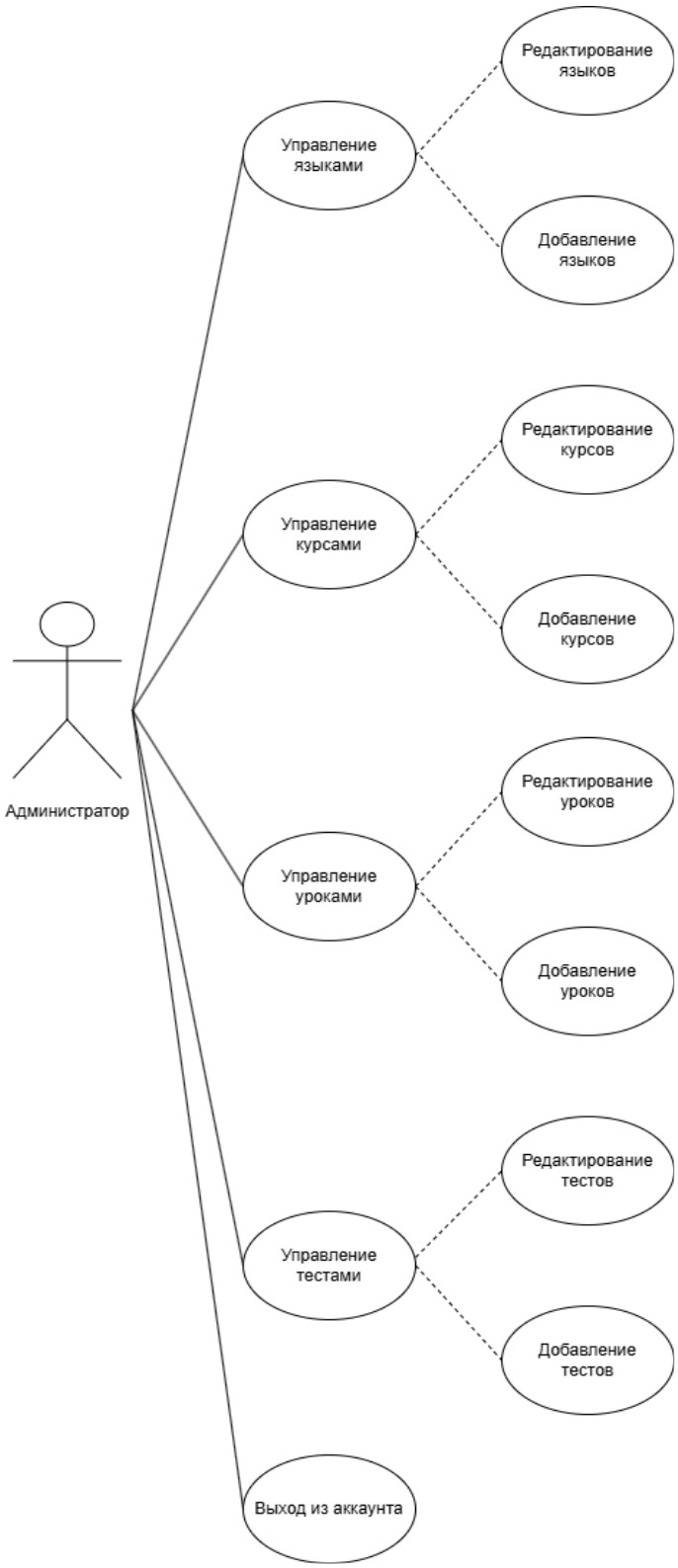
Мобильное приложение для обучения программированию

UML-диаграммы

Приложение А.1 – Диаграмма прецедентов



Приложение А.2 – Диаграмма прецедентов



ПГУ 1.090301.39.003 Д13

Мобильное приложение для обучения программированию
UML – диаграмма прецедентов

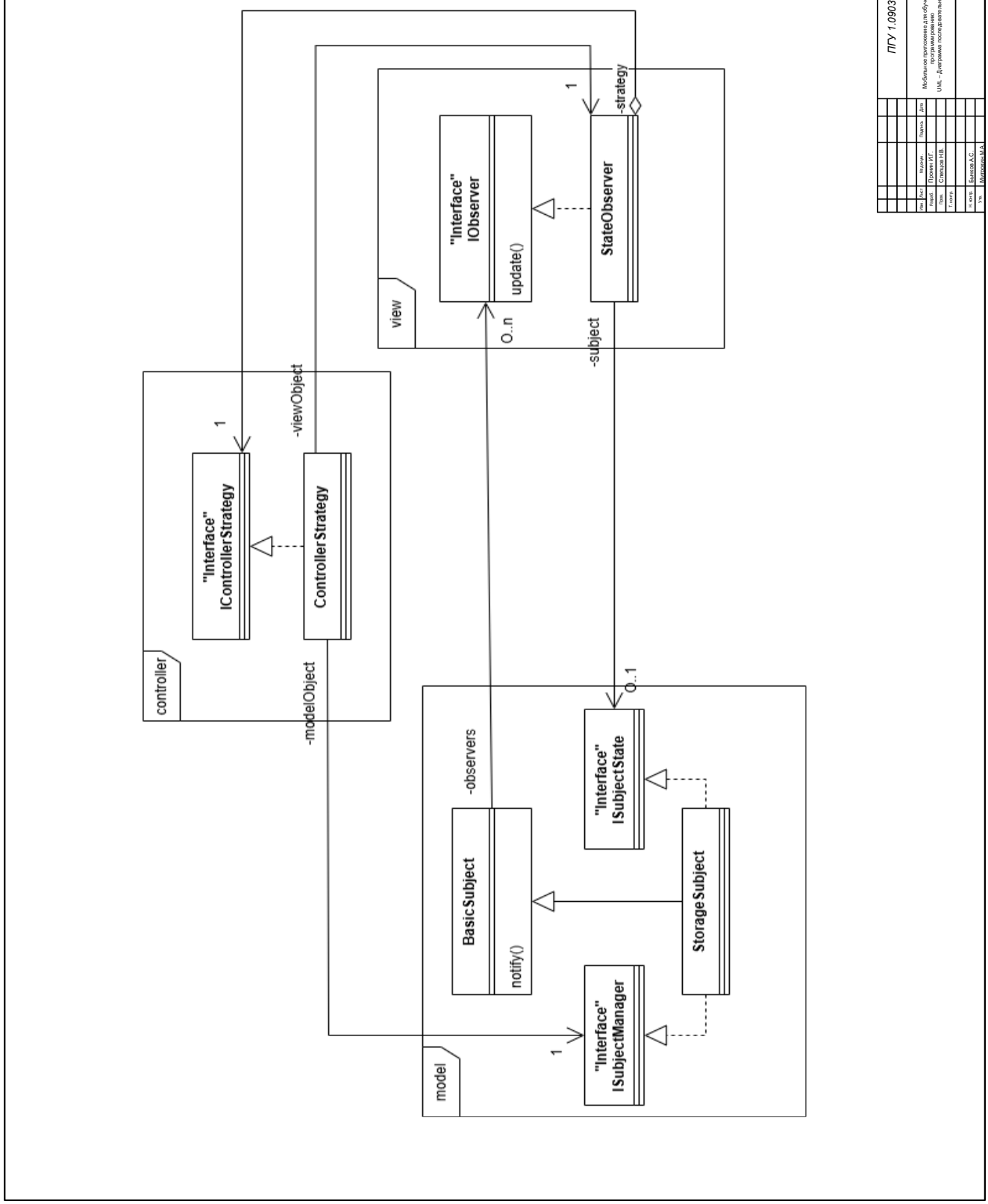
Лит. | Масса | Масштаб
Лист | Листов

ФВТ, гр. 21ВВ72

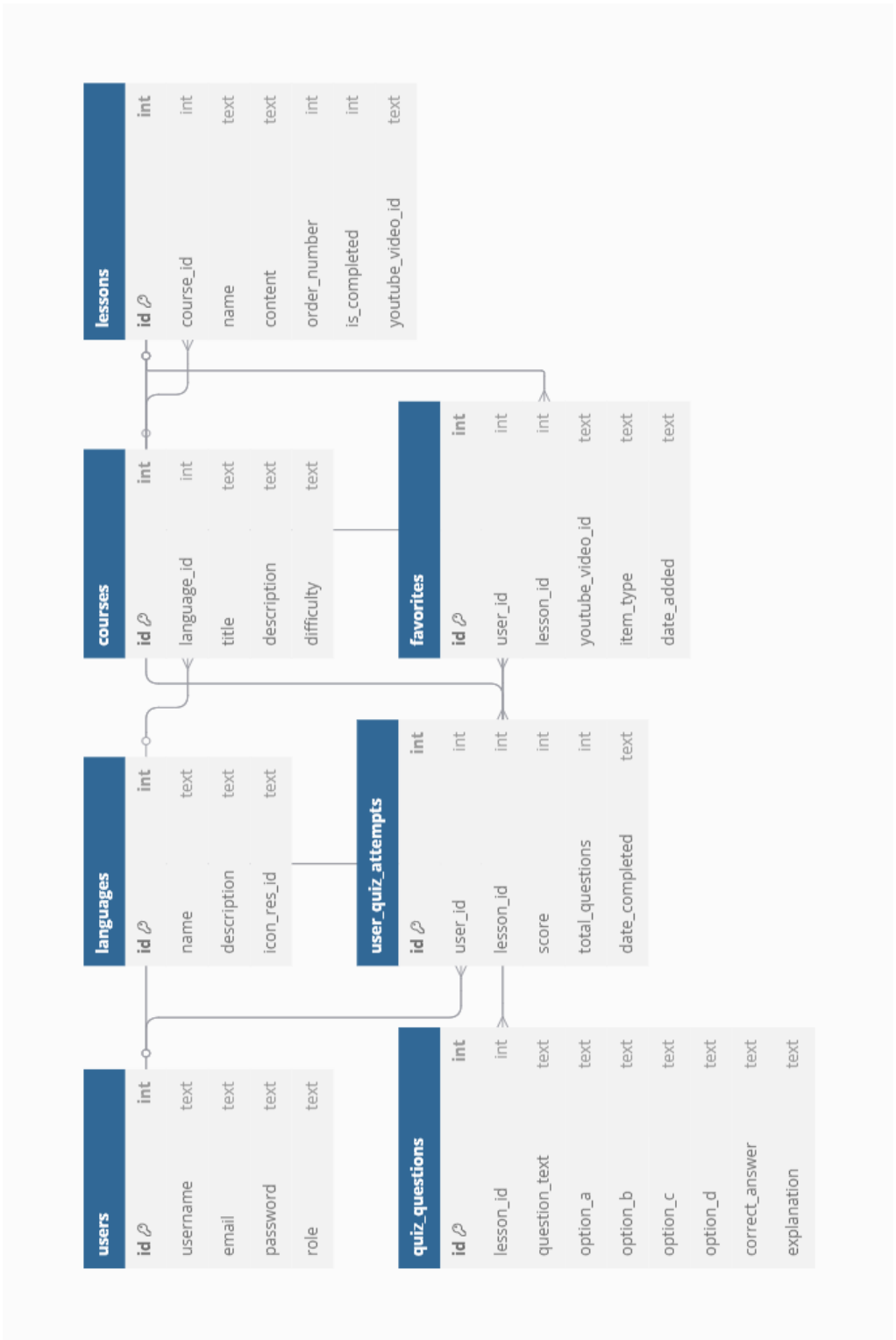
Изм. Лист № докум. Подпись Дата

Разраб. Пронин И.Г.
Пров. Сметкин Н.В.
Т.контр.
Н.контр. Бакшаев А.С.
Утв. Матюхин В.А.

