МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное автономное образовательное учреждение
высшего образования
«Южно-Уральский государственный университет
(национальный исследовательский университет)»
Высшая школа электроники и компьютерных наук
Кафедра системного программирования

ДОПУСТИТЬ К ЗАЩИТЕ

Заведующий кафедрой, д.ф.-м.н.,
профессор

_____ Л.Б. Соколинский

«___»_____ 2024 г.

Разработка Android-приложения «PianoMentor»

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА
ЮУрГУ – 02.03.02.2024.308-287.ВКР

Научный руководитель,
доцент кафедры СП, к.ф.-м.н.
_____ Т.Ю. Маковецкая

Автор работы,
студент группы КЭ-402
_____ Е.С. Серяков

Ученый секретарь
(нормоконтролер)
_____ И.Д. Володченко
«___» _____ 2024 г.

Челябинск, 2024 г.

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное автономное образовательное учреждение
высшего образования
«Южно-Уральский государственный университет
(национальный исследовательский университет)»
Высшая школа электроники и компьютерных наук
Кафедра системного программирования

УТВЕРЖДАЮ
Зав. кафедрой СП
_____ Л.Б. Соколинский
29.01.2024 г.

ЗАДАНИЕ
на выполнение выпускной квалификационной работы бакалавра
студенту группы КЭ-402
Серякову Егору Сергеевичу,
обучающемуся по направлению
02.03.02 «Фундаментальная информатика и информационные технологии»

1. **Тема работы** (утверждена приказом ректора от 22.04.2024 г. № 764/13-12)
Разработка Android-приложения «PianoMentor».
2. **Срок сдачи студентом законченной работы:** 03.06.2024 г.
3. **Исходные данные к работе**

3.1. Kotlin Documentation. [Электронный ресурс] URL: https://kotlinlang.org/docs/home.html (дата обращения: 01.02.2024 г.).

3.2. Филлипс Б., Стюарт К., Марсикано К., Гарднер Б. Android. Программирование для профессионалов. // Издательство Питер, 2021. – 704 с.

3.3. Мартин Р. Чистая архитектура. Искусство разработки ПО. // Издательство Питер, 2018. – 352 с.

4. **Перечень подлежащих разработке вопросов**

4.1. Провести обзор научной литературы.

4.2. Провести обзор аналогичных приложений.

4.3. Спроектировать архитектуру мобильного приложения.

4.4. Спроектировать архитектуру серверного приложения и структуру базы данных.

4.5. Разработать серверную часть системы.

4.6. Разработать клиентскую часть системы.

4.7. Протестировать готовую систему.

5. Дата выдачи задания: 29.01.2024 г.

Научный руководитель,
доцент кафедры СП, к.ф.-м.н. Т.Ю. Маковецкая

Задание принял к исполнению Е.С. Серяков

ОГЛАВЛЕНИЕ

ВВЕДЕНИЕ

Актуальность

В современном мире обучение и развитие навыков становятся все более важными. Музыкальное образование – не исключение. С ростом интереса к музыке, желанием развить свой слух и научиться играть на музыкальном инструменте, обучающие приложения становятся все более привлекательными для широких масс. Одним из самых популярных таких инструментов является фортепиано.

Обучение игре на фортепиано представляет собой комплексный процесс, включающий теорию и практику, развитие техники и музыкальной выразительности. Все это изучается годами в специализированных учреждениях, но не каждый желающий может пойти учиться в подобные места, не каждый желает профессионально развиться в музыке, а просто хочет научиться играть его любимые композиции на музыкальном инструменте, что создает спрос на быстрые, автоматизированные, «карманные» решения вроде мобильных приложений. Они могут помочь развить слух пользователя, познакомить его с теорией музык, что является первым шагом в музыкальное творчество, позволяет подхватить интерес человека и развить его в большое желание.

Постановка задачи

Целью выпускной квалификационной работы является разработка Android-приложения для обучения игре на пианино PianoMentor. Для достижения поставленной цели необходимо решить следующие задачи:

1) провести обзор аналогичных решений;
2) спроектировать мобильное приложение;
3) спроектировать серверное приложение и базу данных;
4) разработать мобильное приложение;
5) разработать серверную часть приложения;
6) протестировать полученную систему.

Структура и содержание работы

Работа состоит из введения, четырех глав, заключения, списка литературы и приложений. Объем работы составляет 79 страниц, объем списка литературы – 26 источников.

В первой главе «Анализ предметной области» описана предметная область проекта, проведен анализ существующих аналогов приложения, выведен функционал, отличающий аналоги друг от друга, получены основные критерии, которым должна соответствовать конечная система.

Вторая глава «Проектирование системы» включает в себя проведение анализа требований к разрабатываемой системе, а также описание главных особенностей архитектуры системы. Эта глава включает в себя выявление функциональных и нефункциональных требований к разрабатываемой системе, описание диаграмм вариантов использования, перечисление компонентов системы, отдельно представлены диаграммы компонентов как мобильного приложения, так и серверного приложений, спроектирована база данных.

В третей главе «Реализация» описаны особенности реализации системы. Детально рассмотрены две функции системы, представлена диаграммы последовательности для одной, для другой же реализована диаграмма деятельности.

В четвертой главе «Тестирование» проведено функциональное тестирование реализованной системы с целью проверки работоспособности системы и проверки соблюдения всех поставленных требований, а также описано проведение тестирования эргономичности (юзабилити-тестирование) мобильного приложения.

В приложениях представлены описания вариантов использований системы, отчёт об юзабилити-тестировании, диаграммы деятельности и последовательности, описывающие два отдельных процесса, протекающих в системе.

1. АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ

В первой главе рассматривается предметная область проекта, которой является обучение игре на фортепиано. Обучение игре на фортепиано является составным процессом, включающим теорию и практику, развитие техники и музыкальной выразительности. Перед тем как ученик заиграет на каком-нибудь музыкальном инструменте, он изучает различного рода упражнения на первичное развитие слуха, читает теорию музыки. Начало развития слуха включает в себя определение музыкальных интервалов разной длины и видов. Для закрепления теоретических знаний, обучающемуся даются тесты.

В сфере обучения игре на фортепиано существует несколько приложений, которые помогут начинающим пианистам развивать свои навыки. Из них можно выделить следующие.

1. Absolute pitch – приложение, позволяющее обучаться пользователю начиная с минимального уровня, заканчивая высоким уровнем со знанием теории.

2. Simply piano – приложение предлагает обучение на основе песен, которые вам нравятся. Оно также оценивает вашу игру и предоставляет обратную связь.

3. Skoove piano – это обучающее приложение для пианистов всех уровней, от начинающих до продвинутых.

Перейдём к рассмотрению каждого из трех перечисленных приложений по отдельности.

Absolute Pitch

Главная страница приложения Absolute Pitch содержит в себе графики для отображения очков, заработанных пользователем за день, а также кнопки, позволяющие открыть тот или иной модуль приложения (рисунок 1). На экране курса «Введение в интервалы», можно увидеть список уроков, преимущественно состоящих из практических занятий (рисунок 1).

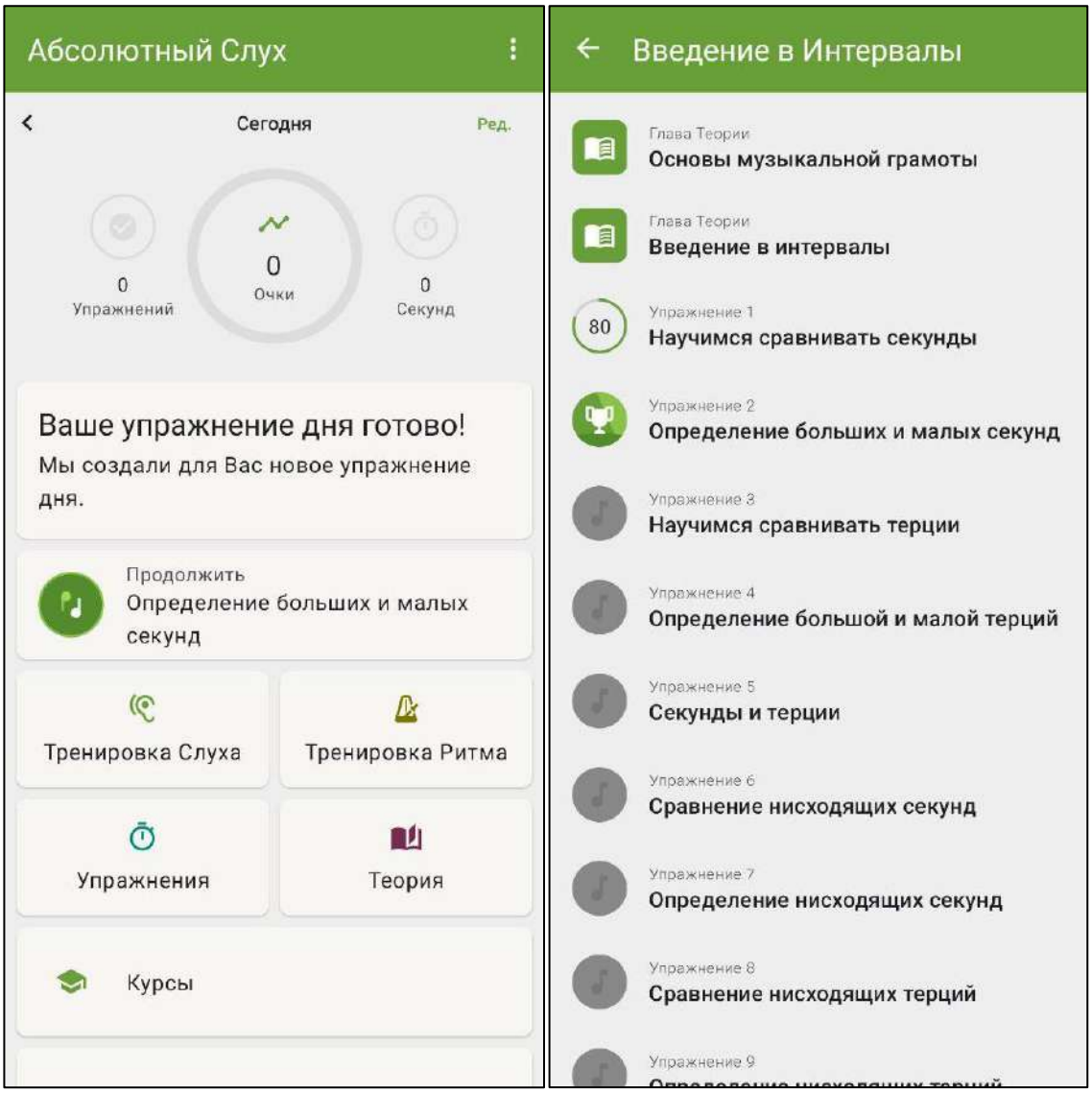

Рисунок 1 – Главная страница приложения Absolute Pitch (слева), экран курса «Введение в интервалы» (справа)

Более интересно рассмотреть, как выглядят практические занятия в данном приложении. Фактически все сводится к следующим процессам.

1. Приложение воспроизводит звуки, которые издают нужные клавиши виртуального пианино, и обозначает начальную ноту в качестве подсказки.

2. Пользователь может нажать на клавиши виртуального пианино для самостоятельного воспроизведения звуков, тем самым получая подсказку и тренируя свой слух для определения высот нот.

3. Пользователь может получить первый опыт понимания закономерностей размещения графически представленных нот на нотном стане.

На рисунке 2 можно увидеть экран практических занятий из рассматриваемого приложения.

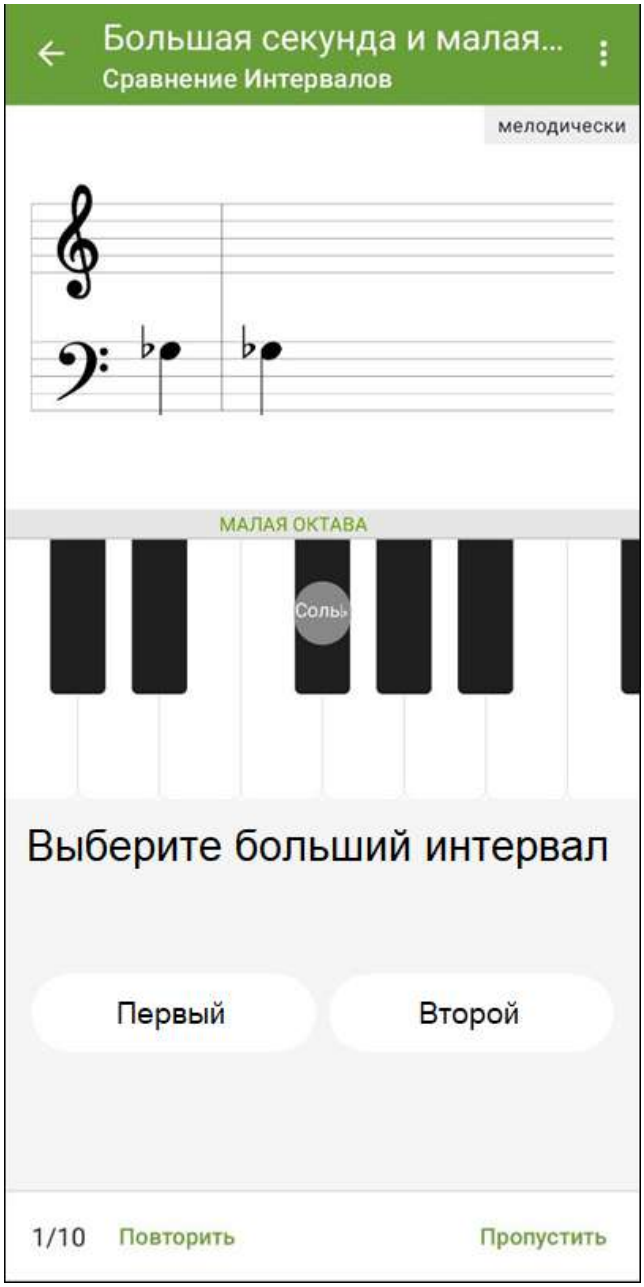

Рисунок 2 – Страница с практическим занятием приложения Absolute Pitch

Таким образом, Absolute Pitch представляет из себя проект с минималистичным дизайном, яркими цветами. Он имеет главные функции для обу-

чающего приложения: структурированные обучающие курсы как с теоретическими уроками, так и с практическими заданиями. Но при этом нет тестов для проверки знаний из лекции по теории.

Perfect Piano

Приложение Perfect Piano не предусматривает структурированного комплексного обучения пользователя, это приложение преследует главной целью позволить пользователю играть различные композиции на виртуальном фортепиано. На рисунке 3 можно увидеть, как выглядит главная страница этого приложения.

Рисунок 3 – Главная страница приложения Perfect Piano

Встроенное виртуальное фортепиано подразумевает настройку, кастомизацию с учетом пожеланий пользователя. Это является главным функционалом системы, остальные функции подразумевают лишь некоторое сетевое взаимодействие между зарегистрировавшимися пользователями, запись процесса игры и загрузка таких записей из файловой системы. На рисунке 4 можно увидеть страницу с виртуальным пианино.