Приложение А.3 – Диаграмма последовательностей



Приложение А.4 – Диаграмма классов



Приложение Б

(Обязательное)

Мобильное приложение для обучения программированию

Листинг и программы

ФайлMainActivity.kt

```
packagecom.example.uchimprogrammirovanie

import android.content.Context
import android.content.Intent
import android.os.Bundle
import android.view.Menu
import android.view.MenuItem
import androidx.activity.viewModels
import androidx.appcompat.app.AppCompatActivity
import androidx.lifecycle.Observer
import androidx.recyclerview.widget.LinearLayoutManager
import com.example.uchimprogrammirovanie.MainApplication
import com.example.uchimprogrammirovanie.auth.LoginActivity
import
com.example.uchimprogrammirovanie.databinding.ActivityMainBinding
import com.example.uchimprogrammirovanie.ui.user.CourseListActivity
import com.example.uchimprogrammirovanie.ui.user.UserLanguageAdapter
import com.example.uchimprogrammirovanie.ui.user.UserLanguageViewModel
import
com.example.uchimprogrammirovanie.ui.user.UserLanguageViewModelFactory
import com.example.uchimprogrammirovanie.ui.user.TestResultsActivity

class MainActivity : AppCompatActivity() {
    private lateinit var binding: ActivityMainBinding
    private val userLanguageViewModel: UserLanguageViewModel by
viewModels {
UserLanguageViewModelFactory((application as
MainApplication).repository)
    }
override fun onCreate(savedInstanceState: Bundle?) {
super.onCreate(savedInstanceState)
        binding = ActivityMainBinding.inflate(layoutInflater)
setContentView(binding.root)
supportActionBar?.title = "Выберитеязык"
val adapter = UserLanguageAdapter { language ->
val intent = Intent(this, CourseListActivity::class.java).apply {
putExtra(CourseListActivity.EXTRA_LANGUAGE_ID, language.id)
putExtra(CourseListActivity.EXTRA_LANGUAGE_NAME,
language.languageName)
            }
            startActivity(intent)
        }
        binding.recyclerViewLanguages.adapter = adapter
        binding.recyclerViewLanguages.layoutManager =
LinearLayoutManager(this)
userLanguageViewModel.allLanguages.observe(this, Observer {
it?.let { adapter.submitList(it) }
        })
    }
override fun onCreateOptionsMenu(menu: Menu): Boolean {
```

```
menuInflater.inflate(R.menu.main_menu, menu)
        return true
    }
override fun onOptionsItemSelected(item: MenuItem): Boolean {
return when (item.itemId) {
            R.id.action_logout -> {
logoutUser()
                true
            }
            R.id.action_view_results -> {
startActivity(Intent(this, TestResultsActivity::class.java))
                true
            }
else -> super.onOptionsItemSelected(item)
        }
    }

private fun logoutUser() {
val sharedPreferences = getSharedPreferences(LoginActivity.PREFS_NAME,
Context.MODE_PRIVATE)
with(sharedPreferences.edit()) {
remove(LoginActivity.KEY_IS_LOGGED_IN)
remove(LoginActivity.KEY_USERNAME)
remove(LoginActivity.KEY_ROLE)
apply()
        }
val intent = Intent(this, LoginActivity::class.java)
        intent.flags = Intent.FLAG_ACTIVITY_NEW_TASK or
Intent.FLAG_ACTIVITY_CLEAR_TASK
        startActivity(intent)
finishAffinity()
    }
}
```

ФайлLoginActivity.kt:

```
package com.example.uchimprogrammirovanie.auth

import android.content.Context
import android.content.Intent
import android.content.SharedPreferences
import android.os.Bundle
import android.widget.Toast
import androidx.activity.viewModels
import androidx.appcompat.app.AppCompatActivity
import com.example.uchimprogrammirovanie.MainApplication
import com.example.uchimprogrammirovanie.MainActivity
import
com.example.uchimprogrammirovanie.databinding.ActivityLoginBinding
import
com.example.uchimprogrammirovanie.ui.admin.AdminDashboardActivity
import com.example.uchimprogrammirovanie.viewmodels.AuthViewModel
import
com.example.uchimprogrammirovanie.viewmodels.AuthViewModelFactory
```

```
import com.example.uchimprogrammirovanie.viewmodels.AuthResult
import com.example.uchimprogrammirovanie.data.db.UserRoles
class LoginActivity : AppCompatActivity() {
    private lateinit var binding: ActivityLoginBinding
    private val authViewModel: AuthViewModel by viewModels {
AuthViewModelFactory((application as MainApplication).repository)
    }
    private lateinit var sharedPreferences: SharedPreferences
    companion object {
        const val PREFS_NAME = "AdminAuthPrefs"
        const val KEY_IS_LOGGED_IN = "isLoggedIn"
        const val KEY_USERNAME = "username"
        const val KEY_ROLE = "userRole"
    }
override fun onCreate(savedInstanceState: Bundle?) {
super.onCreate(savedInstanceState)
        binding = ActivityLoginBinding.inflate(layoutInflater)
setContentView(binding.root)
sharedPreferences = getSharedPreferences(PREFS_NAME,
Context.MODE_PRIVATE)
authViewModel.loginResult.observe(this) { result ->
            when (result) {
                is AuthResult.Success -> {
Toast.makeText(this, result.message, Toast.LENGTH_SHORT).show()
                    val editor = sharedPreferences.edit()
editor.putBoolean(KEY_IS_LOGGED_IN, true)
editor.putString(KEY_USERNAME, result.username)
editor.putString(KEY_ROLE, result.role)
editor.commit()

if (result.role == UserRoles.ROLE_ADMIN) {
startActivity(Intent(this, AdminDashboardActivity::class.java))
                    } else if (result.role == UserRoles.ROLE_USER) {
startActivity(Intent(this, MainActivity::class.java))
                    }
finishAffinity()
                }
                is AuthResult.Error -> {
Toast.makeText(this, result.message, Toast.LENGTH_LONG).show()
                }
                is AuthResult.Loading -> {
                    // Показатьпрогресс
                }
            }
        }
        binding.buttonLogin.setOnClickListener {
val username = binding.editTextLoginUsername.text.toString().trim()
val password = binding.editTextLoginPassword.text.toString().trim()
            authViewModel.loginUser(username, password)
        }
        binding.textViewGoToRegister.setOnClickListener {
startActivity(Intent(this, RegisterActivity::class.java))
        }
```

```
    }
}
```

ФайлSplashScreenActivity.kt:

```
package com.example.uchimprogrammirovanie

import android.annotation.SuppressLint
import android.content.Context
import android.content.Intent
import android.content.SharedPreferences
import android.os.Bundle
import android.os.Handler
import android.os.Looper
import androidx.appcompat.app.AppCompatActivity
import com.example.uchimprogrammirovanie.auth.LoginActivity
import com.example.uchimprogrammirovanie.data.db.UserRoles
import
com.example.uchimprogrammirovanie.ui.admin.AdminDashboardActivity
@SuppressLint("CustomSplashScreen")
class SplashScreenActivity : AppCompatActivity() {
    private lateinit var sharedPreferences: SharedPreferences

override fun onCreate(savedInstanceState: Bundle?) {
super.onCreate(savedInstanceState)
sharedPreferences = getSharedPreferences(LoginActivity.PREFS_NAME,
Context.MODE_PRIVATE)
Handler(Looper.getMainLooper()).postDelayed({
            val isLoggedIn =
sharedPreferences.getBoolean(LoginActivity.KEY_IS_LOGGED_IN, false)
            if (isLoggedIn) {
                val role =
sharedPreferences.getString(LoginActivity.KEY_ROLE, null)
                when (role) {
                    UserRoles.ROLE_ADMIN ->startActivity(Intent(this,
AdminDashboardActivity::class.java))
                    UserRoles.ROLE_USER ->startActivity(Intent(this,
MainActivity::class.java))
else -> startActivity(Intent(this, LoginActivity::class.java))
                }
            } else {
startActivity(Intent(this, LoginActivity::class.java))
}
finish()
        }, 1000)
    }
}
```

ФайлLessons.kt:

```
package com.example.uchimprogrammirovanie.data

data class Answer(
    val text: String,
    val isCorrect: Boolean
```

```
)
data class Question(
    val text: String,
    val answers: List<Answer>,
    val correctAnswerIndex: Int
)
data class Lesson(
    val id: Int,
    val title: String,
    val content: String,
    val test: List<Question>,
    val language: String
)
// Примерурокапо Kotlin
val lessons = listOf(
Lesson(
id = 1,
title = "Урок 1. Переменные и типы данных",
        content = """
            В языке Kotlin переменные можно объявлять с помощью
ключевых слов `val` и `var`.
            `val` используется для неизменяемых переменных (аналог
`final` в Java),
            а `var` — для изменяемых. Kotlin — язык со строгой
типизацией, но может
            автоматически выводить тип. Основные типы данных: `Int`,
`Double`, `String`,
`Boolean`, `Char`.
        """.trimIndent(),
test = listOf(
Question(
                text = "Какой ключевое слово используется для создания
неизменяемой переменной?",
answers = listOf(
Answer("var", false),
Answer("const", false),
Answer("val", true),
Answer("let", false)
                ),
                correctAnswerIndex = 2
            )
        ),
        language = "Kotlin"
),

    // Пример урока по Python
Lesson(
        id = 5,
        title = "Python Урок 1. Переменные и типы данных",
        content = """
            В Python переменные создаются при первом присваивании.
            Тип данных определяется автоматически. Основные типы:
int, float, str, bool. Например: age = 30,
```

```
            username = "Anna", is_active = True.
        """.trimIndent(),
test = listOf(
Question(
                text = "Каксоздатьпеременнуюв Python?",
answers = listOf(
Answer("var x = 5", false),
Answer("let x = 5", false),
Answer("x = 5", true),
Answer("define x = 5", false)
                ),
                correctAnswerIndex = 2
)
        ),
        language = "Python"
    ),

    // Пример урока по Java
Lesson(
        id = 9,
        title = "Java Урок 1. Переменные и типы",
        content = """
            Java — строго типизированный язык. Все переменные нужно
            объявлять с типом: int, double, String, boolean и т.д.
            Пример: int age = 25; String name = "Alex";
        """.trimIndent(),
test = listOf(
Question(
                text = "Какойтиписпользуетсядляцелыхчисел?",
answers = listOf(
Answer("float", false),
Answer("int", true),
Answer("number", false),
Answer("decimal", false)
                ),
                correctAnswerIndex = 1
            )
        ),
        language = "Java"
    )
)
```

Файлbuild.gradle.kts:

```
plugins {
alias(libs.plugins.android.application)
alias(libs.plugins.kotlin.android)
    alias(libs.plugins.google.devtools.ksp)
}

android {
namespace = "com.example.uchimprogrammirovanie"
    compileSdk = 34
```

```
    defaultConfig {
applicationId = "com.example.uchimprogrammirovanie"
        minSdk = 24 // Android 7.0 (Nougat)
        targetSdk = 34
        versionCode = 1
        versionName = "1.0"
testInstrumentationRunner = "androidx.test.runner.AndroidJUnitRunner"
    }

    buildFeatures {
        viewBinding = true
    }

    dependencies {
        implementation(libs.androidx.core.ktx)
implementation(libs.androidx.appcompat)
implementation(libs.androidx.room.runtime)
        implementation(libs.androidx.room.ktx)
        implementation(libs.androidx.lifecycle.viewmodel.ktx)
        implementation(libs.androidx.lifecycle.livedata.ktx)
}
}
```

ФайлAdminLessonAdapter:

```
package com.learnprogramming.app.adapter

import android.view.LayoutInflater
import android.view.View
import android.view.ViewGroup
import android.widget.ImageButton
import android.widget.TextView
import androidx.recyclerview.widget.RecyclerView
import com.learnprogramming.app.R
import com.learnprogramming.app.model.Lesson

class AdminLessonAdapter(
    private val lessons: MutableList<Lesson>,
    private val onEditClick: (Lesson) -> Unit,
    private val onDeleteClick: (Lesson) -> Unit
) : RecyclerView.Adapter<AdminLessonAdapter.LessonViewHolder>() {

    override fun onCreateViewHolder(parent: ViewGroup, viewType: Int):
LessonViewHolder {
        val view =
LayoutInflater.from(parent.context).inflate(R.layout.item_admin_lesson
, parent, false)
        return LessonViewHolder(view)
    }

    override fun onBindViewHolder(holder: LessonViewHolder, position:
Int) {
        val lesson = lessons[position]
```

```
        holder.bind(lesson)
    }

    override fun getItemCount(): Int = lessons.size

    fun updateLessons(newLessons: List<Lesson>) {
        lessons.clear()
        lessons.addAll(newLessons)
        notifyDataSetChanged()
    }

    inner class LessonViewHolder(itemView: View) :
RecyclerView.ViewHolder(itemView) {
        private val lessonNameTextView: TextView =
itemView.findViewById(R.id.textViewAdminLessonName)
        private val editButton: ImageButton =
itemView.findViewById(R.id.buttonEditLesson)
        private val deleteButton: ImageButton =
itemView.findViewById(R.id.buttonDeleteLesson)

        fun bind(lesson: Lesson) {
            lessonNameTextView.text = lesson.name
            // Можнодобавитьотображение ID курсаилиязыка,
еслиэтонеобходимо

            editButton.setOnClickListener { onEditClick(lesson) }
deleteButton.setOnClickListener { onDeleteClick(lesson) }
}
    }
}
```

Файл AdminQuizQuestionAdapter:

```
package com.learnprogramming.app.adapter

import android.view.LayoutInflater
import android.view.View
import android.view.ViewGroup
import android.widget.ImageButton
import android.widget.TextView
import androidx.recyclerview.widget.RecyclerView
import com.learnprogramming.app.R
import com.learnprogramming.app.model.QuizQuestion

class AdminQuizQuestionAdapter(
    private val questions: MutableList<QuizQuestion>,
    private val onEditClick: (QuizQuestion) -> Unit,
    private val onDeleteClick: (QuizQuestion) -> Unit
) :
RecyclerView.Adapter<AdminQuizQuestionAdapter.QuestionViewHolder>() {

    override fun onCreateViewHolder(parent: ViewGroup, viewType: Int):
QuestionViewHolder {
```

```
        val view =
LayoutInflater.from(parent.context).inflate(R.layout.item_admin_quiz_q
uestion, parent, false)
        return QuestionViewHolder(view)
    }

    override fun onBindViewHolder(holder: QuestionViewHolder,
position: Int) {
        holder.bind(questions[position])
    }

    override fun getItemCount(): Int = questions.size

    fun updateQuestions(newQuestions: List<QuizQuestion>) {
        questions.clear()
        questions.addAll(newQuestions)
        notifyDataSetChanged()
    }

    inner class QuestionViewHolder(itemView: View) :
RecyclerView.ViewHolder(itemView) {
        private val questionTextTextView: TextView =
itemView.findViewById(R.id.textViewAdminQuizQuestionText)
        private val editButton: ImageButton =
itemView.findViewById(R.id.buttonEditQuizQuestion)
        private val deleteButton: ImageButton =
itemView.findViewById(R.id.buttonDeleteQuizQuestion)

        fun bind(question: QuizQuestion) {
            questionTextTextView.text = question.questionText
            editButton.setOnClickListener { onEditClick(question) }
deleteButton.setOnClickListener { onDeleteClick(question) }
}
    }
}
```

Файл CourseAdapter:

```
package com.learnprogramming.app.adapter

import android.content.Context
import android.content.Intent
import android.view.LayoutInflater
import android.view.ViewGroup
import androidx.recyclerview.widget.RecyclerView
import com.learnprogramming.app.databinding.ItemCourseBinding
import com.learnprogramming.app.model.Course
import com.learnprogramming.app.ui.LessonsActivity

class CourseAdapter(
    private val context: Context,
    private val courses: List<Course>
) : RecyclerView.Adapter<CourseAdapter.CourseViewHolder>() {
```

```
    override fun onCreateViewHolder(parent: ViewGroup, viewType: Int):
CourseViewHolder {
        val binding =
ItemCourseBinding.inflate(LayoutInflater.from(parent.context), parent,
false)
        return CourseViewHolder(binding)
    }

    override fun onBindViewHolder(holder: CourseViewHolder, position:
Int) {
        val course = courses[position]
        holder.bind(course)
    }

    override fun getItemCount(): Int = courses.size

    inner class CourseViewHolder(private val binding:
ItemCourseBinding) : RecyclerView.ViewHolder(binding.root) {
        fun bind(course: Course) {
            binding.textViewCourseTitle.text = course.title
            binding.textViewCourseDescription.text =
course.description
            binding.textViewCourseDifficulty.text = "Сложность:
${course.difficulty}"

            binding.root.setOnClickListener {
val intent = Intent(context, LessonsActivity::class.java).apply {
putExtra(LessonsActivity.EXTRA_COURSE_ID, course.id)
                    putExtra(LessonsActivity.EXTRA_COURSE_NAME,
course.title)
}
context.startActivity(intent)
}
}
    }
}
```

Файл FavouriteAdapter:

```
package com.learnprogramming.app.adapter

import android.view.LayoutInflater
import android.view.View
import android.view.ViewGroup
import android.widget.ImageButton
import android.widget.ImageView
import android.widget.TextView
import androidx.recyclerview.widget.RecyclerView
import com.learnprogramming.app.R
import com.learnprogramming.app.db.DatabaseHelper
import com.learnprogramming.app.model.FavoriteItem
import java.text.ParseException
import java.text.SimpleDateFormat
import java.util.Locale
```

```
import java.util.TimeZone

class FavoriteAdapter(
    private val favoriteItems: MutableList<FavoriteItem>,
    private val dbHelper: DatabaseHelper,
    private val userId: Long,
    private val onItemAction: (FavoriteItem, String) -> Unit
) : RecyclerView.Adapter<FavoriteAdapter.FavoriteViewHolder>() {

    override fun onCreateViewHolder(parent: ViewGroup, viewType: Int):
FavoriteViewHolder {
        val view =
LayoutInflater.from(parent.context).inflate(R.layout.item_favorite,
parent, false)
        return FavoriteViewHolder(view)
    }

    override fun onBindViewHolder(holder: FavoriteViewHolder,
position: Int) {
        val item = favoriteItems[position]
        holder.bind(item)
    }

    override fun getItemCount(): Int = favoriteItems.size

    inner class FavoriteViewHolder(itemView: View) :
RecyclerView.ViewHolder(itemView) {
        private val imageViewIcon: ImageView =
itemView.findViewById(R.id.imageViewFavoriteIcon)
        private val textViewTitle: TextView =
itemView.findViewById(R.id.textViewFavoriteTitle)
        private val textViewType: TextView =
itemView.findViewById(R.id.textViewFavoriteType)
        private val textViewDateAdded: TextView =
itemView.findViewById(R.id.textViewFavoriteDateAdded)
        private val buttonRemove: ImageButton =
itemView.findViewById(R.id.buttonRemoveFromFavorites)

        fun bind(favoriteItem: FavoriteItem) {
            textViewTitle.text = favoriteItem.fullTitle
            textViewDateAdded.text = "Добавлено:
${formatDate(favoriteItem.dateAdded)}"

            when (favoriteItem.itemType) {
                "lesson" -> {
                    textViewType.text = "Тип: Урок"

imageViewIcon.setImageResource(R.drawable.ic_lesson)
                }
                "youtube_video" -> {
                    textViewType.text = "Тип: YouTube Видео"

imageViewIcon.setImageResource(R.drawable.ic_youtube)
```

```
                }
                else -> {
                    textViewType.text = "Тип: Неизвестно"

imageViewIcon.setImageResource(R.drawable.ic_help_outline)
                }
            }

            itemView.setOnClickListener {
onItemAction(favoriteItem, "open")
}

buttonRemove.setOnClickListener {
val position = adapterPosition
if (position != RecyclerView.NO_POSITION) {
                    val itemToRemove = favoriteItems[position]
                    dbHelper.removeFromFavorites(itemToRemove.id)
                    favoriteItems.removeAt(position)
                    notifyItemRemoved(position)
                    notifyItemRangeChanged(position,
favoriteItems.size)
                    onItemAction(itemToRemove, "delete")
                }
}
}

        private fun formatDate(dateString: String): String {
            return try {
                val inputFormat = SimpleDateFormat("yyyy-MM-
dd'T'HH:mm:ss'Z'", Locale.getDefault())
                inputFormat.timeZone = TimeZone.getTimeZone("UTC")
                val date = inputFormat.parse(dateString)
                val outputFormat = SimpleDateFormat("dd.MM.yyyy
HH:mm", Locale.getDefault())
                outputFormat.timeZone = TimeZone.getDefault()
                date?.let { outputFormat.format(it) } ?: dateString
            } catch (e: ParseException) {
                dateString
            }
        }
    }
}
```

Файл LanduageAdapter:

```
package com.learnprogramming.app.adapter

import android.content.Context
import android.content.Intent
import android.view.LayoutInflater
import android.view.ViewGroup
import androidx.recyclerview.widget.RecyclerView
import com.learnprogramming.app.R
import com.learnprogramming.app.databinding.ItemLanguageBinding
```

```
import com.learnprogramming.app.model.ProgrammingLanguage
import com.learnprogramming.app.ui.CoursesActivity

class LanguageAdapter(
    private val context: Context,
    private val languages: List<ProgrammingLanguage>
) : RecyclerView.Adapter<LanguageAdapter.LanguageViewHolder>() {

    override fun onCreateViewHolder(parent: ViewGroup, viewType: Int):
LanguageViewHolder {
        val binding =
ItemLanguageBinding.inflate(LayoutInflater.from(parent.context),
parent, false)
        return LanguageViewHolder(binding)
    }

    override fun onBindViewHolder(holder: LanguageViewHolder,
position: Int) {
        val language = languages[position]
        holder.bind(language)
    }

    override fun getItemCount(): Int = languages.size

    inner class LanguageViewHolder(private val binding:
ItemLanguageBinding) : RecyclerView.ViewHolder(binding.root) {
        fun bind(language: ProgrammingLanguage) {
            binding.textViewLanguageName.text = language.name
            binding.textViewLanguageDescription.text =
language.description

            language.iconResId?.let {
val resourceId = context.resources.getIdentifier(it, "drawable",
context.packageName)
                if (resourceId != 0) {

binding.imageViewLanguageIcon.setImageResource(resourceId)
                } else {

binding.imageViewLanguageIcon.setImageResource(R.mipmap.ic_launcher)
                }
} ?:
binding.imageViewLanguageIcon.setImageResource(R.mipmap.ic_launcher)

            binding.root.setOnClickListener {
val intent = Intent(context, CoursesActivity::class.java).apply {
putExtra(CoursesActivity.EXTRA_LANGUAGE_ID, language.id)
                    putExtra(CoursesActivity.EXTRA_LANGUAGE_NAME,
language.name)
}
context.startActivity(intent)
}
}
```

```
    }
}
```

Файл LessonAdapter:

```
package com.learnprogramming.app.adapter

import android.view.LayoutInflater
import android.view.ViewGroup
import androidx.recyclerview.widget.RecyclerView
import com.learnprogramming.app.databinding.ItemLessonBinding
import com.learnprogramming.app.model.Lesson

class LessonAdapter(
    private val lessons: List<Lesson>,
    private val onItemClick: (Lesson) -> Unit
) : RecyclerView.Adapter<LessonAdapter.LessonViewHolder>() {

    override fun onCreateViewHolder(parent: ViewGroup, viewType: Int):
LessonViewHolder {
        val binding =
ItemLessonBinding.inflate(LayoutInflater.from(parent.context), parent,
false)
        return LessonViewHolder(binding)
    }

    override fun onBindViewHolder(holder: LessonViewHolder, position:
Int) {
        val lesson = lessons[position]
        holder.bind(lesson)
    }

    override fun getItemCount(): Int = lessons.size

    inner class LessonViewHolder(private val binding:
ItemLessonBinding) : RecyclerView.ViewHolder(binding.root) {
        fun bind(lesson: Lesson) {
            binding.textViewLessonOrder.text = "${lesson.order}."
            binding.textViewLessonTitle.text = lesson.name
            // Тутможнодобавитьлогикудляотображенияиконки (например,
еслиурокпройден)
            //
binding.imageViewLessonAction.setImageResource(R.drawable.ic_check_cir
cle)

            binding.root.setOnClickListener {
onItemClick(lesson)
}
}
    }
}
```