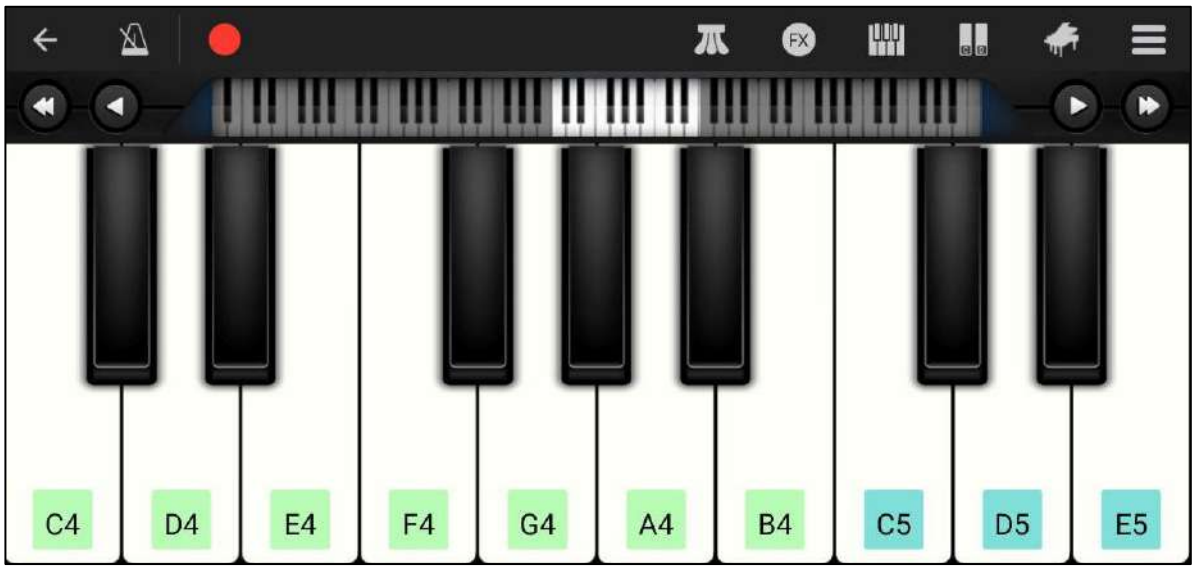

Рисунок 4 – Страница с виртуальным пианино в Perfect Piano

Следовательно, Perfect Piano скорее является приложением для пианистов, имеющих опыт и знания. Они могут показать некоторый результат в рамках, например, многопользовательской игры, могут использовать легкодоступность виртуального музыкального инструмента для скорой записи некоторых композиций.

Skoove Piano

Третье приложение под названием Skoove Piano является самым выдающимся из рассматриваемых. Главная страница и страница приложения представлена на рисунке 5.

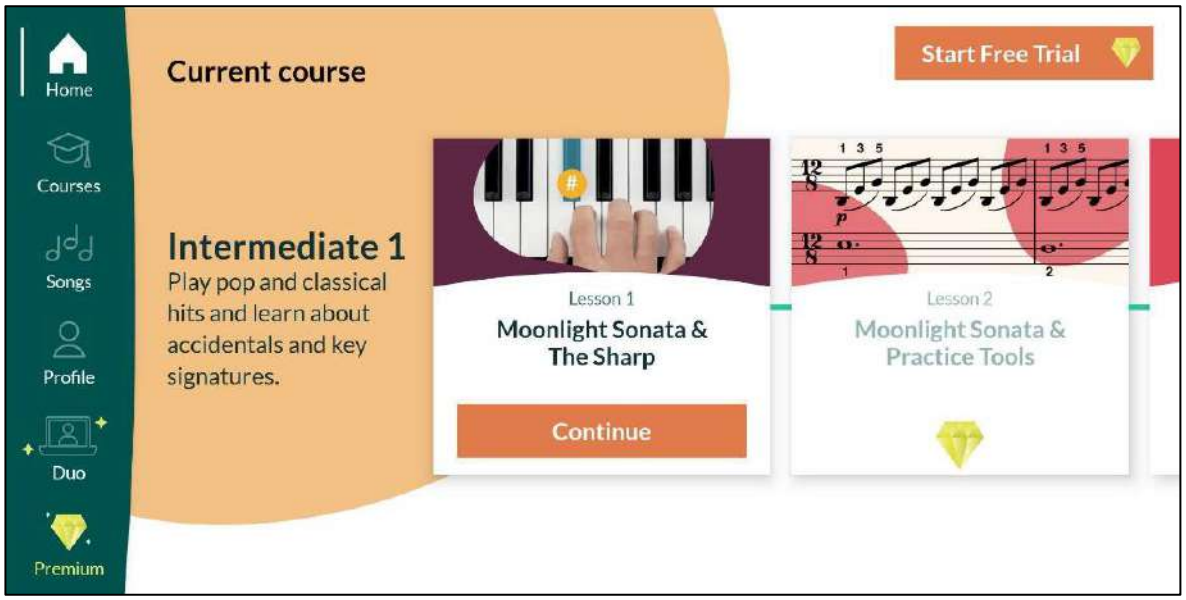

Рисунок 5 – Главная страница приложения Skoove Piano

Интересно приложение тем, что позволяет выбрать интересующие пользователя аспекты обучения и стили музыки. Помимо этого, Skoove Piano имеет разветвленный список курсов, представленный в виде графа, вершинами которого являются кнопки, направляющие пользователя на экраны с занятиями, а также содержанием видеоуроков на английском языке (рисунок 6).

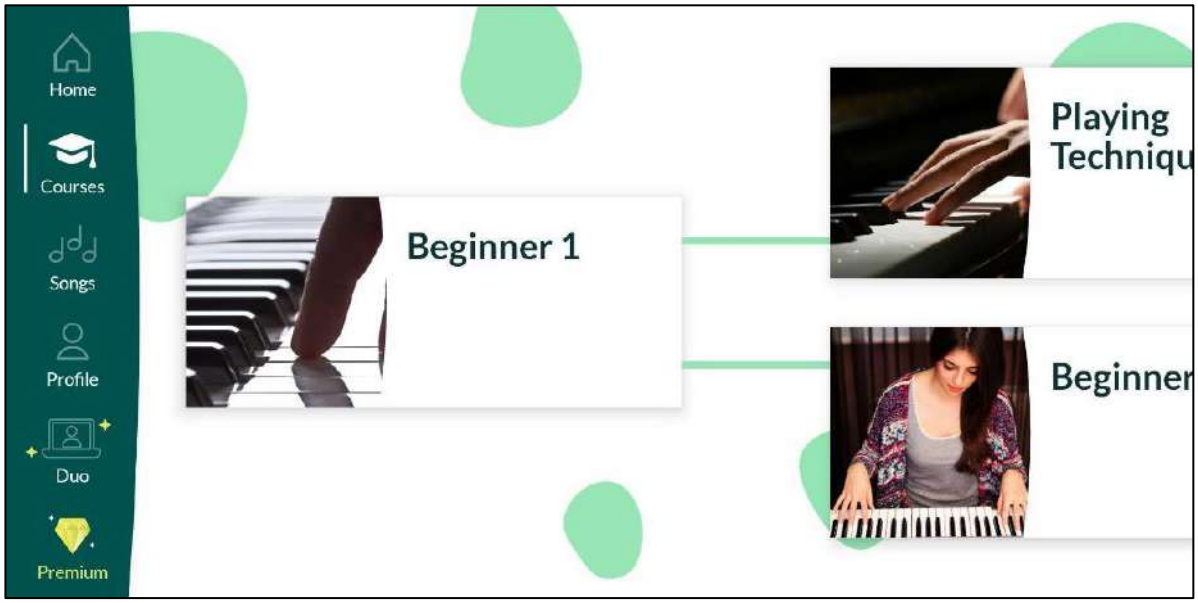

Рисунок 6 – Страница курсов приложения Skoove Piano

Исходя из увиденного в последнем из рассматриваемых приложений, можно сказать, что оно вполне может дать первые знания пользователю, но создатели ориентировались на англоговорящую аудиторию, что видно по интегрированным видеоурокам.

Рассмотренные проекты достаточно сильно отличаются друг от друга как по функционалу, так и по задачам, которые преследовали разработчики при создании своих приложений. В таблице 1 представлена информация о наличии той или иной возможности в вышеуказанных продуктах.

Таблица 1 – Обзор аналогов

Критерии	**Absolute Pitch**	**Perfect Piano**	**Skoove Piano**
Поддержка Android 14 и выше	+	+	+
Сбор статистики пользователей	+	–	–
Наличие структурированных курсов	+	–	+
Возможность играть на виртуальном пианино без упражнений	–	+	–
Наличие теории музыки в виде текста и картинок	+	–	–
Наличие регистрации пользователя	–	+	+
Наличие тестов для проверки знаний теории пользователем	–	–	–

Вывод по первой главе

В результате сравнения аналогов были выявлены главные функции и особенности разработанных ранее приложений сторонними разработчиками, что позволило вывести список тех функций, которые должны присутствовать в разрабатываемом мобильном приложении PianoMentor. Эти функции приведены в таблице 1. Далее на основе этих функций будут сформированы конкретные функциональные и нефункциональные требования и критерии, которые будут определять процесс разработки.

2. ПРОЕКТИРОВАНИЕ СИСТЕМЫ

2.1. Анализ требований к программной системе

На основании информации, полученной в результате изучения предметной области и обзора аналогичных решений, были сформированы функциональные и нефункциональные требования к мобильному приложению PianoMentor.

Функциональные требования к системе

Описание функциональных требований к системе разбито на отдельное описание оных к серверному приложению и к мобильному приложению.

Для серверного приложения был составлен список таких функциональных требований:

1) наличие возможности регистрации и аутентификации пользователей;
2) предоставление механизмов по управлению пользовательскими токенами доступа к серверу со стороны администраторов сервиса;
3) наличие функции отправки файлов клиенту;
4) наличие функции получения файлов от клиента с проверкой на возможный не корректный тип файла;
5) наличие функции предоставления теоретической лекции;
6) наличие функции предоставления тестов;
7) наличие функции предоставления практических занятий;
8) наличие механизма для получения текущей статистики пользователя;
9) наличие механизма для сохранения прогресса обучения пользователя мобильного приложения;
10) наличие механизма по редактированию курсов и элементов курсов.

Для мобильного приложения был выработан следующий список с функциональными требованиями:

1) наличие функции регистрации и аутентификации пользователя;

2) наличие отображения прогресса обучения пользователя;

3) отображение профиля пользователя и возможности управления учетной записью пользователя;

4) наличие возможности чтения лекций;

5) наличие возможности прохождения тестов на проверку теоретических знаний;

6) наличие возможности выполнения практических упражнений;

7) экран практических занятий должен содержать в себе клавиатуру виртуального пианино, воспроизводящее звуки пианино;

8) наличие возможности сыграть на виртуальном пианино помимо каких-либо заданий;

9) наличие экранов для отображения всех лекций из всех курсов, всех тестов из всех курсов, всех практических упражнений из всех курсов.

Нефункциональные требования к системе

При описании нефункциональных требований также используется разбиение по частям системы: отдельно для серверного приложения, отдельно для мобильного приложения.

Для серверного приложения можно выделить следующие нефункциональные требования:

1) API серверного приложения должно быть быстродействующим с низкой степенью задержки;

2) система должна поддерживать одновременную работу нескольких тысяч пользователей с низкой степенью ухудшения производительности;

3) для сообщения сервера и клиента должен использоваться архитектурный стиль REST [15];

4) данные, передаваемые между сервером и клиентом, должны реализовывать протокол HTTPS [16] для сохранения конфиденциальности данных;

5) API должен иметь простые, ясные, самоописывающие URL [17] для доступа к эндпоинтам;

6) API должен иметь адреса URL, закодированные в кодировке UTF-8 [17].

Для мобильного приложения можно выделить следующие нефункциональные требования:

1) интерфейс приложения должен быть на русском языке;

2) приложение должно иметь удобный интерфейс;

3) цветовая гамма, использующаяся в приложении, должна состоять из палитры спокойных цветов;

4) шрифты всех текстов приложения должны быть читаемыми;

5) клавиатура виртуального пианино должна иметь такие размеры, чтобы пальцы человека при нажатии на нее не задевали соседние клавиши.

2.2. Описание вариантов использования

Для моделирования ранее выделенных функциональных требований к системе с помощью языка объектного моделирования UML были созданы диаграммы вариантов использования как для мобильного приложения, так и для серверного приложения.

Серверное приложение

Для начала стоит рассмотреть диаграмму вариантов использования серверного приложения. В моделируемой системе присутствует один актер – клиент, который будет обращаться к Web API (серверному приложению). Под словом «клиент» в данном контексте понимается не человек, а некоторая сторонняя программа, например, мобильное приложение, которая с помощью HTTPS запросов будет получать нужные ей данные от Web API. Диаграмма вариантов использования серверного приложения представлена на рисунке 7.

Рисунок 7 – Диаграмма вариантов использования серверного приложения для разрабатываемой системы

Далее приведены действия, которые может совершать клиент.

1. Зарегистрировать пользователя – создание учетной записи нового пользователя.

2. Аутентифицировать пользователя – вход в систему под учетной записью пользователя.

3. Восстановить access и refresh токены – отзыв старых access & refresh tokens, последующее создание новых с восстановлением доступа к сессии пользователя.

4. Отозвать refresh токен – отзыв refresh токен(-ы) без восстановления доступа к сессии(-ям) пользователя(-ей).

5. Получить данные о курсах – получение данных о доступных курсах.

6. Получить данные об элементе курса – позволяет получить данные, описывающие элемент курса.

7. Получить файл с лекцией – позволяет получить PDF-документ с лекцией по той или иной теме.

8. Получить картинку, привязанную к вопросу теста – возможность получить растровое изображение, которое привязано к вопросу теста.

9. Получить данные статистики пользователя – получение данных о статистике прохождения обучения пользователем.

10. Управлять файловым менеджером – позволяет загружать и скачивать пользовательские файлы, создавать одноразовую ссылку для скачивания файлов, использовать эту ссылку может любой пользователь, получивший эту одноразовую ссылку.

11. Управлять курсами – возможность добавлять новые и редактировать старые курсы.

12. Добавить файл к элементу курса – возможность прикрепить файл к элементу курса.

Мобильное приложение

Далее стоит рассмотреть диаграмму вариантов использования мобильного приложения. В ней присутствует два актера: аутентифицированный пользователь и неаутентифицированный. Построенная диаграмма вариантов использования представлена на рисунке 8.

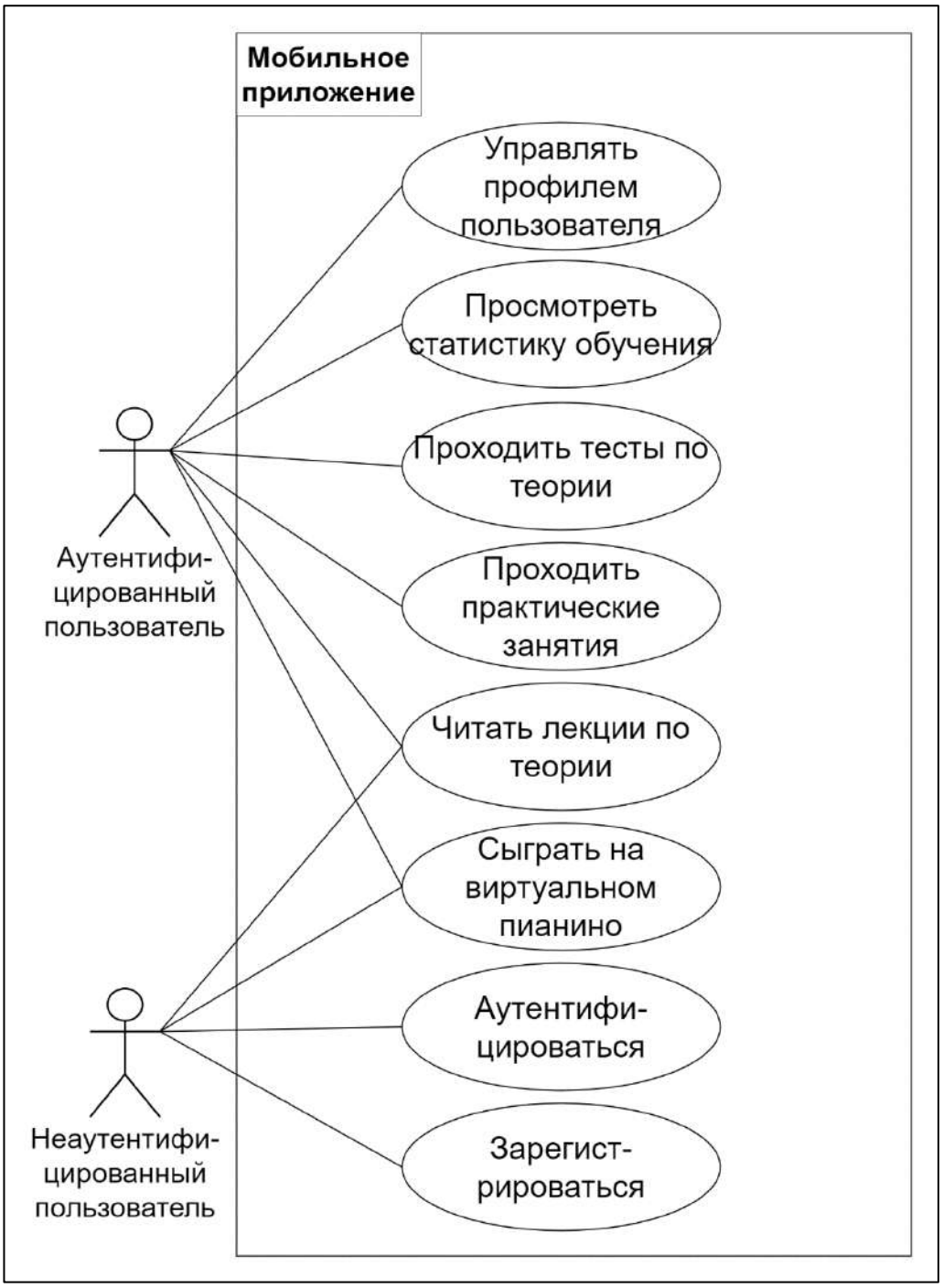

Рисунок 8 – Диаграмма вариантов использования мобильного приложения PianoMentor

Ниже представлены действия, которые может совершать аутентифицированный пользователь.

1. Управлять профилем пользователя подразумевает под собой просмотр профиля, загрузку фотографии пользователя в систему и выход из учетной записи.

2. Просмотреть статистику обучения – возможность просматривать как круговые, так и линейный индикаторы выполнения тех или иных заданий.

3. Проходить тесты по теории – возможность пройти тестирование из нескольких вопросов для проверки знаний пользователя после прочтения лекции по теории.

4. Проходить практические занятия – возможность выполнить задание, которое будет давать первый опыт в развитии слуха у пользователя приложения, а также давать первое понимание основных принципов работы базовых музыкальных единиц.

5. Читать лекции по теории – возможность открыть ту или иную лекцию по определенному теоретическому материалу для прочтения пользователем.

6. Сыграть на виртуальном пианино – возможность использовать виртуальное пианино без прохождения каких-нибудь занятий.

7. Аутентифицироваться – возможность войти в учетную запись пользователя.

8. Зарегистрироваться – возможность создать новую учетную запись пользователя.

Согласно диаграмме вариантов использования для мобильного приложения неаутентифицированный пользователь не может иметь доступа ко всему функционалу, так как в случае сохранения прогресса обучения локально на устройстве данные будут находиться под угрозой полного удаления случае удаления кэша приложения PianoMentor, а прогресс аутентифицированного пользователя автоматически будет отправляться на сервер, в следствии чего аутентифицированный пользователь имеет полный доступ к приложению.

2.3. Архитектура системы

Использование архитектурных шаблонов приложений обусловлено тем, что каждое приложение с наращиванием функционала будет наращивать и кодовую базу, без разделения кодовой базы на логические части, а равно без использования общепринятых подходов в проектировании программ, разработчики обязательно столкнутся с проблемой сложности поддержания работоспособности кода. Поэтому при разработке как мобильного приложения PianoMentor, так и серверного приложения, были учтены основные наработки в плане архитектур IT-продуктов.

Первоочередно стоит описать общее правило построения архитектуры, реализуемое в обоих частях проектируемой системы, таковым является паттерн многослойной архитектуры [1]. Он предполагает разделение всего кода приложения на уровни по функциональному признаку. Таковых уровней может быть ровно столько, сколько требуется. Часто говорится о трех основных: слой представления (Presentation layer или UI layer), слой бизнес-логики (Domain layer) и слой данных (Data layer). На рисунке 9 можно увидеть схему, описывающую многоуровневую архитектуру.

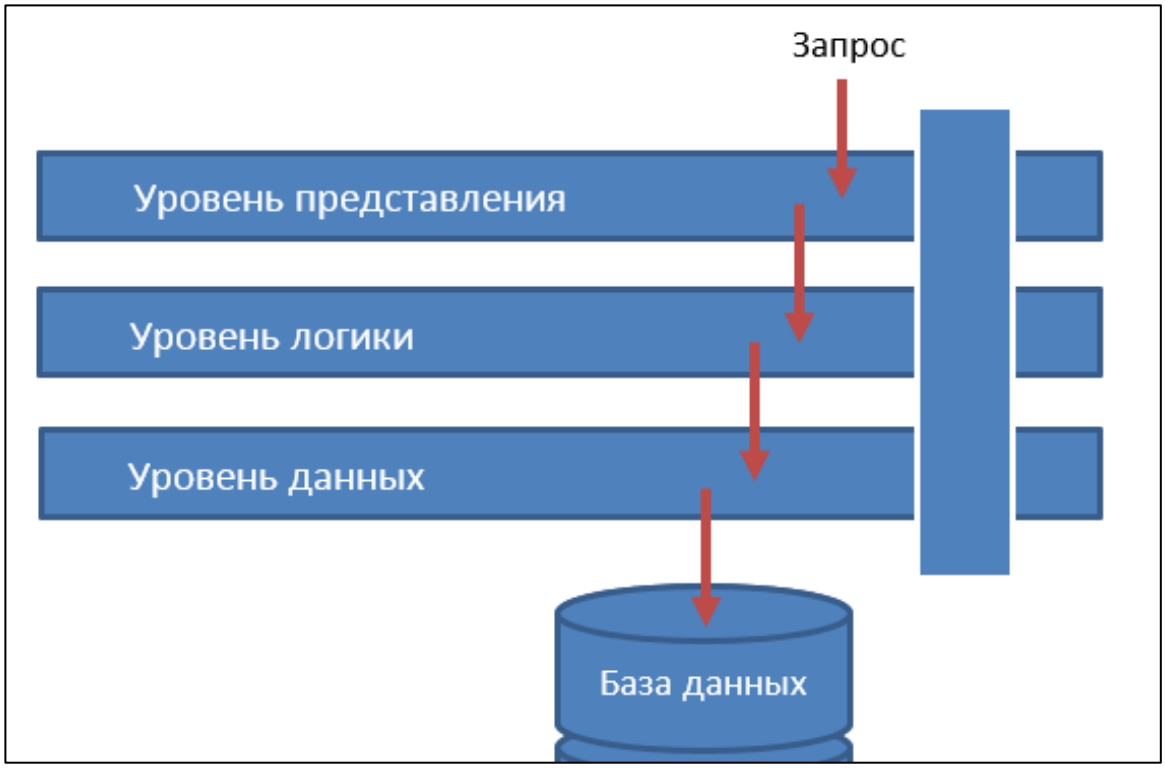

Рисунок 9 – Схема многоуровневой архитектуры

Слой представления занимается исключительно отображением данных для пользователя, обработкой пользовательских событий, а также допускается использование некоторых отдельных способов преобразования данных.

Слой бизнес-логики предназначен для содержания в себе модулей, реализующих расчеты, анализ данных, преобразование сущностей, полученных из слоя данных, и прочее. Этот слой также может содержать интеракторы [2] для обеспечения связи между слоем данных и слоем представления в том случае, если преобразования данных при передаче из слоя в слой не требуется.

Слой данных служит для обеспечения доступа к необходимым данным. Сюда включают классы и интерфейсы, ответственность которых состоит в организации доступа к базе данных, к внешним ресурсам в памяти устройства, к удаленному серверу с помощью сети Интернет и так далее.

В рамках мобильного приложения будет достаточно этих трех слоев, в рамках же серверного приложения будет использован дополнительный, четвертый слой – слой контракта (Contract layer). Его необходимость обуславливается тем, что требуется собрать в одном слое объекты трансфера данных (Data Transfer Object), предназначенные для хранения подготовленных данных, полученных из Data layer, а также содержать в себе классы контрактов для передачи данных между запрашивающим объектом и обрабатывающим объектом по средствам библиотеки MediatR [3]. Касаемо данной библиотеки ведутся споры о том, какой же именно паттерн проектирования она реализует. Одни говорят, что это реализация одноименного паттерна Mediator [4], другие же убеждены, что это реализация лишь паттерна Диспетчер команд [5], что создает предпосылку к Message-Oriented архитектуре [5, 6], тем самым обосновывая опасность использования рассматриваемой библиотеки при разработке приложений. Суть данного паттерна заключается в разрыве зависимостей между объектами, находящимися на уровне представления, и объектами, находящимися на уровне бизнес-логики [4].

В силу того, что разрабатываемое серверное приложение вряд ли сможет вырасти в столь большой проект, для которого будут по-настоящему существенны минусы, несущие с собой библиотекой MediatR, было принято решение прибегнуть к ее использованию.

В процессе разработки мобильного приложения будет использован паттерн проектирования под названием Model-View-ViewModel (MVVM) [7, 8] для более детального разграничения ответственностей в слое представления и для пояснения способа соединения Domain и UI слоев. Коротко описать паттерн MVVM можно следующим образом:

1. Модель (Model) является компонентом, отвечающим за управление данными приложения и хранение бизнес-логики;
2. Представление (View) – часть UI-слоя, отвечающая за отображение пользовательского интерфейса и взаимодействие с пользователем;
3. ViewModel представляет собой прослойку между моделью и представлением, она содержит логику, управляющую отображаемыми данными в представлении, хранит состояние представления, к которому привязан тот или иной класс, реализующий ViewModel, содержит в себе методы, с помощью которых инициируется вызов методов из слоя бизнес-логики [9].

Схематично MVVM представлен на рисунке 10.

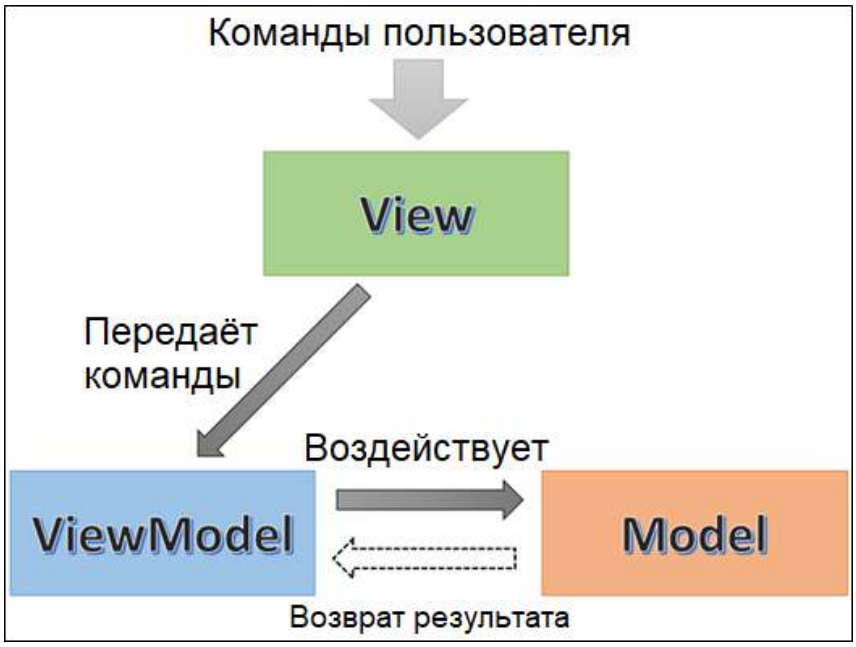

Рисунок 10 – Схема паттерна MVVM

На схеме выше стрелочками показаны потоки управления, начало которые берут от действий пользователя. Действия пользователя не всегда могут достигать модели, паттерн проектирования подразумевает возможность внедрения валидации полученных данных от пользователя на всех уровнях, тем самым заранее освобождая последующие слои от излишней работы [10].

Для серверного приложения был выбран архитектурный паттерн Model-View-Controller (MVC) [11], который является некоторым стандартом для ASP.NET Core приложений и поддерживается изначально этой платформой. MVC и MVVM являются несколько похожими паттернами проектирования. Model и View схожи по своему назначению в обоих рассматриваемых случаях, но ключевым различием MVVM и MVC кроется во взаимодействии представления и модели, в таблице 2 можно увидеть сравнение модулей ViewModel и Controller из двух рассматриваемых шаблонов [12, 13].

Таблица 2 – Сравнение ViewModel и Controller.

Критерии	**ViewModel**	**Controller**
Является посредником между Model и View	+	–
Обновляет состояние View	+	–
Содержит в себе методы, с помощью которых инициируется вызов методов из Domain слоя	+	+
Может содержать в себе код по валидации данных или прочей мелкой работы с данными	+	+
Принимает данные, вводимые пользователем	–	+
Хранит в себе текущее состояние View	+	–
Может обрабатывать информацию, полученную из Domain слоя	+	–
Позволяет использовать двустороннюю привязку данных	+	–

В рамках Web API использование шаблона MVC имеет одну дополнительную особенность. Компонент View отвечает за отображение данных, которые в Web API сериализуются в специальные текстовые форматы обмена данными (JSON, XML, RDF и подобные) [14]. За это отвечают зачастую заранее подготовленные библиотеки. На рисунке 11 представлена схема паттерна MVC.

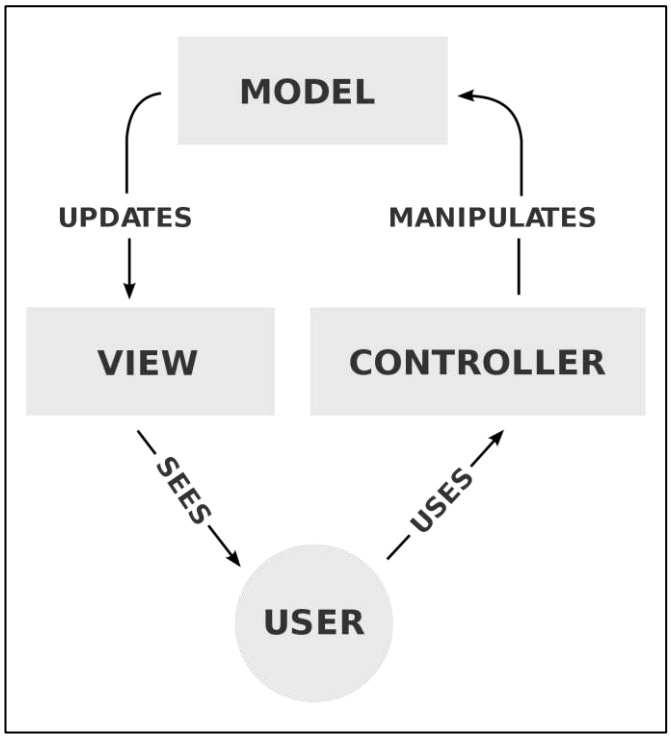

Рисунок 11 – Схема шаблона MVC

2.4. Компоненты системы

В силу того, что разрабатывается полноценная функционирующая система, то стоит изначально рассмотреть общую диаграмму всех компонентов (рисунок 12). На ней представлены три компоненты: база данных, серверное приложение и мобильное приложение. В рамках диаграммы используется нотация ball-and-socket (предусмотренный интерфейс и требуемый интерфейс) для отображения отношений между компонентами, а именно:

1) база данных предоставляет интерфейс, который требуется серверному приложению;

2) серверное приложение предоставляет интерфейс, который требуется мобильному приложению.