Файл QuizAttemptAdapter:

```
package com.learnprogramming.app.adapter
```

```
import android.view.LayoutInflater
import android.view.ViewGroup
import androidx.recyclerview.widget.RecyclerView
import com.learnprogramming.app.databinding.ItemQuizAttemptBinding
import com.learnprogramming.app.db.DatabaseHelper
import com.learnprogramming.app.model.UserQuizAttempt
import java.text.SimpleDateFormat
import java.util.Locale
import java.util.TimeZone

class QuizAttemptAdapter(
    private val attempts: List<UserQuizAttempt>,
    private val dbHelper: DatabaseHelper //
Нужендляполученияименииурока
) : RecyclerView.Adapter<QuizAttemptAdapter.AttemptViewHolder>() {

    override fun onCreateViewHolder(parent: ViewGroup, viewType: Int):
AttemptViewHolder {
        val binding =
ItemQuizAttemptBinding.inflate(LayoutInflater.from(parent.context),
parent, false)
        return AttemptViewHolder(binding)
    }

    override fun onBindViewHolder(holder: AttemptViewHolder, position:
Int) {
        val attempt = attempts[position]
        holder.bind(attempt)
    }

    override fun getItemCount(): Int = attempts.size

    inner class AttemptViewHolder(private val binding:
ItemQuizAttemptBinding) : RecyclerView.ViewHolder(binding.root) {
        fun bind(attempt: UserQuizAttempt) {
            val lesson = dbHelper.getLessonById(attempt.lessonId)
            binding.textViewAttemptLessonName.text = lesson?.name ?:
"Урокненайден"

            val percentage = if (attempt.totalQuestions > 0)
(attempt.score * 100 / attempt.totalQuestions) else 0
            binding.textViewAttemptScore.text = "Результат:
${attempt.score}/${attempt.totalQuestions} ($percentage%)"

            try {
                val isoFormat = SimpleDateFormat("yyyy-MM-
dd'T'HH:mm:ss'Z'", Locale.getDefault())
                isoFormat.timeZone = TimeZone.getTimeZone("UTC")
                val date = isoFormat.parse(attempt.dateCompleted)
                val displayFormat = SimpleDateFormat("dd MMM yyyy,
HH:mm", Locale.getDefault())
                binding.textViewAttemptDate.text = "Дата: ${date?.let
{ displayFormat.format(it) } ?: attempt.dateCompleted}"
```

```
            } catch (e: Exception) {
                binding.textViewAttemptDate.text = "Дата:
${attempt.dateCompleted}" // Fallback to raw date if parsing fails
            }
        }
    }
}
```

Файл AuthManager:

```
package com.learnprogramming.app.auth

import android.content.Context
import android.content.SharedPreferences

object AuthManager {
    private const val PREFS_NAME = "AuthPrefs"
    private const val KEY_IS_LOGGED_IN = "isLoggedIn"
    private const val KEY_USER_ID = "userId"
    private const val KEY_USERNAME = "username"
    private const val KEY_USER_ROLE = "userRole"
    private const val KEY_USER_EMAIL = "userEmail"

    private fun getPreferences(context: Context): SharedPreferences {
        return context.getSharedPreferences(PREFS_NAME,
Context.MODE_PRIVATE)
    }

    fun setLoggedIn(context: Context, isLoggedIn: Boolean, userId:
Long = -1L, username: String? = null, role: String? = null, email:
String? = null) {
        val editor = getPreferences(context).edit()
        editor.putBoolean(KEY_IS_LOGGED_IN, isLoggedIn)
        if (isLoggedIn) {
            editor.putLong(KEY_USER_ID, userId)
            editor.putString(KEY_USERNAME, username)
            editor.putString(KEY_USER_ROLE, role)
            editor.putString(KEY_USER_EMAIL, email)
        } else {
            editor.remove(KEY_USER_ID)
            editor.remove(KEY_USERNAME)
            editor.remove(KEY_USER_ROLE)
            editor.remove(KEY_USER_EMAIL)
        }
        editor.apply()
    }

    fun isLoggedIn(context: Context): Boolean {
        return getPreferences(context).getBoolean(KEY_IS_LOGGED_IN,
false)
    }

    fun getLoggedInUserId(context: Context): Long {
```

```
        return getPreferences(context).getLong(KEY_USER_ID, -1L)
    }

    fun getLoggedInUsername(context: Context): String? {
return getPreferences(context).getString(KEY_USERNAME, null)
    }

    fun getLoggedInUserRole(context: Context): String? {
return getPreferences(context).getString(KEY_USER_ROLE, null)
    }

    fun getLoggedInUserEmail(context: Context): String? {
        return getPreferences(context).getString(KEY_USER_EMAIL, null)
    }

    fun logout(context: Context) {
        setLoggedIn(context, false)
    }
}
```

Файл LoginActivity:

```
package com.learnprogramming.app.auth

import android.content.Intent
import android.os.Bundle
import android.widget.Toast
import androidx.appcompat.app.AppCompatActivity
import com.learnprogramming.app.MainActivity // Предполагается, что
MainActivity - этоглавныйэкранпослевхода
import com.learnprogramming.app.R
import com.learnprogramming.app.databinding.ActivityLoginBinding
import com.learnprogramming.app.db.DatabaseHelper

class LoginActivity : AppCompatActivity() {

    private lateinit var binding: ActivityLoginBinding
    private lateinit var dbHelper: DatabaseHelper

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        binding = ActivityLoginBinding.inflate(layoutInflater)
        setContentView(binding.root)

        dbHelper = DatabaseHelper(this)

        // Проверка, еслипользовательужевошел, перенаправляемна
MainActivity
        if (AuthManager.isLoggedIn(this)) {
            startActivity(Intent(this, MainActivity::class.java))
            finish()
            return
        }
```

```
        binding.buttonLogin.setOnClickListener {
val username = binding.editTextLoginUsername.text.toString().trim()
            val password =
binding.editTextLoginPassword.text.toString().trim()

            if (username.isEmpty() || password.isEmpty()) {
                Toast.makeText(this, "Пожалуйста, заполнитевсеполя",
Toast.LENGTH_SHORT).show()
                return@setOnClickListener
            }

            val user = dbHelper.getUserByUsername(username)

            if (user != null && user.password == password) { //
Вреальномприложенииисравнениехэшейпаролей
                AuthManager.setLoggedIn(
                    this,
                    true,
                    user.id,
                    user.username,
                    user.role,
                    user.email
                )
                Toast.makeText(this, "Входвыполненуспешно!",
Toast.LENGTH_SHORT).show()
                val intent = Intent(this, MainActivity::class.java)
                startActivity(intent)
                finish()
            } else {
                Toast.makeText(this,
"Неверноеимяпользователяилипароль", Toast.LENGTH_SHORT).show()
            }
}

binding.textViewGoToRegister.setOnClickListener {
startActivity(Intent(this, RegisterActivity::class.java))
}
}
}
```

Файл RegisterActivity:

```
package com.learnprogramming.app.auth

import android.content.Intent
import android.os.Bundle
import android.widget.Toast
import androidx.appcompat.app.AppCompatActivity
import com.learnprogramming.app.R
import com.learnprogramming.app.databinding.ActivityRegisterBinding
import com.learnprogramming.app.db.DatabaseHelper
import com.learnprogramming.app.model.User
```

```
import com.learnprogramming.app.ui.AdminDashboardActivity
import com.learnprogramming.app.MainActivity

class RegisterActivity : AppCompatActivity() {

    private lateinit var binding: ActivityRegisterBinding
    private lateinit var dbHelper: DatabaseHelper

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        binding = ActivityRegisterBinding.inflate(layoutInflater)
        setContentView(binding.root)

        dbHelper = DatabaseHelper(this)

        binding.buttonRegister.setOnClickListener {
val username = binding.editTextRegisterUsername.text.toString().trim()
            val email =
binding.editTextRegisterEmail.text.toString().trim()
            val password =
binding.editTextRegisterPassword.text.toString().trim()
            // val confirmPassword =
binding.editTextConfirmPassword.text.toString().trim() //
Удаляемиликомментируем, еслиполеудалено

            val selectedRoleId =
binding.radioGroupRole.checkedRadioButtonId
val role = if (selectedRoleId == R.id.radioButtonAdmin) "admin" else
"user"

            if (username.isEmpty() || email.isEmpty() ||
password.isEmpty()) {
                Toast.makeText(this, "Пожалуйста, заполнитевсеполя",
Toast.LENGTH_SHORT).show()
                return@setOnClickListener
            }

            // Проверка, существуетлипользовательстакимименем
            if (dbHelper.getUserByUsername(username) != null) {
                Toast.makeText(this,
"Пользовательстакимименемужесуществует", Toast.LENGTH_SHORT).show()
                return@setOnClickListener
            }

            // TODO: Добавитьвалидацию email ипароля (сложность,
формат)

val user = User(username = username, password = password, email =
email, role = role)
            val createdUserId = dbHelper.addUser(user)
            AuthManager.setLoggedIn(this, true, createdUserId,
username, role, email)
```

```
            Toast.makeText(this, "Регистрацияуспешна",
Toast.LENGTH_SHORT).show()
            // Определяем,
кудаперейтипослерегистрациивзависимостиотроли
            val targetClass = if (role == "admin")
AdminDashboardActivity::class.java else MainActivity::class.java
val intent = Intent(this, targetClass)
            intent.flags = Intent.FLAG_ACTIVITY_NEW_TASK or
Intent.FLAG_ACTIVITY_CLEAR_TASK
startActivity(intent)
            finish()
}

binding.textViewGoToLogin.setOnClickListener {
// Простозакрываем RegisterActivity, т.к. LoginActivity
должнабытьподнейвстеке
            finish()
}
}
}
```