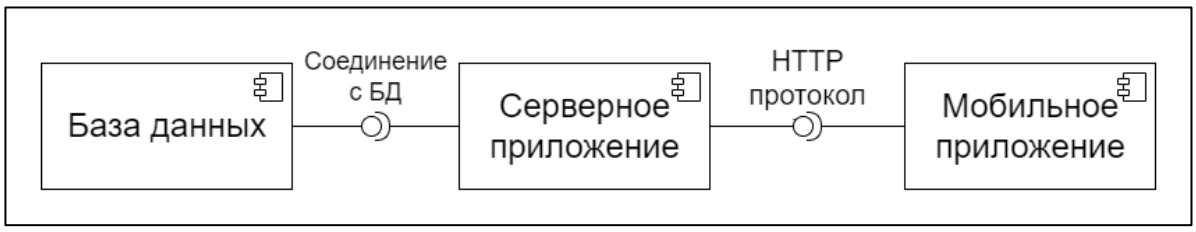

Рисунок 12 – Диаграмма компонентов разрабатываемой системы

Компоненты мобильного приложения

На рисунке 13 можно увидеть диаграмму компонентов мобильного приложения.

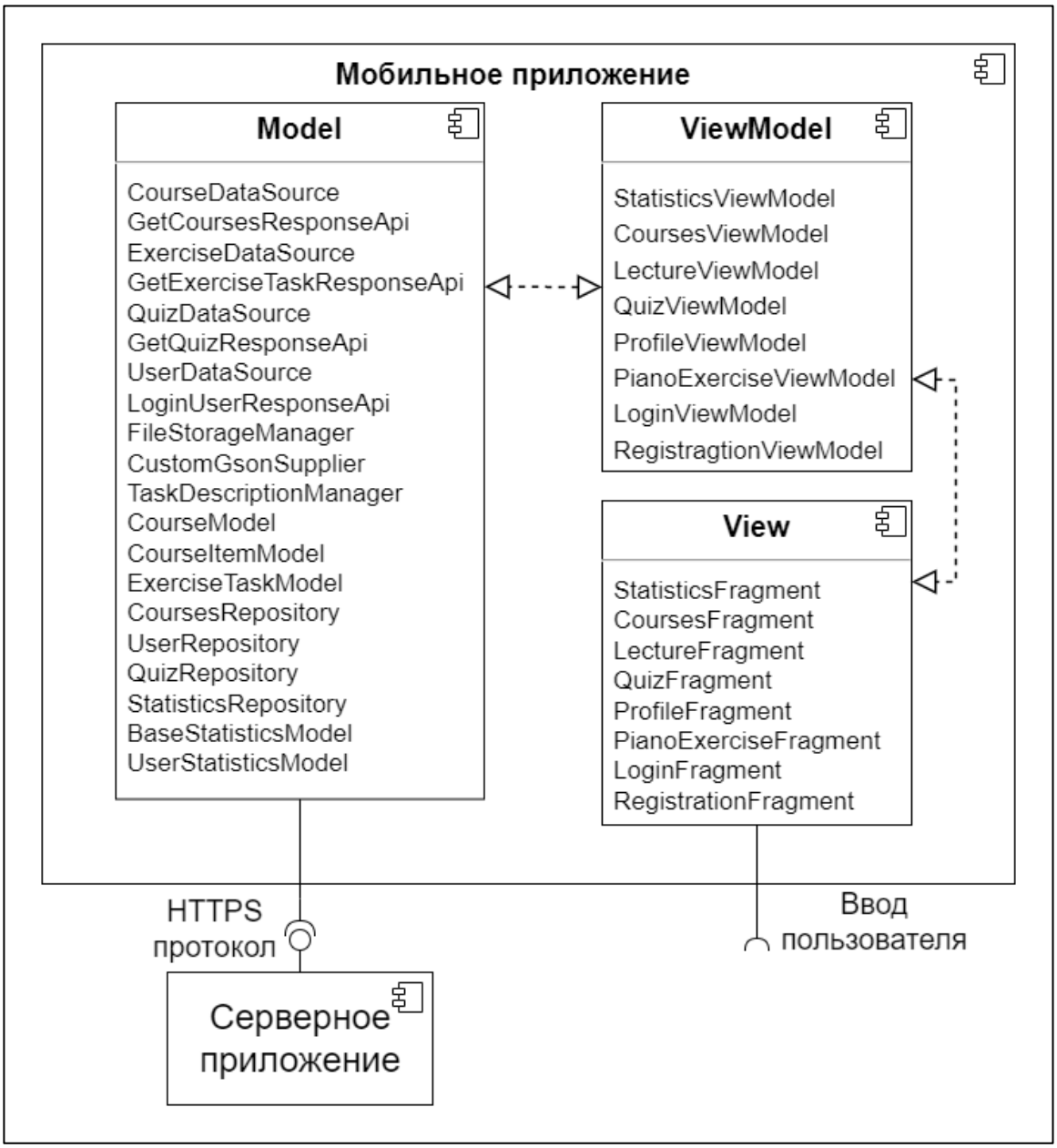

Рисунок 13 – Диаграмма компонентов мобильного приложения

На диаграмме компонентов мобильного приложения можно увидеть более подробное описание того, какие элементы в себе содержат компоненты Model, View и ViewModel, а также связи между компонентами. Далее представлено более подробное описание каждого элемента.

Компонент Model содержит в себе элементы, приведенные ниже.

1. CourseDataSource – класс, отвечающий за получение данных о курсах из удаленного источника для мобильного приложения.
2. GetCoursesResponseApi – класс, представляющий ответ API на запрос получения списка курсов.
3. ExerciseDataSource – класс, отвечающий за получение данных об упражнениях из удаленного источника для мобильного приложения.
4. GetExerciseTaskResponseApi – класс, представляющий ответ API на запрос получения задания упражнения.
5. QuizDataSource – класс, отвечающий за получение данных о тестах на знание лекций по теории из удаленного источника для мобильного приложения.
6. GetQuizResponseApi – класс, представляющий ответ API на запрос получения теста, содержит вопросы и возможные ответы для пользователя.
7. UserDataSource – класс, отвечающий за получение и управление данными пользователя из удаленного источника для мобильного приложения.
8. LoginUserResponseApi – класс, представляющий ответ API на запрос аутентификации пользователя, содержит информацию о статусе входа и токены при успешной аутентификации.
9. FileStorageManager – класс, отвечающий за локальное хранение файлов на мобильном устройстве, обеспечивая сохранение и доступ к данным.
10. CustomGsonSupplier – класс, предоставляющий настроенный экземпляр объекта типа Gson (предоставляется одноименной библиотекой) для сериализации и десериализации JSON-данных в приложении.
11. TaskDescriptionManager – класс, предоставляющий хранящиеся локально текстовые описания практических заданий.
12. CourseModel – класс, предоставляющий модель курса.

13. CourseItemModel – класс, предоставляющий модель элемента курса.

14. ExerciseTaskModel – класс, предоставляющий модель задания упражнения.

15. CoursesRepository – класс, отвечающий за управление и доступ к данным о курсах.

16. UserRepository – класс, отвечающий за управление и доступ к данным пользователя.

17. QuizRepository – класс, отвечающий за управление и доступ к данным о тестах на знание теоретических тестов.

18. StatisticsRepository – класс, отвечающий за управление и доступ к статистическим данным пользователя.

19. BaseStatisticsModel – класс, предоставляющий базовую модель прогресса прохождения обучения пользователем.

20. UserStatisticsModel – класс, предоставляющий модель, содержащую всю пользовательскую статистику прохождения обучения.

Компонент ViewModel содержит в себе элементы, приведенные ниже.

1. StatisticsViewModel – класс, управляющий данными, связанными со статистикой пользователя, включая загрузку и обновление данных статистики.

2. CoursesViewModel – класс, управляющий данными, связанными с курсами, включая загрузку списка курсов и обработку действий пользователя.

3. LectureViewModel – класс, управляющий данными, связанными с лекциями, включая загрузку pdf-файлов лекций.

4. QuizViewModel – класс, управляющий данными, связанными с тестами, включая загрузку вопросов и обработку ответов пользователя.

5. ProfileViewModel – класс, управляющий данными, связанными с профилем пользователя, включая загрузку и обновление данных профиля.

6. PianoExerciseViewModel – класс, управляющий данными, связанными с упражнениями по игре на пианино, включая загрузку и обновление данных упражнений.

7. LoginViewModel – класс, управляющий данными, связанными с процессом логина пользователя, включая обработку ввода пользователя и запросы к API.

8. RegistragtionViewModel – класс, управляющий данными, связанными с процессом регистрации пользователя, включая обработку ввода пользователя и запросы к API.

Компонент View содержит в себе элементы, приведенные ниже.

1. StatisticsFragment – фрагмент, отображающий статистику пользователя, включая прогресс в курсах и другие метрики.

2. CoursesFragment – фрагмент, отображающий список доступных курсов и предоставляющий доступ к деталям курса.

3. LectureFragment – фрагмент, отображающий содержимое конкретной лекции.

4. QuizFragment – фрагмент, отображающий вопросы теста и позволяющий пользователю отвечать на них.

5. ProfileFragment – фрагмент, отображающий информацию профиля пользователя и предоставляющий возможность редактировать эти данные.

6. PianoExerciseFragment – фрагмент, отображающий упражнения по игре на пианино, включая ноты и инструкции.

7. LoginFragment – фрагмент, отображающий форму для ввода логина пользователя и обрабатывающий процесс аутентификации.

8. RegistrationFragment – фрагмент, отображающий форму для регистрации нового пользователя и обрабатывающий процесс создания нового аккаунта.

Компоненты серверного приложения

Далее перейдем к рассмотрению диаграммы компонентов серверного приложения, которую можно увидеть на рисунке 14.

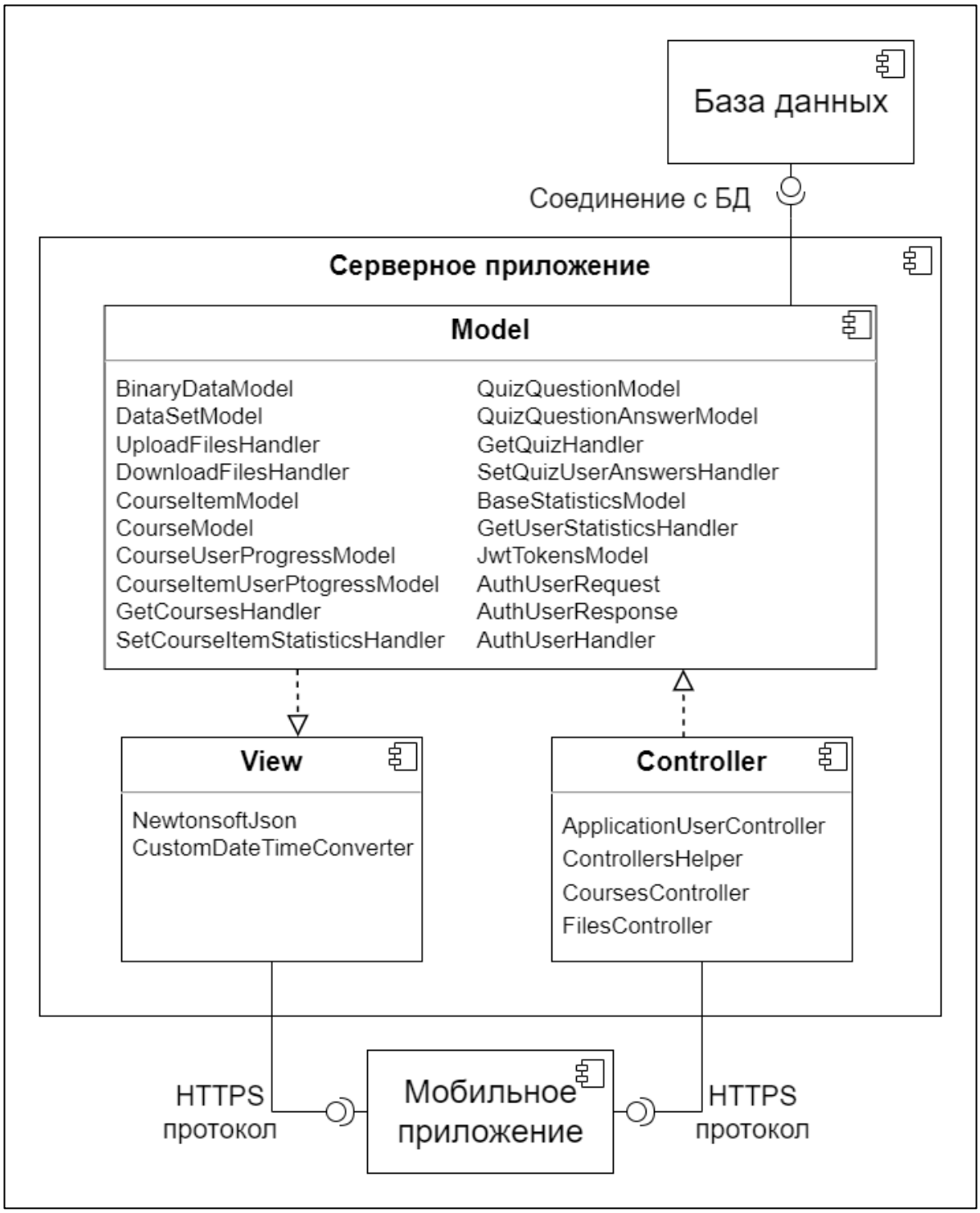

Рисунок 14 – Диаграмма компонентов серверного приложения

Диаграмма компонентов серверного приложения дает возможность схематически изобразить состав рассматриваемой системы с указанием связей между каждым элементом. Далее описан каждый элемент каждого из трех компонентов серверного приложения.

Компонент Model содержит следующие элементы.

1. BinaryDataModel – класс, представляющий модель метаданных для идентификации некоторого двоичного файла.
2. DataSetModel – класс, представляющий модель метаданных для идентификации набора двоичных данных.
3. UploadFilesHandler – обрабатывает загрузку файлов на сервер, сохраняя их и возвращая информацию о загруженных файлах.
4. DownloadFilesHandler – обрабатывает скачивание файлов с сервера, предоставляя бинарные данные по запросу.
5. CourseItemModel – класс, представляющий элемент курса, который может быть лекцией, упражнением или тестом.
6. CourseModel – класс, представляющий модель курса, содержащий данные о курсе, такие как название, описание и прогресс прохождения этого курса.
7. CourseUserProgressModel – класс, представляющий прогресс пользователя в курсе. Включает свойства для идентификатора пользователя, идентификатора курса и текущего статуса прогресса.
8. CourseItemUserProgressModel – класс, представляющий прогресс пользователя по конкретному элементу курса. Включает свойства для идентификатора пользователя, идентификатора элемента курса и текущего статуса прогресса.
9. GetCoursesHandler – класс, обрабатывающий запросы на получение списка доступных курсов для пользователя, возвращая соответствующую информацию.