Файл DatabaseHelper:

```
package com.learnprogramming.app.db

import android.annotation.SuppressLint
import android.content.ContentValues
import android.content.Context
import android.database.sqlite.SQLiteDatabase
import android.database.sqlite.SQLiteOpenHelper
import com.learnprogramming.app.model.*

class DatabaseHelper(context: Context) : SQLiteOpenHelper(context,
DATABASE_NAME, null, DATABASE_VERSION) {

    companion object {
        private const val DATABASE_VERSION = 5
        private const val DATABASE_NAME = "LearnProgramming.db"

        // Таблица Users
        private const val TABLE_USERS = "users"
        private const val KEY_USER_ID = "id"
        private const val KEY_USER_USERNAME = "username"
        private const val KEY_USER_EMAIL = "email"
        private const val KEY_USER_PASSWORD = "password"
        private const val KEY_USER_ROLE = "role"

        // Таблица ProgrammingLanguages
        private const val TABLE_LANGUAGES = "languages"
        private const val KEY_LANG_ID = "id"
        private const val KEY_LANG_NAME = "name"
        private const val KEY_LANG_DESCRIPTION = "description"
        private const val KEY_LANG_ICON_RES_ID = "icon_res_id"
```

```
        // Таблица Courses
        private const val TABLE_COURSES = "courses"
        private const val KEY_COURSE_ID = "id"
        private const val KEY_COURSE_LANG_ID = "language_id"
        private const val KEY_COURSE_TITLE = "title"
        private const val KEY_COURSE_DESCRIPTION = "description"
        private const val KEY_COURSE_DIFFICULTY = "difficulty"

        // Таблица Lessons
        private const val TABLE_LESSONS = "lessons"
        private const val KEY_LESSON_ID = "id"
        private const val KEY_LESSON_COURSE_ID = "course_id"
        private const val KEY_LESSON_NAME = "name"
        private const val KEY_LESSON_CONTENT = "content"
        private const val KEY_LESSON_ORDER_NUMBER = "order_number"
        private const val KEY_LESSON_IS_COMPLETED = "is_completed"
        private const val KEY_LESSON_YOUTUBE_VIDEO_ID =
"youtube_video_id"

        // Таблица QuizQuestions
        private const val TABLE_QUIZ_QUESTIONS = "quiz_questions"
        private const val KEY_QUESTION_ID = "id"
        private const val KEY_QUESTION_LESSON_ID = "lesson_id"
        private const val KEY_QUESTION_TEXT = "question_text"
        private const val KEY_QUESTION_OPTION_A = "option_a"
        private const val KEY_QUESTION_OPTION_B = "option_b"
        private const val KEY_QUESTION_OPTION_C = "option_c"
        private const val KEY_QUESTION_OPTION_D = "option_d"
        private const val KEY_QUESTION_CORRECT_ANSWER =
"correct_answer"
        private const val KEY_QUESTION_EXPLANATION = "explanation"

        // Таблица UserQuizAttempts
        private const val TABLE_USER_QUIZ_ATTEMPTS =
"user_quiz_attempts"
        private const val KEY_ATTEMPT_ID = "id"
        private const val KEY_ATTEMPT_USER_ID = "user_id"
        private const val KEY_ATTEMPT_LESSON_ID = "lesson_id"
        private const val KEY_ATTEMPT_SCORE = "score"
        private const val KEY_ATTEMPT_TOTAL_QUESTIONS =
"total_questions"
        private const val KEY_ATTEMPT_DATE_COMPLETED =
"date_completed"

        // Таблица Favorites (Избранное)
        private const val TABLE_FAVORITES = "favorites"
        private const val KEY_FAVORITE_ID = "id"
        private const val KEY_FAVORITE_USER_ID = "user_id"
        private const val KEY_FAVORITE_LESSON_ID = "lesson_id"
        private const val KEY_FAVORITE_YOUTUBE_VIDEO_ID =
"youtube_video_id"
        private const val KEY_FAVORITE_ITEM_TYPE = "item_type"
```

```
        private const val KEY_FAVORITE_DATE_ADDED = "date_added"
    }

    override fun onCreate(db: SQLiteDatabase?) {
        val createUserTable = ("CREATE TABLE " + TABLE_USERS + "("
                + KEY_USER_ID + " INTEGER PRIMARY KEY AUTOINCREMENT,"
                + KEY_USER_USERNAME + " TEXT UNIQUE,"
                + KEY_USER_EMAIL + " TEXT UNIQUE,"
                + KEY_USER_PASSWORD + " TEXT,"
                + KEY_USER_ROLE + " TEXT DEFAULT 'user'" + ")")
        db?.execSQL(createUserTable)

        val createLanguageTable = ("CREATE TABLE " + TABLE_LANGUAGES +
"("
                + KEY_LANG_ID + " INTEGER PRIMARY KEY AUTOINCREMENT,"
                + KEY_LANG_NAME + " TEXT UNIQUE,"
                + KEY_LANG_DESCRIPTION + " TEXT,"
                + KEY_LANG_ICON_RES_ID + " TEXT" + ")")
        db?.execSQL(createLanguageTable)

        val createCourseTable = ("CREATE TABLE " + TABLE_COURSES + "("
                + KEY_COURSE_ID + " INTEGER PRIMARY KEY
AUTOINCREMENT,"
                + KEY_COURSE_LANG_ID + " INTEGER,"
                + KEY_COURSE_TITLE + " TEXT,"
                + KEY_COURSE_DESCRIPTION + " TEXT,"
                + KEY_COURSE_DIFFICULTY + " TEXT,"
                + "FOREIGN KEY(" + KEY_COURSE_LANG_ID + ") REFERENCES
" + TABLE_LANGUAGES + "(" + KEY_LANG_ID + ")" + ")")
        db?.execSQL(createCourseTable)

        val createLessonTable = ("CREATE TABLE " + TABLE_LESSONS + "("
                + KEY_LESSON_ID + " INTEGER PRIMARY KEY
AUTOINCREMENT,"
                + KEY_LESSON_COURSE_ID + " INTEGER,"
                + KEY_LESSON_NAME + " TEXT,"
                + KEY_LESSON_CONTENT + " TEXT,"
                + KEY_LESSON_ORDER_NUMBER + " INTEGER,"
                + KEY_LESSON_IS_COMPLETED + " INTEGER DEFAULT 0,"
                + KEY_LESSON_YOUTUBE_VIDEO_ID + " TEXT,"
                + "FOREIGN KEY(" + KEY_LESSON_COURSE_ID + ")
REFERENCES " + TABLE_COURSES + "(" + KEY_COURSE_ID + ")" + ")")
        db?.execSQL(createLessonTable)

        val createQuizQuestionTable = ("CREATE TABLE " +
TABLE_QUIZ_QUESTIONS + "("
                + KEY_QUESTION_ID + " INTEGER PRIMARY KEY
AUTOINCREMENT,"
                + KEY_QUESTION_LESSON_ID + " INTEGER,"
                + KEY_QUESTION_TEXT + " TEXT,"
                + KEY_QUESTION_OPTION_A + " TEXT,"
                + KEY_QUESTION_OPTION_B + " TEXT,"
                + KEY_QUESTION_OPTION_C + " TEXT,"
```

```
                + KEY_QUESTION_OPTION_D + " TEXT,"
                + KEY_QUESTION_CORRECT_ANSWER + " TEXT,"
                + KEY_QUESTION_EXPLANATION + " TEXT,"
                + "FOREIGN KEY(" + KEY_QUESTION_LESSON_ID + ")
REFERENCES " + TABLE_LESSONS + "(" + KEY_LESSON_ID + ")" + ")")
        db?.execSQL(createQuizQuestionTable)

        val createUserQuizAttemptsTable = ("CREATE TABLE " +
TABLE_USER_QUIZ_ATTEMPTS + "("
                + KEY_ATTEMPT_ID + " INTEGER PRIMARY KEY
AUTOINCREMENT,"
                + KEY_ATTEMPT_USER_ID + " INTEGER,"
                + KEY_ATTEMPT_LESSON_ID + " INTEGER,"
                + KEY_ATTEMPT_SCORE + " INTEGER,"
                + KEY_ATTEMPT_TOTAL_QUESTIONS + " INTEGER,"
                + KEY_ATTEMPT_DATE_COMPLETED + " TEXT,"
                + "FOREIGN KEY(" + KEY_ATTEMPT_USER_ID + ") REFERENCES
" + TABLE_USERS + "(" + KEY_USER_ID + "),"
                + "FOREIGN KEY(" + KEY_ATTEMPT_LESSON_ID + ")
REFERENCES " + TABLE_LESSONS + "(" + KEY_LESSON_ID + ")" + ")")
        db?.execSQL(createUserQuizAttemptsTable)

        val createFavoritesTable = ("CREATE TABLE " + TABLE_FAVORITES
+ "("
                + KEY_FAVORITE_ID + " INTEGER PRIMARY KEY
AUTOINCREMENT,"
                + KEY_FAVORITE_USER_ID + " INTEGER,"
                + KEY_FAVORITE_LESSON_ID + " INTEGER,"
                + KEY_FAVORITE_YOUTUBE_VIDEO_ID + " TEXT,"
                + KEY_FAVORITE_ITEM_TYPE + " TEXT NOT NULL,"
                + KEY_FAVORITE_DATE_ADDED + " TEXT,"
                + "FOREIGN KEY(" + KEY_FAVORITE_USER_ID + ")
REFERENCES " + TABLE_USERS + "(" + KEY_USER_ID + "),"
                + "FOREIGN KEY(" + KEY_FAVORITE_LESSON_ID + ")
REFERENCES " + TABLE_LESSONS + "(" + KEY_LESSON_ID + ")" + ")")
        db?.execSQL(createFavoritesTable)

        populateInitialData(db)
    }

    override fun onUpgrade(db: SQLiteDatabase?, oldVersion: Int,
newVersion: Int) {
        if (oldVersion < 2) {
            val createUserQuizAttemptsTable = ("CREATE TABLE " +
TABLE_USER_QUIZ_ATTEMPTS + "("
                    + KEY_ATTEMPT_ID + " INTEGER PRIMARY KEY
AUTOINCREMENT,"
                    + KEY_ATTEMPT_USER_ID + " INTEGER,"
                    + KEY_ATTEMPT_LESSON_ID + " INTEGER,"
                    + KEY_ATTEMPT_SCORE + " INTEGER,"
                    + KEY_ATTEMPT_TOTAL_QUESTIONS + " INTEGER,"
                    + KEY_ATTEMPT_DATE_COMPLETED + " TEXT,"
                    + "FOREIGN KEY(" + KEY_ATTEMPT_USER_ID + ")
```

```
REFERENCES " + TABLE_USERS + "(" + KEY_USER_ID + "),"
                    + "FOREIGN KEY(" + KEY_ATTEMPT_LESSON_ID + ")
REFERENCES " + TABLE_LESSONS + "(" + KEY_LESSON_ID + ")" + ")")
            db?.execSQL(createUserQuizAttemptsTable)
        }
        if (oldVersion < 3) {
            val createFavoritesTable = ("CREATE TABLE " +
TABLE_FAVORITES + " ("
                    + KEY_FAVORITE_ID + " INTEGER PRIMARY KEY
AUTOINCREMENT,"
                    + KEY_FAVORITE_USER_ID + " INTEGER,"
                    + KEY_FAVORITE_LESSON_ID + " INTEGER,"
                    + KEY_FAVORITE_YOUTUBE_VIDEO_ID + " TEXT,"
                    + KEY_FAVORITE_ITEM_TYPE + " TEXT NOT NULL,"
                    + KEY_FAVORITE_DATE_ADDED + " TEXT,"
                    + "FOREIGN KEY(" + KEY_FAVORITE_USER_ID + ")
REFERENCES " + TABLE_USERS + "(" + KEY_USER_ID + "),"
                    + "FOREIGN KEY(" + KEY_FAVORITE_LESSON_ID + ")
REFERENCES " + TABLE_LESSONS + "(" + KEY_LESSON_ID + ")" + ")")
            db?.execSQL(createFavoritesTable)
        }
        if (oldVersion < 4) { // Обновлениедлядобавления
youtube_video_id
            db?.execSQL("ALTER TABLE " + TABLE_LESSONS + " ADD COLUMN
" + KEY_LESSON_YOUTUBE_VIDEO_ID + " TEXT")
        }
        if (oldVersion < 5) { //
Обновлениедлядобавленияролипользователя
            db?.execSQL("ALTER TABLE " + TABLE_USERS + " ADD COLUMN "
+ KEY_USER_ROLE + " TEXT DEFAULT 'user'")
        }
    }

    private fun populateInitialData(db: SQLiteDatabase?) {
    // --- Методыдляработыс Users ---
    fun addUser(user: User): Long {
        val db = this.writableDatabase
val values = ContentValues()
        values.put(KEY_USER_USERNAME, user.username)
        values.put(KEY_USER_EMAIL, user.email)
        values.put(KEY_USER_PASSWORD, user.password)
        values.put(KEY_USER_ROLE, user.role)

        val id = db.insert(TABLE_USERS, null, values)
        db.close()
        return id
    }

    fun getUserByUsername(username: String): User? {
val db = this.readableDatabase
val cursor = db.query(
            TABLE_USERS, arrayOf(KEY_USER_ID, KEY_USER_USERNAME,
KEY_USER_EMAIL, KEY_USER_PASSWORD, KEY_USER_ROLE),
```

```
            KEY_USER_USERNAME + "=?", arrayOf(username), null, null,
null, null
        )
        var user: User? = null
        if (cursor != null && cursor.moveToFirst()) {
            user = User(
                id =
cursor.getLong(cursor.getColumnIndexOrThrow(KEY_USER_ID)),
                username =
cursor.getString(cursor.getColumnIndexOrThrow(KEY_USER_USERNAME)),
                email =
cursor.getString(cursor.getColumnIndexOrThrow(KEY_USER_EMAIL)),
                password =
cursor.getString(cursor.getColumnIndexOrThrow(KEY_USER_PASSWORD)),
                role =
cursor.getString(cursor.getColumnIndexOrThrow(KEY_USER_ROLE))
            )
            cursor.close()
        }
        db.close()
        return user
    }

    private fun addLanguage(db: SQLiteDatabase?, language:
ProgrammingLanguage): Long {
        val values = ContentValues()
        values.put(KEY_LANG_NAME, language.name)
        values.put(KEY_LANG_DESCRIPTION, language.description)
        values.put(KEY_LANG_ICON_RES_ID, language.iconResId)
        return db?.insert(TABLE_LANGUAGES, null, values) ?: -1L
    }

    private fun addCourse(db: SQLiteDatabase?, course: Course): Long {
        val values = ContentValues()
        values.put(KEY_COURSE_LANG_ID, course.languageId)
        values.put(KEY_COURSE_TITLE, course.title)
        values.put(KEY_COURSE_DESCRIPTION, course.description)
        values.put(KEY_COURSE_DIFFICULTY, course.difficulty)
        return db?.insert(TABLE_COURSES, null, values) ?: -1L
    }

    private fun addLessonInternal(db: SQLiteDatabase?, lesson:
Lesson): Long {
        val values = ContentValues()
        values.put(KEY_LESSON_COURSE_ID, lesson.courseId)
        values.put(KEY_LESSON_NAME, lesson.name)
        values.put(KEY_LESSON_CONTENT, lesson.content)
        values.put(KEY_LESSON_ORDER_NUMBER, lesson.order)
        values.put(KEY_LESSON_IS_COMPLETED, if (lesson.isCompleted) 1
else 0)
        values.put(KEY_LESSON_YOUTUBE_VIDEO_ID, lesson.youtubeVideoId)
        return db?.insert(TABLE_LESSONS, null, values) ?: -1L
    }
```

```
    fun addPublicQuizQuestion(question: QuizQuestion): Long { //
Измененос private addQuizQuestion
        val db = this.writableDatabase // Добавлено, если
addQuizQuestion был private
        val values = ContentValues()
        values.put(KEY_QUESTION_LESSON_ID, question.lessonId)
        values.put(KEY_QUESTION_TEXT, question.questionText)
        values.put(KEY_QUESTION_OPTION_A, question.optionA)
        values.put(KEY_QUESTION_OPTION_B, question.optionB)
        values.put(KEY_QUESTION_OPTION_C, question.optionC)
        values.put(KEY_QUESTION_OPTION_D, question.optionD)
        values.put(KEY_QUESTION_CORRECT_ANSWER,
question.correctAnswer)
        values.put(KEY_QUESTION_EXPLANATION, question.explanation)
        val id = db?.insert(TABLE_QUIZ_QUESTIONS, null, values) ?: -1L
// Добавлено, если addQuizQuestion был private
        // db.close() // Незакрываемздесь,
еслииспользуетсявдругомместеиливцепочкевызовов
        return id
    }

    fun getQuizQuestionById(questionId: Int): QuizQuestion? {
val db = this.readableDatabase
var question: QuizQuestion? = null
        val cursor = db.query(
            TABLE_QUIZ_QUESTIONS, null, "$KEY_QUESTION_ID = ?",
arrayOf(questionId.toString()), null, null, null
        )
        if (cursor.moveToFirst()) {
            question = QuizQuestion(
                id =
cursor.getInt(cursor.getColumnIndexOrThrow(KEY_QUESTION_ID)),
                lessonId =
cursor.getInt(cursor.getColumnIndexOrThrow(KEY_QUESTION_LESSON_ID)),
                questionText =
cursor.getString(cursor.getColumnIndexOrThrow(KEY_QUESTION_TEXT)),
                optionA =
cursor.getString(cursor.getColumnIndexOrThrow(KEY_QUESTION_OPTION_A)),
                optionB =
cursor.getString(cursor.getColumnIndexOrThrow(KEY_QUESTION_OPTION_B)),
                optionC =
cursor.getString(cursor.getColumnIndexOrThrow(KEY_QUESTION_OPTION_C)),
                optionD =
cursor.getString(cursor.getColumnIndexOrThrow(KEY_QUESTION_OPTION_D)),
                correctAnswer =
cursor.getString(cursor.getColumnIndexOrThrow(KEY_QUESTION_CORRECT_ANS
WER)),
                explanation =
cursor.getString(cursor.getColumnIndexOrThrow(KEY_QUESTION_EXPLANATION
))
            )
        }
```

```
        cursor.close()
        db.close()
        return question
    }

    fun updateQuizQuestion(question: QuizQuestion): Int {
        val db = this.writableDatabase
val values = ContentValues().apply {
put(KEY_QUESTION_LESSON_ID, question.lessonId)
            put(KEY_QUESTION_TEXT, question.questionText)
            put(KEY_QUESTION_OPTION_A, question.optionA)
            put(KEY_QUESTION_OPTION_B, question.optionB)
            put(KEY_QUESTION_OPTION_C, question.optionC)
            put(KEY_QUESTION_OPTION_D, question.optionD)
            put(KEY_QUESTION_CORRECT_ANSWER, question.correctAnswer)
            put(KEY_QUESTION_EXPLANATION, question.explanation)
}
val result = db.update(TABLE_QUIZ_QUESTIONS, values, "$KEY_QUESTION_ID
= ?", arrayOf(question.id.toString()))
        db.close()
        return result
    }

    fun getAllProgrammingLanguages(): List<ProgrammingLanguage> {
        val languages = mutableListOf<ProgrammingLanguage>()
        val selectQuery = "SELECT * FROM " + TABLE_LANGUAGES
        val db = this.readableDatabase
val cursor = db.rawQuery(selectQuery, null)

        if (cursor.moveToFirst()) {
            do {
                val language = ProgrammingLanguage(
                    id =
cursor.getLong(cursor.getColumnIndexOrThrow(KEY_LANG_ID)),
                    name =
cursor.getString(cursor.getColumnIndexOrThrow(KEY_LANG_NAME)),
                    description =
cursor.getString(cursor.getColumnIndexOrThrow(KEY_LANG_DESCRIPTION)),
                    iconResId =
cursor.getString(cursor.getColumnIndexOrThrow(KEY_LANG_ICON_RES_ID))
                )
                languages.add(language)
            } while (cursor.moveToNext())
        }
        cursor.close()
        db.close()
        return languages
    }

    fun getCoursesForLanguage(languageId: Long): List<Course> {
        val courses = mutableListOf<Course>()
        val db = this.readableDatabase
val cursor = db.query(
```

```
            TABLE_COURSES, null, KEY_COURSE_LANG_ID + "=?",
arrayOf(languageId.toString()), null, null, KEY_COURSE_TITLE
        )

        if (cursor.moveToFirst()) {
            do {
                val course = Course(
                    id =
cursor.getLong(cursor.getColumnIndexOrThrow(KEY_COURSE_ID)),
                    languageId =
cursor.getLong(cursor.getColumnIndexOrThrow(KEY_COURSE_LANG_ID)),
                    title =
cursor.getString(cursor.getColumnIndexOrThrow(KEY_COURSE_TITLE)),
                    description =
cursor.getString(cursor.getColumnIndexOrThrow(KEY_COURSE_DESCRIPTION))
,
                    difficulty =
cursor.getString(cursor.getColumnIndexOrThrow(KEY_COURSE_DIFFICULTY))
                )
                courses.add(course)
            } while (cursor.moveToNext())
        }
        cursor.close()
        db.close()
        return courses
    }

    fun getLessonsForCourse(courseId: Long): List<Lesson> {
        val lessons = mutableListOf<Lesson>()
        val db = this.readableDatabase
val cursor = db.query(
            TABLE_LESSONS,
arrayOf(KEY_LESSON_ID, KEY_LESSON_COURSE_ID, KEY_LESSON_NAME,
KEY_LESSON_CONTENT, KEY_LESSON_ORDER_NUMBER, KEY_LESSON_IS_COMPLETED,
KEY_LESSON_YOUTUBE_VIDEO_ID),
            "$KEY_LESSON_COURSE_ID = ?",
arrayOf(courseId.toString()),
            null, null, "$KEY_LESSON_ORDER_NUMBER ASC"
        )

        if (cursor.moveToFirst()) {
            do {
                val lesson = Lesson(
                    id =
cursor.getInt(cursor.getColumnIndexOrThrow(KEY_LESSON_ID)),
                    courseId =
cursor.getInt(cursor.getColumnIndexOrThrow(KEY_LESSON_COURSE_ID)),
                    name =
cursor.getString(cursor.getColumnIndexOrThrow(KEY_LESSON_NAME)),
                    content =
cursor.getString(cursor.getColumnIndexOrThrow(KEY_LESSON_CONTENT)),
                    order =
cursor.getInt(cursor.getColumnIndexOrThrow(KEY_LESSON_ORDER_NUMBER)),
```

```
                    isCompleted =
cursor.getInt(cursor.getColumnIndexOrThrow(KEY_LESSON_IS_COMPLETED))
== 1,
                    youtubeVideoId =
cursor.getString(cursor.getColumnIndexOrThrow(KEY_LESSON_YOUTUBE_VIDEO
_ID))
                )
                lessons.add(lesson)
            } while (cursor.moveToNext())
        }
        cursor.close()
        // db.close() // Незакрываемздесь,
еслииспользуетсявдругомместе
        return lessons
    }

    fun getQuizQuestionsForLesson(lessonId: Int): List<QuizQuestion> {
        val questions = mutableListOf<QuizQuestion>()
        val db = this.readableDatabase
val cursor = db.query(
            TABLE_QUIZ_QUESTIONS, null, "$KEY_QUESTION_LESSON_ID = ?",
arrayOf(lessonId.toString()), null, null, null
        )

        if (cursor.moveToFirst()) {
            do {
                val question = QuizQuestion(
                    id =
cursor.getInt(cursor.getColumnIndexOrThrow(KEY_QUESTION_ID)),
                    lessonId =
cursor.getInt(cursor.getColumnIndexOrThrow(KEY_QUESTION_LESSON_ID)),
                    questionText =
cursor.getString(cursor.getColumnIndexOrThrow(KEY_QUESTION_TEXT)),
                    optionA =
cursor.getString(cursor.getColumnIndexOrThrow(KEY_QUESTION_OPTION_A)),
                    optionB =
cursor.getString(cursor.getColumnIndexOrThrow(KEY_QUESTION_OPTION_B)),
                    optionC =
cursor.getString(cursor.getColumnIndexOrThrow(KEY_QUESTION_OPTION_C)),
                    optionD =
cursor.getString(cursor.getColumnIndexOrThrow(KEY_QUESTION_OPTION_D)),
                    correctAnswer =
cursor.getString(cursor.getColumnIndexOrThrow(KEY_QUESTION_CORRECT_ANS
WER)),
                    explanation =
cursor.getString(cursor.getColumnIndexOrThrow(KEY_QUESTION_EXPLANATION
))
                )
                questions.add(question)
            } while (cursor.moveToNext())
        }
        cursor.close()
        db.close()
```

```
        return questions
    }

    fun addLesson(lesson: Lesson): Long {
        val db = this.writableDatabase
val values = ContentValues().apply {
put(KEY_LESSON_COURSE_ID, lesson.courseId)
            put(KEY_LESSON_NAME, lesson.name)
            put(KEY_LESSON_CONTENT, lesson.content)
            put(KEY_LESSON_ORDER_NUMBER, lesson.order)
            put(KEY_LESSON_IS_COMPLETED, if (lesson.isCompleted) 1
else 0)
            put(KEY_LESSON_YOUTUBE_VIDEO_ID, lesson.youtubeVideoId)
}
val id = db.insert(TABLE_LESSONS, null, values)
        return id
    }

    fun getLessonById(lessonId: Int): Lesson? { // Changed from Long
to Int for lessonId
        val db = this.readableDatabase
var lesson: Lesson? = null
        val cursor = db.query(
            TABLE_LESSONS,
arrayOf(KEY_LESSON_ID, KEY_LESSON_COURSE_ID, KEY_LESSON_NAME,
KEY_LESSON_CONTENT, KEY_LESSON_ORDER_NUMBER, KEY_LESSON_IS_COMPLETED,
KEY_LESSON_YOUTUBE_VIDEO_ID),
            "$KEY_LESSON_ID = ?",
arrayOf(lessonId.toString()), // lessonId is Int
            null, null, null
        )

        if (cursor.moveToFirst()) {
            lesson = Lesson(
                id =
cursor.getInt(cursor.getColumnIndexOrThrow(KEY_LESSON_ID)),
                courseId =
cursor.getInt(cursor.getColumnIndexOrThrow(KEY_LESSON_COURSE_ID)),
                name =
cursor.getString(cursor.getColumnIndexOrThrow(KEY_LESSON_NAME)),
                content =
cursor.getString(cursor.getColumnIndexOrThrow(KEY_LESSON_CONTENT)),
                order =
cursor.getInt(cursor.getColumnIndexOrThrow(KEY_LESSON_ORDER_NUMBER)),
                isCompleted =
cursor.getInt(cursor.getColumnIndexOrThrow(KEY_LESSON_IS_COMPLETED))
== 1,
                youtubeVideoId =
cursor.getString(cursor.getColumnIndexOrThrow(KEY_LESSON_YOUTUBE_VIDEO
_ID))
            )
        }
        cursor.close()
```

```
        return lesson
    }
     // Overload for Long if needed, or stick to Int. For now, let's
assume lessonId is Int globally for getLessonById.
fun getLessonById(lessonId: Long): Lesson? {
        return getLessonById(lessonId.toInt()) // Delegate to Int
version
    }

    fun updateLessonCompletionStatus(lessonId: Int, isCompleted:
Boolean): Int {
        val db = this.writableDatabase
val values = ContentValues().apply {
put(KEY_LESSON_IS_COMPLETED, if (isCompleted) 1 else 0)
}
val rowsAffected = db.update(TABLE_LESSONS, values, "$KEY_LESSON_ID =
?", arrayOf(lessonId.toString()))
        return rowsAffected
    }

    fun addUserQuizAttempt(userId: Long, lessonId: Int, score: Int,
totalQuestions: Int): Long {
        val db = this.writableDatabase
val values = ContentValues()
        values.put(KEY_ATTEMPT_USER_ID, userId)
        values.put(KEY_ATTEMPT_LESSON_ID, lessonId)
        values.put(KEY_ATTEMPT_SCORE, score)
        values.put(KEY_ATTEMPT_TOTAL_QUESTIONS, totalQuestions)

        val sdf = java.text.SimpleDateFormat("yyyy-MM-
dd'T'HH:mm:ss'Z'", java.util.Locale.getDefault())
        sdf.timeZone = java.util.TimeZone.getTimeZone("UTC")
        values.put(KEY_ATTEMPT_DATE_COMPLETED,
sdf.format(java.util.Date()))

        val id = db.insert(TABLE_USER_QUIZ_ATTEMPTS, null, values)
        return id
    }

    fun getUserQuizAttempts(userId: Long): List<UserQuizAttempt> {
        val attempts = mutableListOf<UserQuizAttempt>()
        val db = this.readableDatabase
val cursor = db.query(
            TABLE_USER_QUIZ_ATTEMPTS, null, KEY_ATTEMPT_USER_ID +
"=?",
arrayOf(userId.toString()), null, null, KEY_ATTEMPT_DATE_COMPLETED + "
DESC"
        )

        if (cursor.moveToFirst()) {
            do {
                val attempt = UserQuizAttempt(
                    id =
```

```
cursor.getLong(cursor.getColumnIndexOrThrow(KEY_ATTEMPT_ID)),
                    userId =
cursor.getLong(cursor.getColumnIndexOrThrow(KEY_ATTEMPT_USER_ID)),
                    lessonId =
cursor.getInt(cursor.getColumnIndexOrThrow(KEY_ATTEMPT_LESSON_ID)),
                    score =
cursor.getInt(cursor.getColumnIndexOrThrow(KEY_ATTEMPT_SCORE)),
                    totalQuestions =
cursor.getInt(cursor.getColumnIndexOrThrow(KEY_ATTEMPT_TOTAL_QUESTIONS
)),
                    dateCompleted =
cursor.getString(cursor.getColumnIndexOrThrow(KEY_ATTEMPT_DATE_COMPLET
ED))
                )
                attempts.add(attempt)
            } while (cursor.moveToNext())
        }
        cursor.close()
        return attempts
    }

    fun addLessonToFavorites(userId: Long, lessonId: Long): Long {
        val db = this.writableDatabase
val values = ContentValues().apply {
put(KEY_FAVORITE_USER_ID, userId)
            put(KEY_FAVORITE_LESSON_ID, lessonId)
            put(KEY_FAVORITE_ITEM_TYPE, "lesson")
            put(KEY_FAVORITE_DATE_ADDED, getCurrentDateTime())
}
val id = db.insert(TABLE_FAVORITES, null, values)
        db.close()
        return id
    }

    fun addYouTubeVideoToFavorites(userId: Long, youtubeVideoId:
String): Long {
        val db = this.writableDatabase
val values = ContentValues().apply {
put(KEY_FAVORITE_USER_ID, userId)
            put(KEY_FAVORITE_YOUTUBE_VIDEO_ID, youtubeVideoId)
            put(KEY_FAVORITE_ITEM_TYPE, "youtube_video")
            put(KEY_FAVORITE_DATE_ADDED, getCurrentDateTime())
}
val id = db.insert(TABLE_FAVORITES, null, values)
        db.close()
        return id
    }

    fun removeFromFavorites(favoriteId: Long) {
        val db = this.writableDatabase
db.delete(TABLE_FAVORITES, "$KEY_FAVORITE_ID = ?",
arrayOf(favoriteId.toString()))
        db.close()
```

```
    }

    fun isLessonInFavorites(userId: Long, lessonId: Long): Boolean {
        val db = this.readableDatabase
val cursor = db.query(
            TABLE_FAVORITES,
arrayOf(KEY_FAVORITE_ID),
            "$KEY_FAVORITE_USER_ID = ? AND $KEY_FAVORITE_LESSON_ID = ?
AND $KEY_FAVORITE_ITEM_TYPE = ?",
arrayOf(userId.toString(), lessonId.toString(), "lesson"),
            null, null, null
        )
        val exists = cursor.count > 0
        cursor.close()
        db.close()
        return exists
    }

    fun isYouTubeVideoInFavorites(userId: Long, youtubeVideoId:
String): Boolean {
        val db = this.readableDatabase
val cursor = db.query(
            TABLE_FAVORITES,
arrayOf(KEY_FAVORITE_ID),
            "$KEY_FAVORITE_USER_ID = ? AND
$KEY_FAVORITE_YOUTUBE_VIDEO_ID = ? AND $KEY_FAVORITE_ITEM_TYPE = ?",
arrayOf(userId.toString(), youtubeVideoId, "youtube_video"),
            null, null, null
        )
        val exists = cursor.count > 0
        cursor.close()
        db.close()
        return exists
    }

    @SuppressLint("Range")
    fun getFavoriteIdByLessonAndUser(userId: Long, lessonId: Long):
Long {
        val db = this.readableDatabase
var favoriteId: Long = -1
        val cursor = db.query(
            TABLE_FAVORITES,
arrayOf(KEY_FAVORITE_ID),
            "$KEY_FAVORITE_USER_ID = ? AND $KEY_FAVORITE_LESSON_ID = ?
AND $KEY_FAVORITE_ITEM_TYPE = ?",
arrayOf(userId.toString(), lessonId.toString(), "lesson"),
            null, null, null
        )
        if (cursor.moveToFirst()) {
            favoriteId =
cursor.getLong(cursor.getColumnIndex(KEY_FAVORITE_ID))
        }
        cursor.close()
```

```
        db.close()
        return favoriteId
    }

    @SuppressLint("Range")
    fun getUserFavorites(userId: Long): List<FavoriteItem> {
        val favoritesList = mutableListOf<FavoriteItem>()
        val db = this.readableDatabase
val cursor = db.query(
            TABLE_FAVORITES,
            null,
            "$KEY_FAVORITE_USER_ID = ?",
arrayOf(userId.toString()),
            null, null, "$KEY_FAVORITE_DATE_ADDED DESC"
        )

        if (cursor.moveToFirst()) {
            do {
                val favoriteId =
cursor.getLong(cursor.getColumnIndex(KEY_FAVORITE_ID))
                val lessonIdFromDb = if
(!cursor.isNull(cursor.getColumnIndex(KEY_FAVORITE_LESSON_ID)))
cursor.getLong(cursor.getColumnIndex(KEY_FAVORITE_LESSON_ID)) else
null
                val youtubeVideoId =
cursor.getString(cursor.getColumnIndex(KEY_FAVORITE_YOUTUBE_VIDEO_ID))
                val itemType =
cursor.getString(cursor.getColumnIndex(KEY_FAVORITE_ITEM_TYPE))
                val dateAdded =
cursor.getString(cursor.getColumnIndex(KEY_FAVORITE_DATE_ADDED))

                val lessonName = lessonIdFromDb?.let {
getLessonById(it)?.name }
val fullTitle = if (itemType == "lesson" && lessonName != null) {
                    lessonName
                } else if (itemType == "youtube_video" &&
youtubeVideoId != null) {
                    "YouTube: $youtubeVideoId"
                } else {
                    "Неизвестныйэлемент"
                }

                favoritesList.add(FavoriteItem(
                    id = favoriteId,
                    userId = userId,
                    lessonId = lessonIdFromDb,
                    youtubeVideoId = youtubeVideoId,
                    itemType = itemType,
                    dateAdded = dateAdded,
                    lessonName = if (itemType == "lesson") lessonName
else null,
                    fullTitle = fullTitle
                ))
```

```
            } while (cursor.moveToNext())
        }
        cursor.close()
        db.close()
        return favoritesList
    }

    private fun getCurrentDateTime(): String {
        val sdf = java.text.SimpleDateFormat("yyyy-MM-
dd'T'HH:mm:ss'Z'", java.util.Locale.getDefault())
        sdf.timeZone = java.util.TimeZone.getTimeZone("UTC")
        return sdf.format(java.util.Date())
    }

    fun getAllLessons(): List<Lesson> {
        val lessons = mutableListOf<Lesson>()
        val selectQuery = "SELECT * FROM $TABLE_LESSONS ORDER BY
$KEY_LESSON_COURSE_ID, $KEY_LESSON_ORDER_NUMBER"
        val db = this.readableDatabase
val cursor = db.rawQuery(selectQuery, null)

        if (cursor.moveToFirst()) {
            do {
                val lesson = Lesson(
                    id =
cursor.getInt(cursor.getColumnIndexOrThrow(KEY_LESSON_ID)),
                    courseId =
cursor.getInt(cursor.getColumnIndexOrThrow(KEY_LESSON_COURSE_ID)),
                    name =
cursor.getString(cursor.getColumnIndexOrThrow(KEY_LESSON_NAME)),
                    content =
cursor.getString(cursor.getColumnIndexOrThrow(KEY_LESSON_CONTENT)),
                    order =
cursor.getInt(cursor.getColumnIndexOrThrow(KEY_LESSON_ORDER_NUMBER)),
                    isCompleted =
cursor.getInt(cursor.getColumnIndexOrThrow(KEY_LESSON_IS_COMPLETED))
== 1,
                    youtubeVideoId =
cursor.getString(cursor.getColumnIndexOrThrow(KEY_LESSON_YOUTUBE_VIDEO
_ID))
                )
                lessons.add(lesson)
            } while (cursor.moveToNext())
        }
        cursor.close()
        db.close()
        return lessons
    }

    fun deleteLesson(lessonId: Int): Int {
        val db = this.writableDatabase
// Сначалаудаляемсвязанныевопросыиз quiz_questions,
чтобыизбежатьнарушения FOREIGN KEY
```

```
        db.delete(TABLE_QUIZ_QUESTIONS, "$KEY_QUESTION_LESSON_ID = ?",
arrayOf(lessonId.toString()))
        // Затемудаляемурок
        val result = db.delete(TABLE_LESSONS, "$KEY_LESSON_ID = ?",
arrayOf(lessonId.toString()))
        db.close()
        return result
    }

    fun getAllCourses(): List<Course> {
        val courses = mutableListOf<Course>()
        val selectQuery = "SELECT * FROM $TABLE_COURSES ORDER BY
$KEY_COURSE_TITLE"
        val db = this.readableDatabase
val cursor = db.rawQuery(selectQuery, null)

        if (cursor.moveToFirst()) {
            do {
                val course = Course(
                    id =
cursor.getLong(cursor.getColumnIndexOrThrow(KEY_COURSE_ID)),
                    languageId =
cursor.getLong(cursor.getColumnIndexOrThrow(KEY_COURSE_LANG_ID)),
                    title =
cursor.getString(cursor.getColumnIndexOrThrow(KEY_COURSE_TITLE)),
                    description =
cursor.getString(cursor.getColumnIndexOrThrow(KEY_COURSE_DESCRIPTION))
,
                    difficulty =
cursor.getString(cursor.getColumnIndexOrThrow(KEY_COURSE_DIFFICULTY))
                )
                courses.add(course)
            } while (cursor.moveToNext())
        }
        cursor.close()
        db.close()
        return courses
    }

    fun updateLesson(lesson: Lesson): Int {
        val db = this.writableDatabase
val values = ContentValues().apply {
put(KEY_LESSON_COURSE_ID, lesson.courseId)
            put(KEY_LESSON_NAME, lesson.name)
            put(KEY_LESSON_CONTENT, lesson.content)
            put(KEY_LESSON_ORDER_NUMBER, lesson.order)
            put(KEY_LESSON_IS_COMPLETED, if (lesson.isCompleted) 1
else 0)
            put(KEY_LESSON_YOUTUBE_VIDEO_ID, lesson.youtubeVideoId)
}
val result = db.update(TABLE_LESSONS, values, "$KEY_LESSON_ID = ?",
arrayOf(lesson.id.toString()))
        db.close()
```

```
        return result
    }

    fun deleteQuizQuestion(questionId: Int): Int {
        val db = this.writableDatabase
val result = db.delete(TABLE_QUIZ_QUESTIONS, "$KEY_QUESTION_ID = ?",
arrayOf(questionId.toString()))
        db.close()
        return result
    }
}
```