10. SetCourseItemStatisticsHandler – класс, обрабатывающий запросы на установку и обновление статистики по элементам курса для пользователя, обновляя соответствующие данные.

11. QuizQuestionModel – класс, представляющий вопрос теста, содержащий текст вопроса, варианты ответов и картинку, если она требуется

12. QuizQuestionAnswerModel – класс, представляющий ответ на вопрос теста.

13. GetQuizHandler – класс, обрабатывающий запросы на получение викторины, возвращая вопросы и возможные ответы для пользователя.

14. SetQuizUserAnswersHandler – класс, обрабатывающий запросы на установку и сохранение ответов пользователя на вопросы викторины, обновляя данные в системе.

15. BaseStatisticsModel – класс для статистики пользователя.

16. GetUserStatisticsHandler – класс, обрабатывающий запросы на получение статистики пользователя, возвращая информацию о его успехах и прогрессе.

17. JwtTokensModel – класс, представляющий модель для работы с JWT-токенами, используемыми для аутентификации и авторизации пользователей.

18. AuthUserRequest – класс, представляющий модель запроса на аутентификацию пользователя, включая данные для входа, такие как имя пользователя и пароль.

19. AuthUserResponse – класс, представляющий модель ответа на запрос аутентификации, включая информацию о статусе входа и JWT-токен при успешной аутентификации.

20. AuthUserHandler – класс, обрабатывающий запросы на аутентификацию пользователей, проверяя данные для входа и выдавая JWT-токены для успешной аутентификации.

Компонент Controller содержит следующие элементы.

1. ApplicationUserController – контроллер, который управляет действиями, связанными с пользователями приложения.

2. ControllersHelper – вспомогательный класс, который предоставляет общие методы и утилиты для использования в различных контроллерах.

3. CoursesController – контроллер, который управляет действиями, связанными с курсами.

4. FilesController – контроллер, который управляет действиями, связанными с файлами.

Компонент View включает следующие элементы.

1. NewtonsoftJson - библиотека для работы с JSON в .NET. Используется для сериализации и десериализации объектов в JSON и из JSON.

2. CustomDateTimeConverter – класс, представляющий конвертер для работы с датами и временем.

2.5. Проектирование базы данных

Отдельное внимание стоит уделить проектированию базы данных. Требуется спроектировать отношения между таблицами так, чтобы БД удовлетворяла не менее третьей нормальной форме [26], способствовало эффективной обработке и удобной организации данных. В результате проектирования БД была создана 31 таблица, определены связи между таблицами и поля каждой таблицы.

Далее будут описаны таблицы и отношения между ними, используя схематично представление. В силу обширности базы данных, ее проект будет описываться по частям, таковые части будут разделены по логическому признаку.

На рисунке 15 представлен проект части базы данных, где изображены таблицы, хранящие информацию о пользователях, и связи между ними.

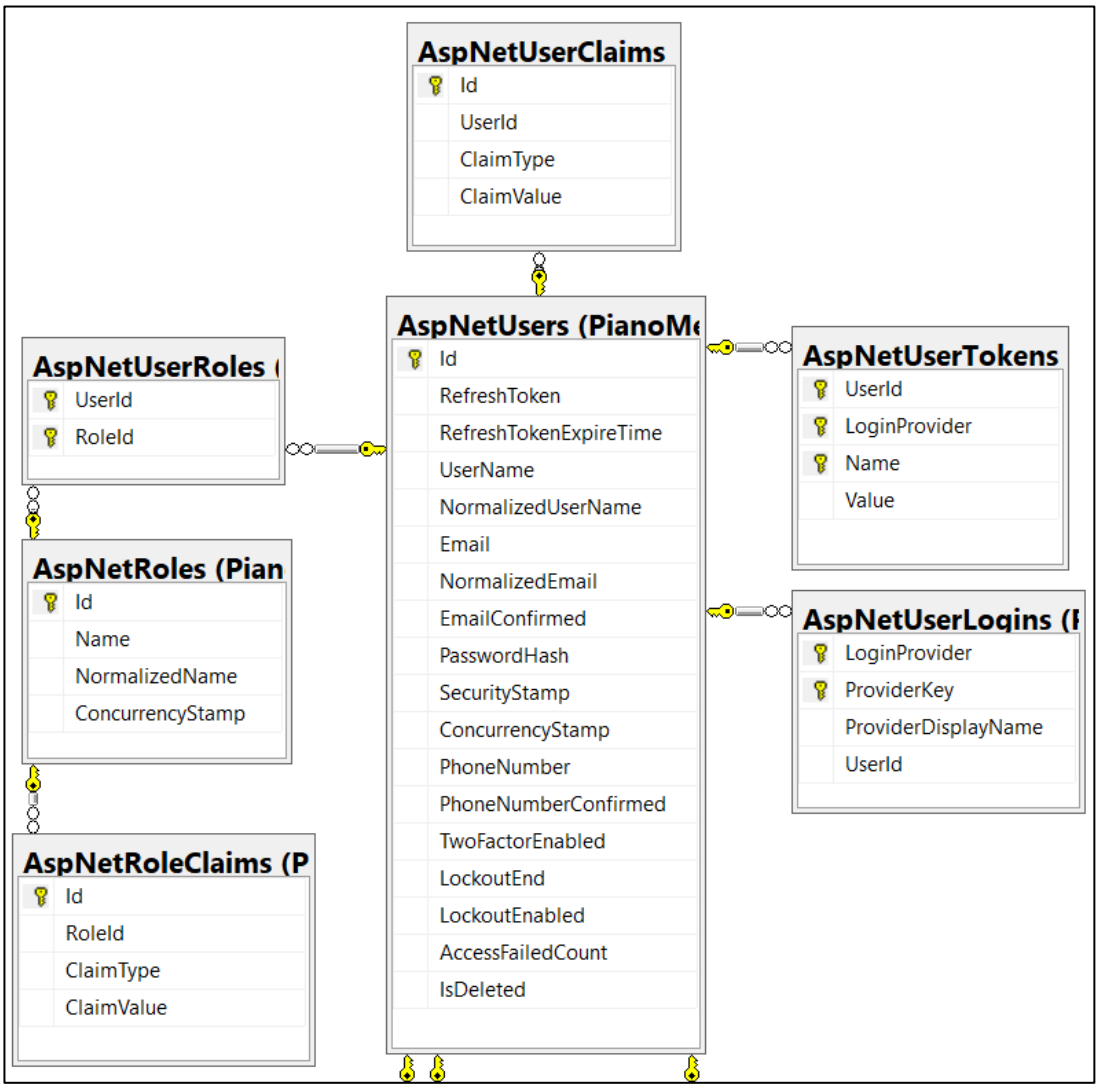

Рисунок 15 – Проект части базы данных (пользователи)

В таблице 3 можно видеть описание таблиц, отвечающих за хранение данных пользователей.

Таблица 3 – Описание назначений таблиц пользователей в базе данных

Название таблицы	Описание
AspNetUsers	Содержит основную информацию о пользователях
AspNetRoles	Содержит доступные пользовательские роли
AspNetRoleClaims	Содержит утверждения для ролей
AspNetUserRoles	Устанавливает связь между пользователями и их ролями
AspNetUserClaims	Содержит пользовательские утверждения (claims)
AspNetUserLogins	Содержит информацию о внешних учетных записях
AspNetUserTokens	Содержит токены аутентификации пользователей

На рисунке 16 представлен проект части базы данных, отвечающей за хранение данных об учебных курсах, а также указаны пометки связей с таблицами из иных частей проекта.

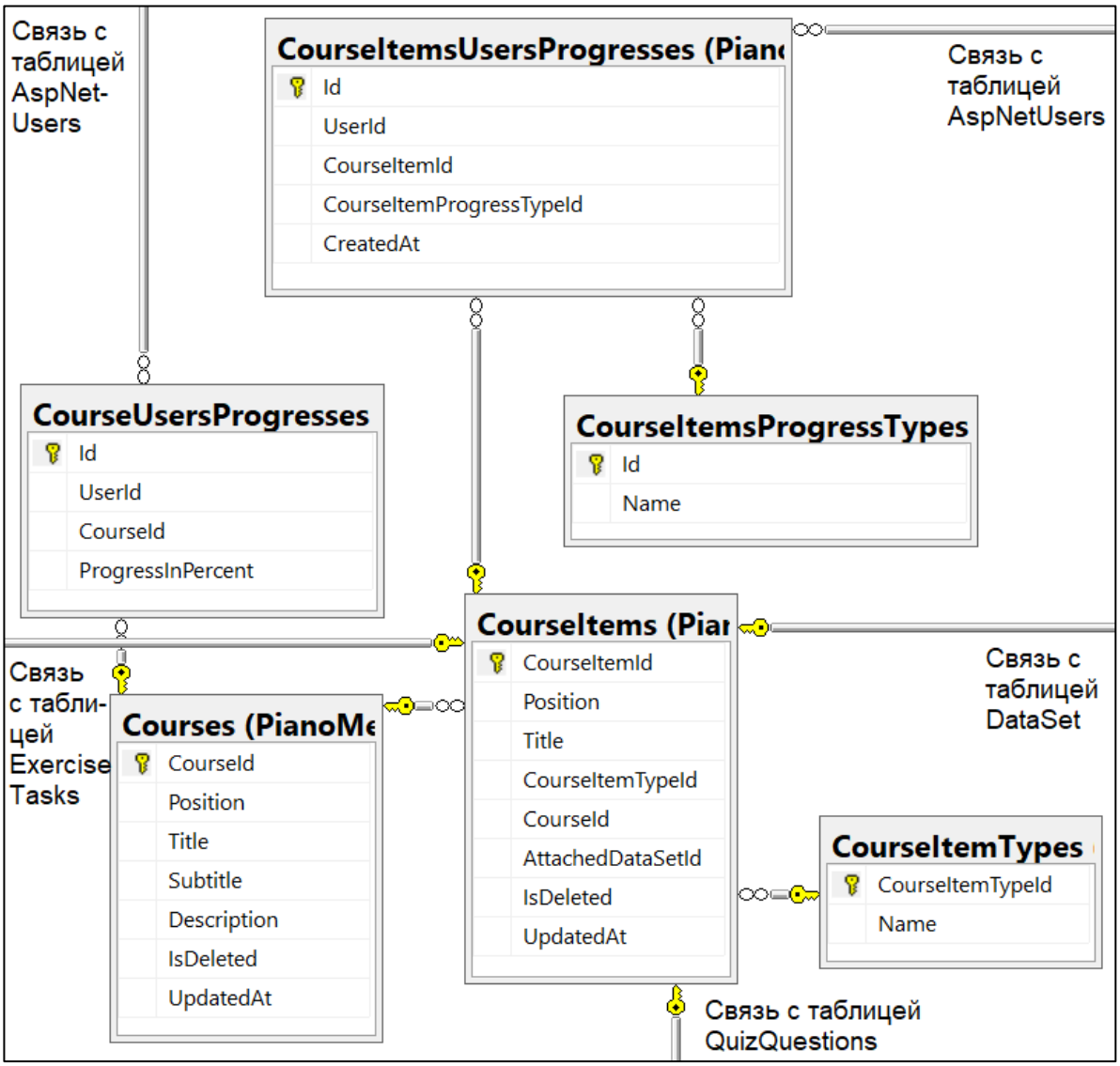

Рисунок 16 – Проект части базы данных (курсы)

В таблице 4 представлено описание таблиц, хранящих данные о курсах.

Таблица 4 – Описание назначений таблиц курсов в базе данных

Название таблицы	Описание
Courses	Содержит информацию о курсах
CourseItems	Содержит информацию об элементах курсов
CourseItemTypes	Содержит типы элементов курсов
CpurseUserProgresses	Содержит записи о прогрессе прохождения курсов
CourseItemsUsersProgresses	Содержит записи о прогрессе прохождения элементов курсов
CourseItemsProgressTypes	Содержит типы прогресса прохождения course item

На рисунке 17 представлен проект части базы данных, отвечающий за хранение информации о вопросах тестов.

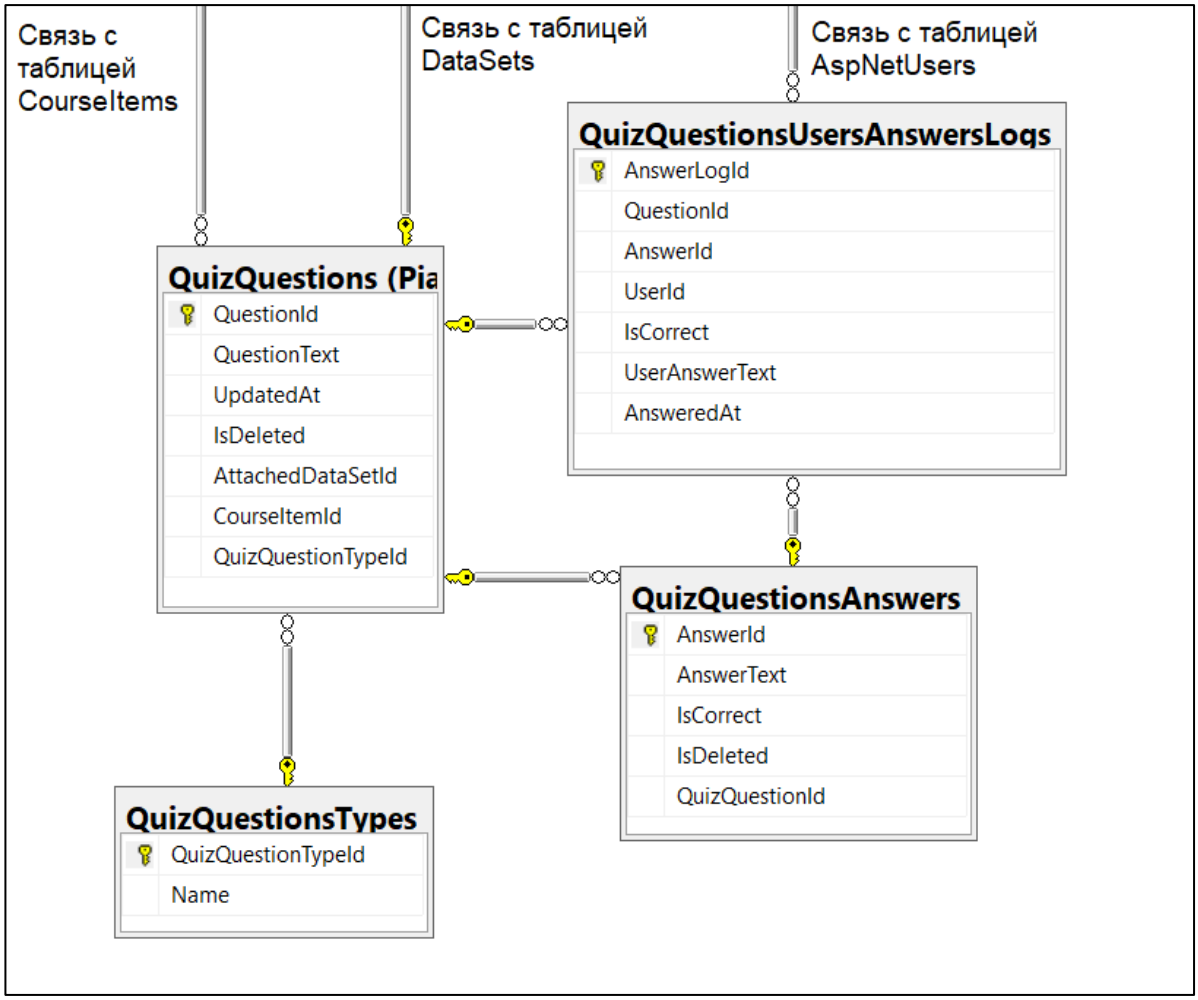

Рисунок 17 – Схема части базы данных (вопросы)

В таблице 5 представлено описание таблиц, предназначенных для хранения данных о вопросах, используемых в тестах.