ФайлFavoritesFragment:

```
package com.learnprogramming.app.fragment

import android.os.Bundle
import android.view.LayoutInflater
import android.view.View
import android.view.ViewGroup
import android.widget.TextView
import androidx.fragment.app.Fragment
import androidx.recyclerview.widget.LinearLayoutManager
import androidx.recyclerview.widget.RecyclerView
import com.learnprogramming.app.R
import com.learnprogramming.app.adapter.FavoriteAdapter //
Будетсозданпозже
import com.learnprogramming.app.auth.AuthManager
import com.learnprogramming.app.db.DatabaseHelper
import com.learnprogramming.app.model.FavoriteItem

class FavoritesFragment : Fragment() {

    private lateinit var recyclerViewFavorites: RecyclerView
    private lateinit var textViewNoFavorites: TextView
    private lateinit var favoriteAdapter: FavoriteAdapter
    private lateinit var dbHelper: DatabaseHelper
    private var userId: Long = -1
    private var favoriteItems: List<FavoriteItem> = emptyList() //
Измененона List

    override fun onCreateView(
        inflater: LayoutInflater, container: ViewGroup?,
        savedInstanceState: Bundle?
    ): View? {
        val view = inflater.inflate(R.layout.fragment_favorites,
container, false)
        dbHelper = DatabaseHelper(requireContext())
        userId = AuthManager.getLoggedInUserId(requireContext())

        recyclerViewFavorites =
view.findViewById(R.id.recyclerViewFavorites)
        textViewNoFavorites =
view.findViewById(R.id.textViewNoFavorites)
```

```
        recyclerViewFavorites.layoutManager =
LinearLayoutManager(context)
        // Адаптербудетинициализированв onResume,
чтобыобновлятьсписокпривозвращениинафрагмент
        return view
    }

    override fun onResume() {
        super.onResume()
        loadFavorites()
    }

    private fun loadFavorites() {
        if (userId != -1L) {
            favoriteItems = dbHelper.getUserFavorites(userId) //
favoriteItems теперь List
            if (favoriteItems.isEmpty()) {
                recyclerViewFavorites.visibility = View.GONE
textViewNoFavorites.visibility = View.VISIBLE
textViewNoFavorites.text = "Визбранномпоканичегонет."
            } else {
                recyclerViewFavorites.visibility = View.VISIBLE
textViewNoFavorites.visibility = View.GONE
// favoriteAdapter ожидает MutableList, нужнобудетизменитьконструктор
FavoriteAdapter
                // илиздесьпреобразоватьв MutableList,
еслиадаптербудетегоизменять.
// Пока что, если FavoriteAdapter не меняет список, можно передать
List.
                // ИЛИ, если FavoriteAdapter требует MutableList для
удаления, то:
                // favoriteAdapter =
FavoriteAdapter(favoriteItems.toMutableList()) { favoriteItem ->
                favoriteAdapter =
FavoriteAdapter(favoriteItems.toMutableList(), dbHelper, userId) {
favoriteItem, action ->// Передаем dbHelper и userId
                    if (action == "delete") {
                        loadFavorites() // Обновляем список после
удаления
                    }
                    // TODO: Обработка клика на элемент избранного
(открытие)
}
recyclerViewFavorites.adapter = favoriteAdapter
            }
        } else {
            recyclerViewFavorites.visibility = View.GONE
textViewNoFavorites.visibility = View.VISIBLE
textViewNoFavorites.text = "Пожалуйста, войдите в аккаунт, чтобы
видеть избранное."
```