Таблица 5 – Описание назначений таблиц вопросов в базе данных

Название таблицы	Описание
QuizQuestion	Содержит информацию о вопросах, которые используются в тестах на знание материалов лекций
QuizQuestionsAnswers	Содержит информацию о вариантах ответов на вопросы
QuizQuestionsTypes	Содержит типы вопросов
QuizQuestionsUsersAnswersLogs	Содержит все записи о тех ответах, которые выбрали пользователи при прохождении тех или иных тестов

На рисунке 18 представлен проект части базы данных для хранения текстов для элемента ViewPager в мобильном приложении и для хранения метаданных о пользовательских файлах.

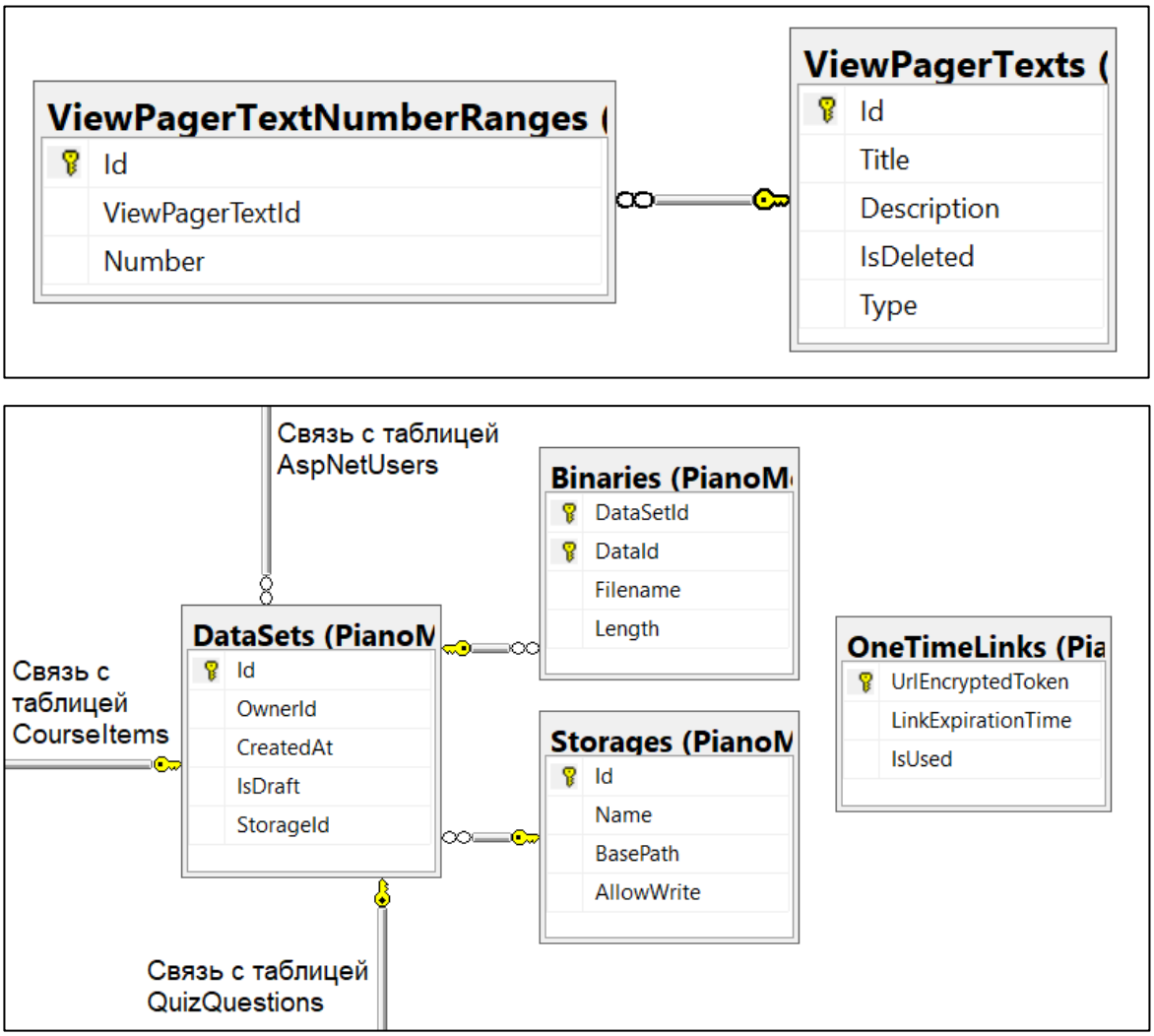

Рисунок 18 – Проект частей базы данных с текстами (сверху) и метаданными о пользовательских файлах (снизу)

В таблице 6 содержится описание таблиц, изображенных на рисунке 18.

Таблица 6 – Описание назначений таблиц для хранения небольших текстов и метаданных пользовательских файлов

Название таблицы	Описание
ViewPagerTexts	Содержит короткие текста
ViewPagerTextNumberRanges	Содержит промежутки данных для коротких текстов
DataSets	Содержит метаданные наборов бинарных файлов
Binaries	Содержит метаданные бинарных файлов
Storages	Содержит записи о возможных хранилищах данных
OneTimeLinks	Содержит записи об одноразовых ссылках на данные

На рисунке 19 представлен проект части базы данных для хранения данных о практических упражнениях.

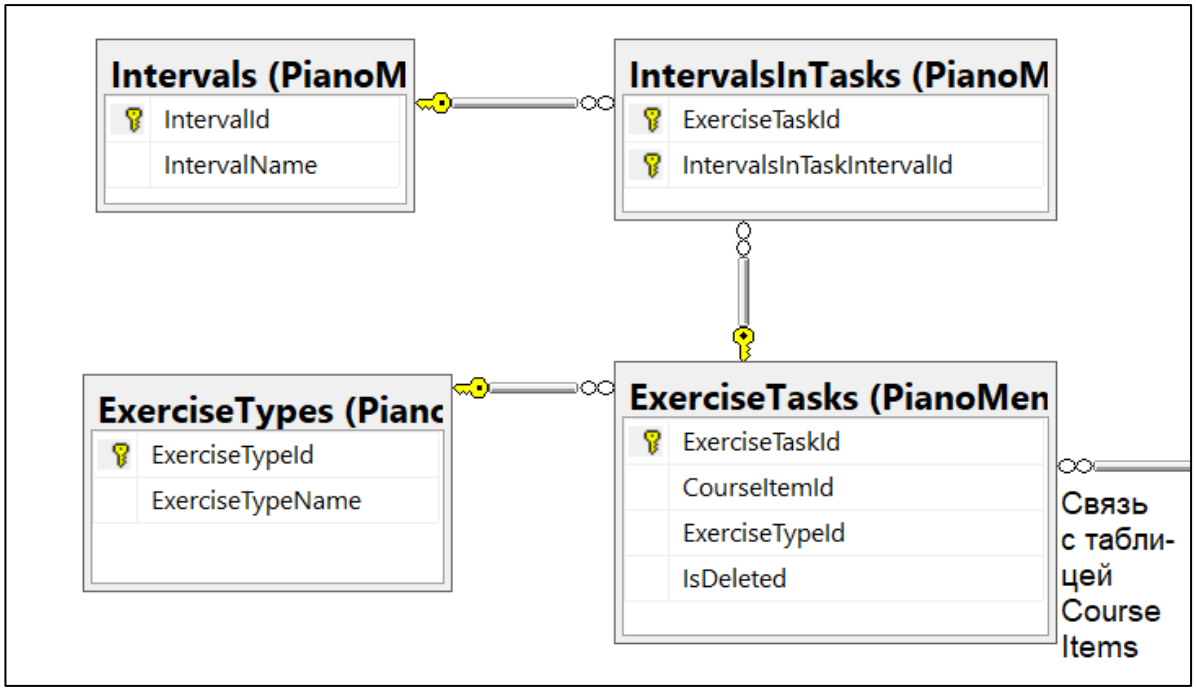

Рисунок 19 – Проект части базы данных с практическими упражнениями

В таблице 7 содержится описание таблиц, изображенных на рисунке 19.

Таблица 7 – Описание назначений таблиц для хранения данных о практических упражнениях

Название таблицы	Описание
ExerciseTasks	Является связующей таблицей, хранящая внешние ключи на сущности из таблиц IntervalsInTasks, ExerciseTypes, предоставляющая информацию нужную для определения задания на упражнение
ExerciseTypes	Содержит типы упражнений
Intervals	Содержит названия музыкальных интервалов
IntervalsInTasks	Таблица, реализующая связь многие-ко-многим между таблицами Intervals и ExerciseTasks

2.6. UI/UX-дизайн мобильного приложения

Для начала стоит оттолкнуться от определения принципов, по которым будет строиться UI/UX-дизайн приложения, таковыми являются:

1) преследование минимализма и простоты всего интерфейса;
2) использование интуитивной навигации по системе;

3) использование единообразных цветовой гаммы, шрифтов и геометрических форм элементов;

4) добавление простейших анимаций при переходе с экрана на экран.

В качестве акцентных цветов выбраны королевский пурпурный Крайола и классический белый цвет. Использована треугольная гармония цвета. Согласно этому, дополнительными цветами были выбраны сепия Крайола (близок к бледно-коричневому), арлекин (близок к нежно-зеленому). Сервисом для прототипирования была выбрана Figma.

Макет главной страницы и меню можно увидеть на рисунке 20.

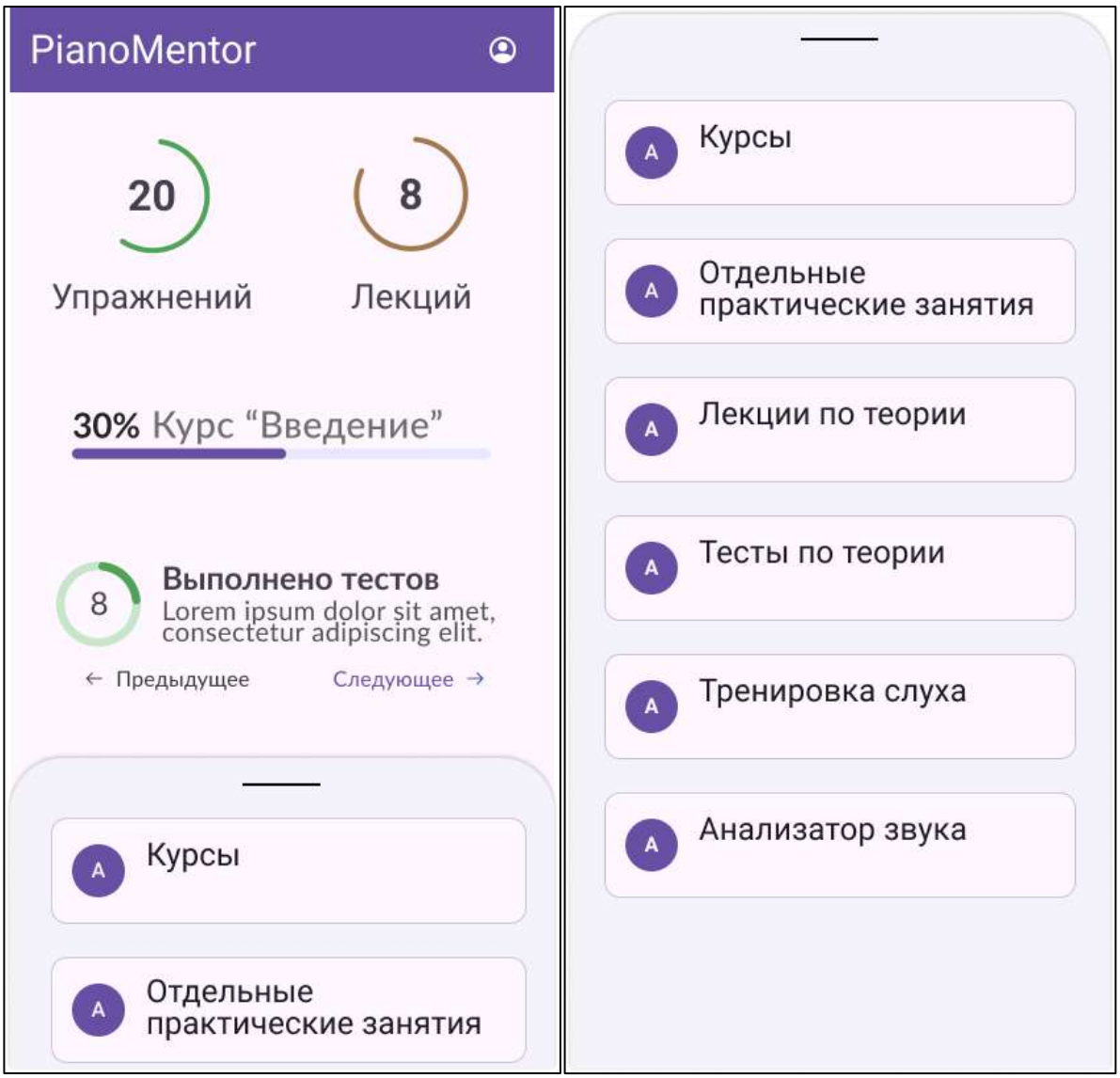

Рисунок 20 – Макет главной страницы (слева) и главного меню (справа)

На главном экране пользователь будет видеть статистику прохождения обучения в приложении. Снизу экрана представлен выдвижной элемент, являющийся главным меню.

Макет экрана со списком занятий можно увидеть на рисунке 21, представляющий из себя вертикальный список элементов с названиями занятия и различными иконками, символизирующие те или иные занятия. Также здесь имеет место цветовая индикация прогресса прохождения элемента, где бледно-фиолетовый фон иконки означает, что занятие не было начато, светло-синий – занятие начато, но не закончено, зеленый – занятие успешно пройдено, красный – занятие провалено.

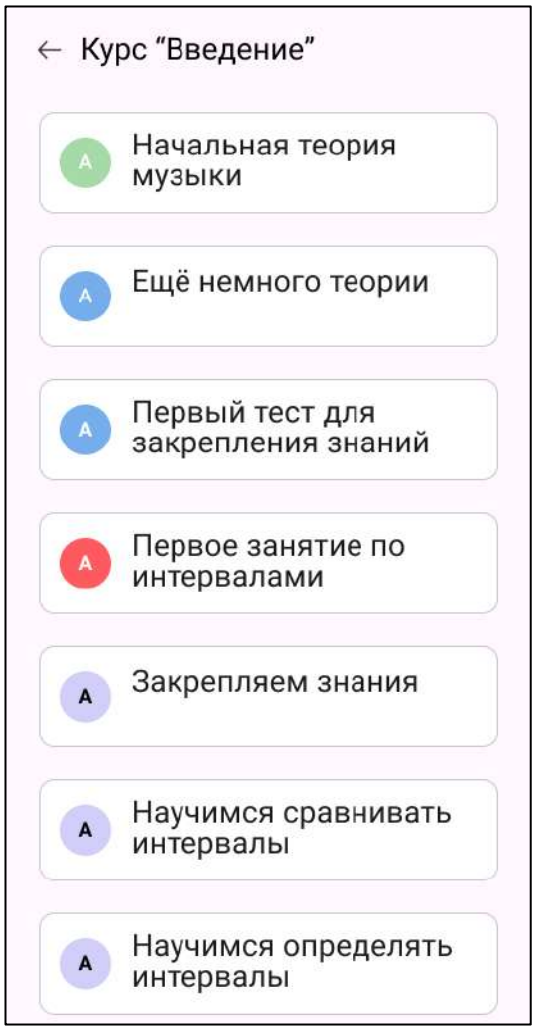

Рисунок 21 – Макет экрана выбора способа входа в учетную запись

Макеты экранов лекции и тестов, закрепляющих материал, пройденный на лекциях представлены на рисунке 22.