Приложение В

(Обязательное)

Мобильное приложение для обучения программированию

Презентация

Мобильное приложение для обучения программированию

Автор работы:
студент гр. 21ВВП2 Пронин И.Г.
Научный руководитель:
к.т.н., доцент каф. ВТ Слепцов Н.В.

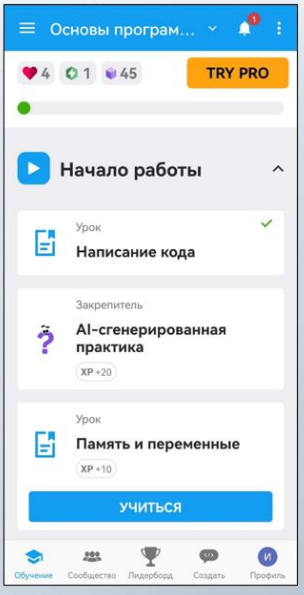
Актуальность разработки

- Востребованность дистанционного обучения
- Свободный темп обучения
- Создание собственных курсов

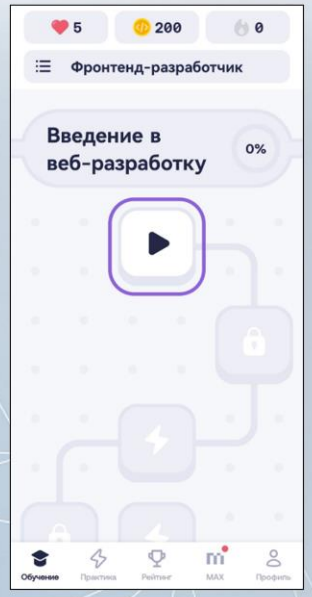
Анализ существующих аналогов

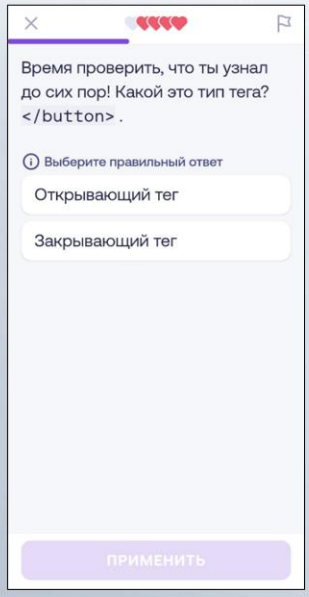
- SoloLearn
- Mimo
- CodeCademy Go

SoloLearn

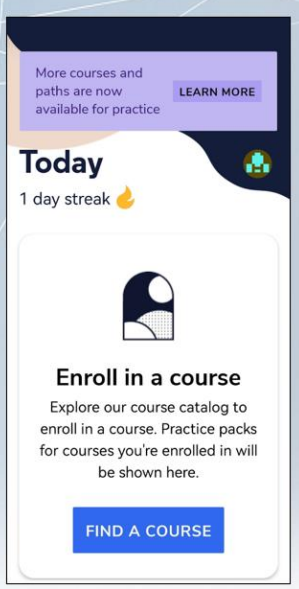


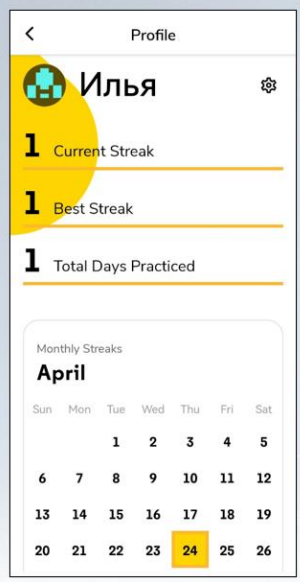
Mimo





CodeCademy GO





Приложение должно реализовывать следующий функционал:

- Авторизация и регистрация
- Доступ к разным языкам, курсам и урокам
- История пройденных тестов
- Статистика
- Избранное
- Профиль пользователя

Технологии и средства разработки

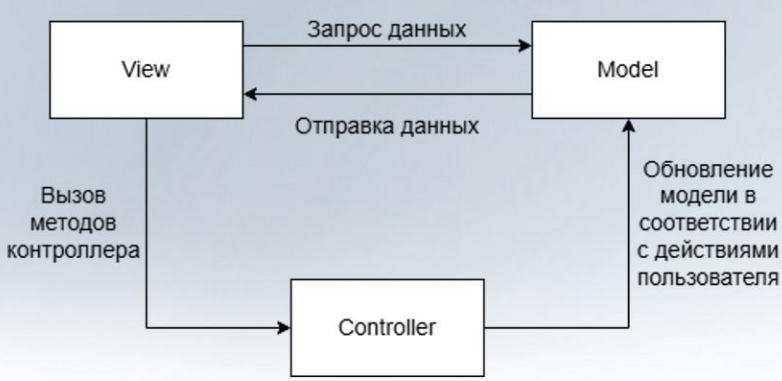
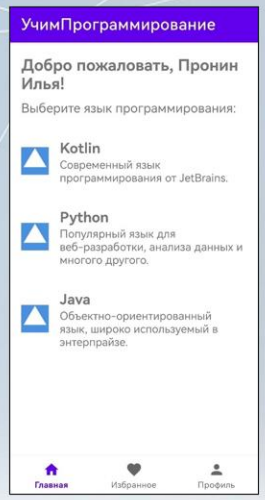
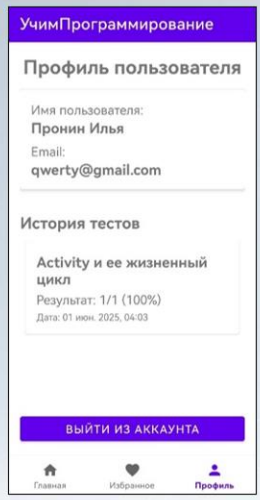
Android Studio

Kotlin

SQLite

Архитектура приложения

Приложение построено на основе архитектурного шаблона Model-View-Controller (MVC)



Основной интерфейс приложения