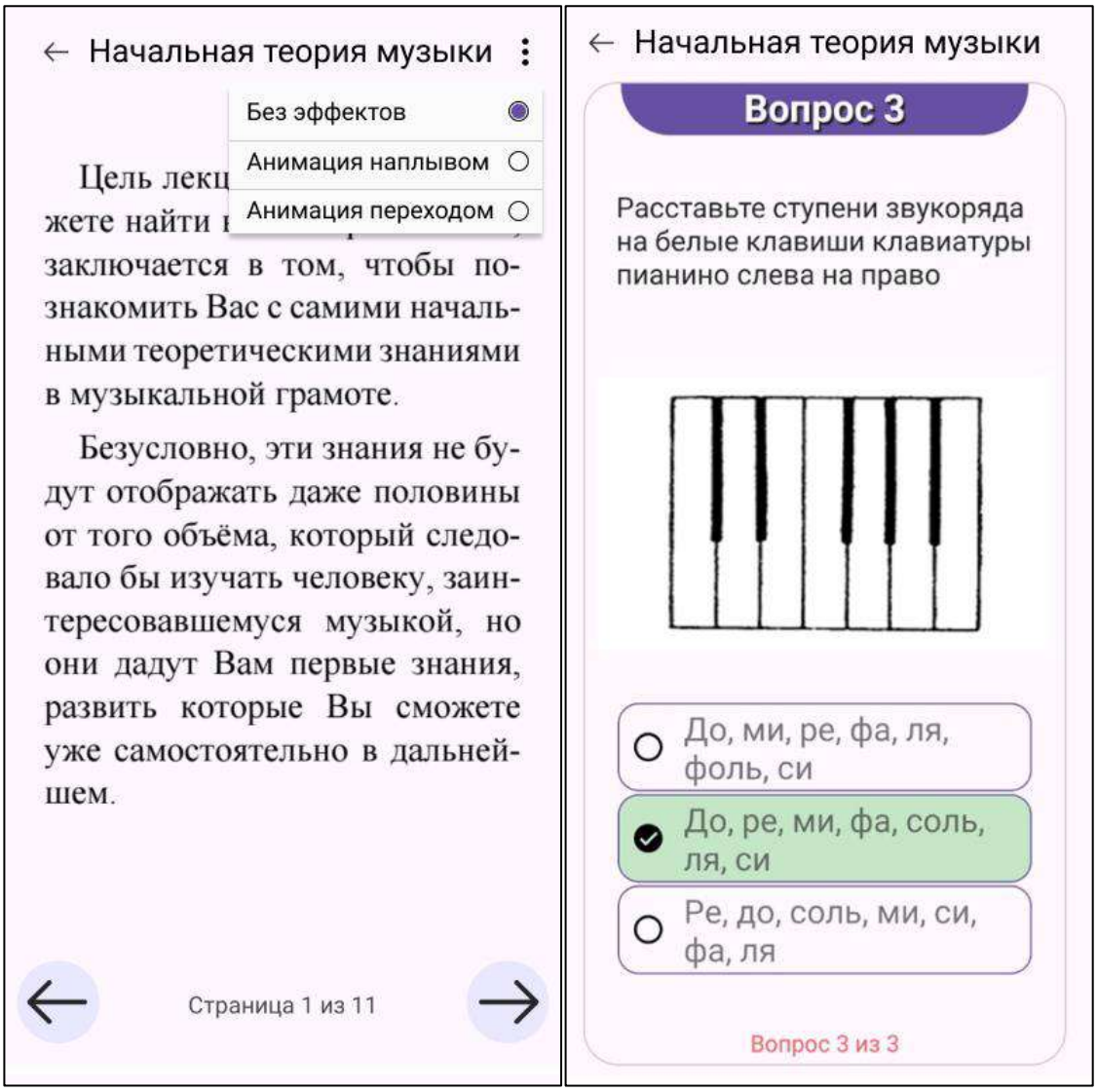

Рисунок 22 – Макет экрана лекции (слева) и тестов (справа)

Экран чтения лекций отображает подготовленный pdf-документ, выводя по одной странице на экране. Доступно два способа навигации по страницам: навигационные кнопки снизу и свайпы. Также пользователь сможет изменять способ анимации перелистывания страниц в контекстном меню. Следующим пунктом в курсе будет тест по прочитанному материалу. Каждый вопрос представляет некоторую карточку с заголовком, основной частью и счетчиком вопросов, навигация между вопросами осуществляется горизонтальными свайпами. Стоит отметить, что размеры каждого варианта ответа и размеры шрифтов как вопроса, так и вариантов ответов адаптируются под наличие свободного пространства для его заполнения.

Ниже представлен макет практического задания (рисунок 23)

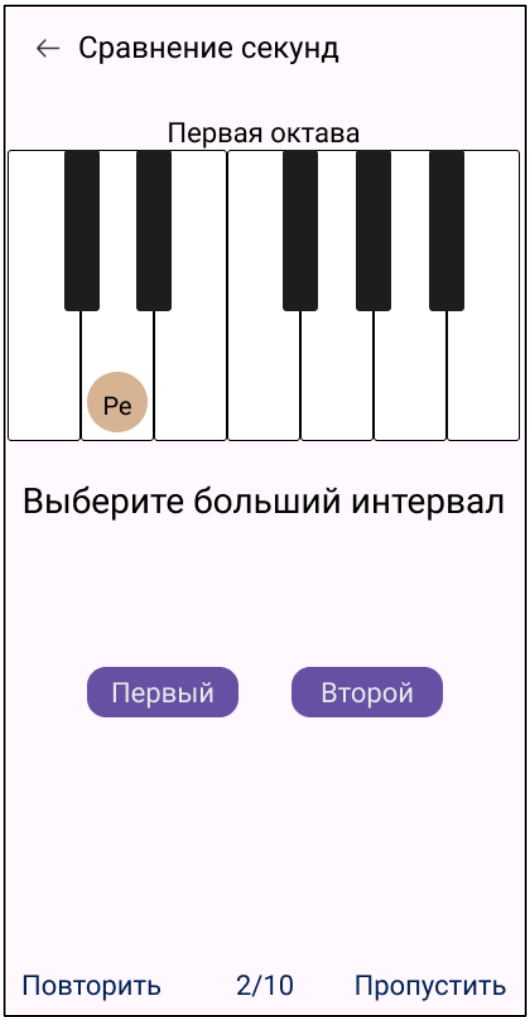

Рисунок 23 – Макет экрана практического задания

Приложение будет проигрывать нужные интервалы, далее пользователь должен дать ответ на поставленное задание. После приложение отобразит верный ответ и позволит перейти к следующему интервалу.

Вывод по второй главе

На основе анализа функциональных и нефункциональных требований были определены конкретные задачи, достижение которых необходимо при разработке системы. Анализ архитектурных паттернов и выделение принципов UI/UX-дизайна позволили выбрать самые подходящие шаблоны, которые помогут разработать легко поддающуюся изменениям, жизнеспособную систему. Описание компонентов системы и модели базы данных помогло глубже описать то, как должна выглядеть система внутри.

3. РЕАЛИЗАЦИЯ

3.1. Выбор технологий для реализации системы

Обзор технологий для реализации системы стоит рассматривать с двух позиций: с позиции выбора технологий для реализации серверного приложения и базы данных, а также с позиции выбора технологий для реализации мобильного приложения.

Для реализации серверного приложения был выбран язык программирования C# и высокопроизводительная платформа разработки веб-приложений ASP.NET Core [18], созданные компанией Microsoft. ASP.NET Core предоставляет мощный инструментарий для создания веб-приложений, включая поддержку маршрутизации, контроллеров, моделей представления и многое другое. Для доступа к базе данных из кода серверного приложения использован ORM-фреймворк (англ. Object-Relational Mapping, рус. объектно-реляционное отображение) под названием Entity Framework, который является самым популярным ORM-фреймворком для использования из языков, которые поддерживает платформа .NET [22], кроме этого, Entity Framework имеет встроенную защиту от SQL-инъекций [19], используя параметризированные запросы в БД, вместо простого подставления строк в строку с запросом [20].

Все нужные программные продукты и библиотеки уже поставляются с пакетом программ .NET SDK (англ. software development kit, рус. комплект для разработки программного обеспечения) и пакетом стандартных библиотек от компании Microsoft [21].

В качестве СУБД (системы управления базами данных) было выбрано одно из самых популярных – MS SQL. Работа с ним удобна, его производительность является достойной для масштабных проектов, следовательно при дальнейшем улучшении системы не будет требоваться замены СУБД.

В качестве интегрированной среды разработки для написания серверного приложения была выбрана IDE Visual Studio 2022, которая является

одной из самых популярных для разработки приложений на платформе .NET в силу наличия широкого набора инструментов и возможностей, которые значительно упрощают и ускоряют процесс разработки.

Языком для разработки мобильного приложения PianoMentor был выбран именно Kotlin в силу того, что этот язык является нативным языком для написания Android-приложений [23], что избавляет его от минусов кроссплатформенных технологий, популярность языка позволяет легко находить различные документации и советы по его использованию.

Отдельного рассмотрения требуют конкретные технологии View и Jetpack Compose. Их назначение – это предоставление удобных инструментов для создания интерфейсов.

Технология View представляет собой абстрактный класс [24], позволяющий управлять графическими элементами на экране приложения. По сути, реализуя класс View, можно создавать различные объекты, инкапсулирующие в себе весь функционал того или иного графического элемента, тем самым разработчик получает простой в освоении инструмент для создания новых, уникальных графических элементов. Также View предлагает обширный список различных свойств для управления визуальными элементами, что позволяет динамически управлять ими. Верстка каждого экрана и элемента происходит с использованием XML, это позволяет заранее прописать структуру и внешний вид экранов приложения, а при необходимости изменять в процессе выполнения программы.

В отличие от View, где для начала нужно создать верстку экрана с помощью языка разметки XML, Jetpack Compose позволяет создавать визуальные компоненты с помощью языка программирования Kotlin [25], такой подход называет декларативным подходом. Этот способ освобождает разработчика от нужды изучать XML верстку, и позволяет сосредоточить все силы на изучении Kotlin. Однако, не смотря на столь внушительное преимущество стоит отметить тот факт, что эта технология является молодой и не столь стабильной, как технология View. Помимо этого, для Jetpack Compose

не успело накопиться в достаточной мере за прошедшее с момента появления этой технологии время различной обучающей литературы, с помощью котором можно было бы быстро и уверенно разбираться в новом инструменте. В силу перечисленного выше мой выбор пал именно на технологию View, как технологию, зарекомендовавшую себя в течение большого периода времени.

Касаемо интегрированной среды разработки для написания мобильного приложения можно выделить одну главную и самую удобную IDE под названием Android Studio. Функционал данной среды разработки обширен и позволяет выполнять на столько широкий спектр операций, что на рынке даже не представлено достойной альтернативы, способной конкурировать с Android Studio.

3.2. Реализация процесса аутентификации пользователя

Одна из основных особенностей системы заключается в реализации системы учета пользователей. Процесс аутентификации пользователя в системе отражен в двух диаграммах деятельности: одна отображает процесс в рамках мобильного приложения (приложение В, рисунок 1), другая отображает процесс в рамках серверного приложения (приложение Г, рисунок 2).

Изначально пользователь открывает экран с полями ввода email и пароля от учетной записи (рисунок 24). В момент начала ввода пользователем данных от своей учетной записи, начинается процесс их валидации, результат которой отображается под полями ввода (рисунок 24). Код методов, производящих валидацию, можно увидеть в листинге 1.

Листинг 1 – Методы для валидации email и пароля

```
fun loginDataChanged(email: String, password: String) {
    when {
        !isEmailValid(email) -> _loginForm.value =
LoginFormState(usernameError = R.string.invalid_email)
        password.length < 6 -> _loginForm.value = LoginFormState(password-
LengthError = R.string.password_too_short)
        !containsLowercase(password) -> _loginForm.value =
LoginFormState(passwordLowercaseError = R.string.password_no_lowercase)
        !containsUppercase(password) -> _loginForm.value =
LoginFormState(passwordUppercaseError = R.string.password_no_uppercase)
```

```
        !containsDigit(password) ->_loginForm.value = LoginFormState(pass-
wordDigitError = R.string.password_no_digit)
        else -> _loginForm.value = LoginFormState(isDataValid = true)
    }
}
private fun isEmailValid(email: String): Boolean {
    return email.isNotEmpty() && PatternsCompat.EMAIL_AD-
DRESS.matcher(email).matches()
}
private fun containsLowercase(password: String): Boolean {
    return password.isNotEmpty() && password.any { it.isLowerCase() }
}
private fun containsUppercase(password: String): Boolean {
    return password.isNotEmpty() && password.any { it.isUpperCase() }
}
private fun containsDigit(password: String): Boolean {
    return password.isNotEmpty() && password.any { it.isDigit() }
```

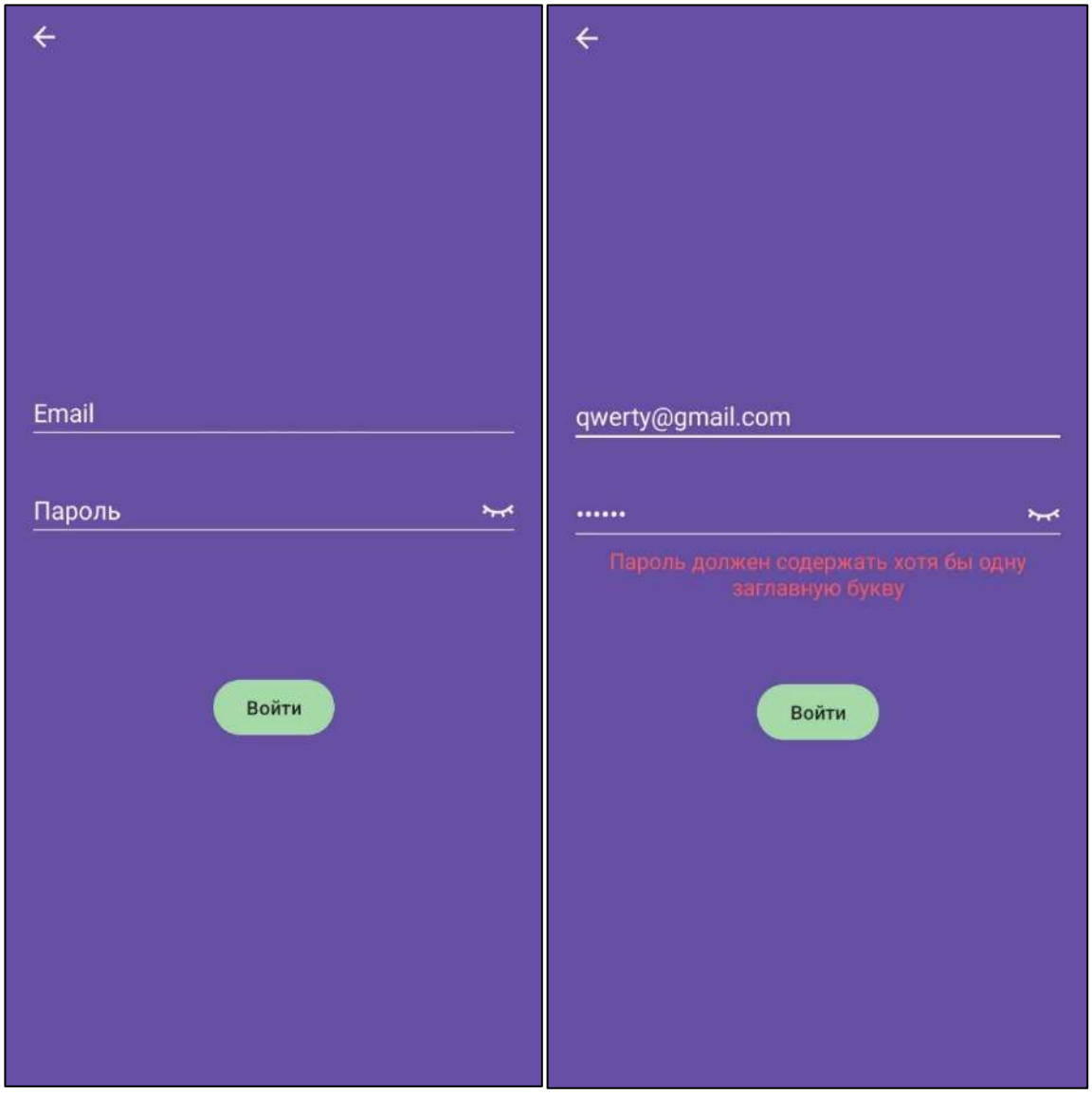

Рисунок 24 – Экран входа в учетную запись пользователя (слева), отображение подсказок при вводе данных (справа)

Когда пользователь введен данные, удовлетворяющие условиям валидации, то станет активной кнопка «Войти». После нажатия на эту кнопку произойдет получение значений из полей ввода (email и пароль) и запустится процесс отправки запроса на аутентификацию пользователя. Изначально этот запрос попадет в domain слой мобильного приложения, где сформируется и отправится HTTPS-запрос серверу с введенными пользователями email и паролем. В том же самом месте будет происходит прием ответа от сервера, тут будет происходить фактически первое преобразование ответа от сервера, в сущности, которыми будет оперировать само приложение. Такой подход позволяет разделить обязанности тех или иных классов внутри кода приложения и легче изменять код в случае необходимости. В листинге 2 можно увидеть метод, принимающий ответ сервера и производящий первую обработку данных.

Листинг 2 – Метод, принимающий и обрабатывающий ответ сервера

```
suspend fun login(request: LoginUserRequest): ActionResult<LoginUserRe-
sponse> {
    return try {val callResult = apiService.loginUser(request)
        when (callResult.failedMessage) {
            null -> {
                ActionResult.Success(callResult)
            }
            else -> {
                ActionResult.NormalError(callResult)
            }
        }
    } catch (e: Exception) {
        ActionResult.ExceptionError(IOException(e.message))
    }
}
```

Дальнейшая работа происходит на уровне серверного приложения, контроллер примет запрос и направит его соответствующему обработчику, который изначально попытается найти запись о пользователя в базе данных. Если запись не была найдена, то сервер вернет ошибку с кодом 404, означающим, что сущность не была найдена, иначе программа перейдет к проверке правильности пароля. Здесь также произойдет ветвление: если поступил на вход неправильный пароль, то сервер вернет ошибку с кодом 400, означающим, что запрос был сформирован не корректно, иначе обработка

запроса продолжится. Код поиска пользователя, проверки пароля приведен в листинге 3.

Листинг 3 – Код поиска пользователя и проверки пароля

```
var managedUser = await _userManager.FindByEmailAsync(request.Email);
if (managedUser == null)
{
	return new AuthUserResponse
	{
		FailedMessage = $"No accounts registered with {request.Email}"
	};
}

if (!await _userManager.CheckPasswordAsync(managedUser, request.Password))
{
	return new AuthUserResponse
	{
		FailedMessage = $"Incorrect password for {request.Email}"
	};
}
```

Далее будут запрошены сервером у базы данных роли пользователя, которые требуются для генерации access токена, являющегося своего рода ключами доступа к определенному функционалу сервера, а также к данным, принадлежащим конкретному пользователю. Далее создаются access токен и refresh токен, устанавливаются их срока годности и отправляется запрос на сохранение информации о начавшейся сессии в базу данных. В результате этой работы сервер отсылает обратно по сети мобильному приложению ответ, содержащий access токен, refresh токен и время их сроков годности. Код создания access токена и refresh токена, а также отправка результатов представлен в листинге 4.

Листинг 4 – Код создания access токена, refresh токена и отправка результата

```
var roles = await _userManager.GetRolesAsync(managedUser);

var (accessToken, accessTokenExpiryDateTime) = _tokenService.Cre-
ateAccessToken(managedUser, roles);
managedUser.RefreshToken = _tokenService.CreateRefreshToken();
managedUser.RefreshTokenExpireTime = DateTime.UtcNow.AddDays(_config.Get-
Section("Jwt:RefreshTokenValidityInDays").Get<int>());

var updatingResult = await _userManager.UpdateAsync(managedUser);

if (!updatingResult.Succeeded)
{
	return new AuthUserResponse
	{
		FailedMessage = $"Cannot update user information for {re-
quest.Email}"
```

```
      };
}


return new AuthUserResponse
{
      JwtTokensModel = new Contract.Models.JwtTokens.JwtTokensModel
      {
            AccessToken = accessToken,
            AccessTokenExpireTime = accessTokenExpiryDateTime,
            RefreshToken = managedUser.RefreshToken,
            RefreshTokenExpireTime = managedUser.RefreshTokenExpireTime
      },
      UserName = managedUser.UserName,
      Roles = roles,
      UserId = managedUser.Id,
      Email = request.Email
};
```

Мобильное приложение принимает запрос и обрабатывает его, как было уже сказано ранее. В дальнейшем удобные для оперирования внутри мобильного приложения данные проходят через domain слой и попадают в UI слой. Если мобильное приложение получило в качестве ответа ошибку, то эта ошибка будет обработана и отображена пользователю, иначе мобильное приложение запустит главный экран приложения. В листинге 5 можно увидеть код, отвечающий за отображение главного экрана приложения в случае удачной аутентификации пользователя, а также отвечающий за вывод сообщения об ошибке в процессе аутентификации.

Листинг 5 – Код, отвечающий за конечную обработку результата аутентификации

```
loginViewModel.loginResultUI.observe(viewLifecycleOwner,
    Observer { loginResult ->
        loginResult ?: return@Observer
        loadingProgressBar.visibility = View.GONE
        loginResult.error?.let {
            showLoginFailed(it)
            passwordEditText.text.clear()
        }
        loginResult.success?.let {
            updateUiWithUser(it)
            val options = NavOptions.Builder()
                .setLaunchSingleTop(false)
                .setPopUpTo(R.id.statisticsFragment, true)
                .build()
            findNavController().navigate(R.id.action_successful_loggedIn,
null, options)
        }
    })
```

Результат успешной аутентификации представлен на рисунке 25.

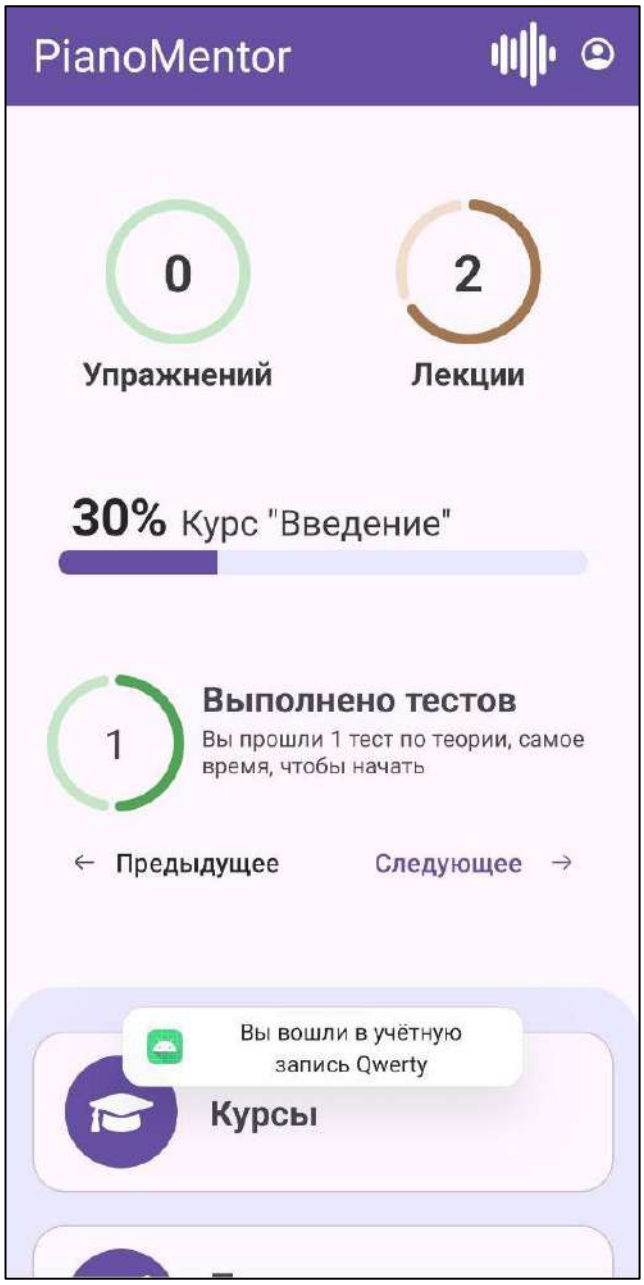

Рисунок 25 – Главный экран приложения после успешной аутентификации

3.3. Реализация отображения вопросов теста

При дальнейшем рассмотрении стоит обратить внимание на часть функционала, связанного с тестами на знание материала пройденных лекций, а именно на процесс отображения вопросов теста. Диаграмму, визуализирующую данное явление можно увидеть на рисунке 3 приложения Д.

Изначально пользователь должен перейти на экран теста, после чего мобильное приложение в отдельном потоке запросит все вопросы этого теста. Перед выполнением этого потока следует рассмотреть взаимодействие

между пользователем и объектом QuizViewModel, которое на диаграмме отображено с помощью фрагмента alt (alterative) для ветвления. Процесс прохождения теста сохраняется на сервере, поэтому важно узнать намерения пользователя перед отображением теста. Программа определит текущее состояние теста: если тест провален, появится диалоговое окно с вопросом о дальнейших действиях (это может быть просмотр последней попытки или перепрохождение теста). Если тест решен правильно, новая попытка не предоставляется. Если тест начат, но не завершен, показывается последняя попытка с возможностью ее завершить. Листинг 6 показывает метод для отображения диалогового окна.

Листинг 6 – Метод, отвечающий за отображение диалогового окна

```
fun showAlertDialog(quizStatus: CourseItemProgressType, context: Context,
quizCompleteButton: MaterialButton) {
    when (quizStatus) {
        CourseItemProgressType.FAILED -> {
            AlertDialog.Builder(context)
                .setTitle(context.getString(R.string.quiz_failed))
                .setMessage(context.getString(R.string.re-
start_quiz_or_show_results))
                .setPositiveButton(context.getString(R.string.re-
start_quiz)) { _, _ ->
                    quizCompleteButton.visibility = View.VISIBLE
                    _alertDialogResult.postValue(true)
                }
                .setNegativeButton(context.getString(R.string.show_re-
sults)) { _, _ ->
                    quizCompleteButton.visibility = View.GONE
                    _alertDialogResult.postValue(false)
                }
                .setOnCancelListener {
                    quizCompleteButton.visibility = View.GONE
                    _alertDialogResult.postValue(false)
                }
                .show()
        }
        CourseItemProgressType.COMPLETED -> {
            quizCompleteButton.visibility = View.GONE
            _alertDialogResult.postValue(false)
        }
        else -> {
            _alertDialogResult.postValue(true)
        }
    }
}
```

На рисунке 26 представлено диалоговое окно с вопросом выбора дальнейшего действия.

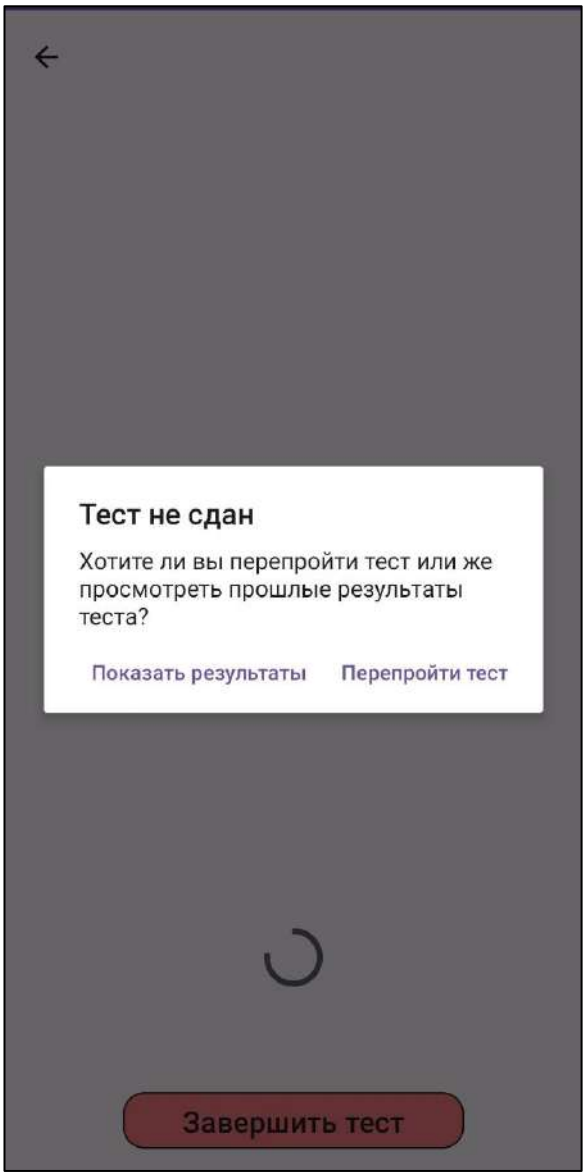

Рисунок 26 – Диалоговое окно с выбором дальнейшего действия

Далее стоит перейти к процессу получения вопросов теста. В целом, этот процесс по своей структуре не сильно отличается от аутентификации пользователя. Тут точно также формируется HTTPS-запрос, отправляющийся на сервер через сеть. После приема ответа от сервера этот запрос обрабатывается мобильным приложением. В листинге 7 представлен код, отправляющий запрос и принимающий ответ.

Листинг 7 – Код, отправляющий запрос и принимающий ответ

```
suspend fun getCourseItemQuiz(courseId: Int, courseItemId: Int, userId:
Long): ActionResult<GetQuizResponseApi> {
    return try {
        val result = apiService.getCourseItemQuiz(courseId, courseItemId,
userId)
```

```
        when (result.errors) {
            null -> ActionResult.Success(result)
            else -> ActionResult.NormalError(result)
        }
    } catch (e: Exception) {
        ActionResult.ExceptionError(IOException("${e.message}"))
    }
}
```

Серверное приложение принимает запрос на уровне контроллера, который передаст дальше этот запрос в обработчик. Процесс начнется с получения сущностей самих вопросов из базы данных, после будут получены последние ответы пользователя, если таковые были. В итоге все эти данные собираются в один ответ. В листинге 8 представлен код для получения вопросов теста с пользовательскими ответами.

Листинг 8 – Код, отвечающий за получение вопросов теста с ответами

```
var questions = _dbContext.QuizQuestions
      .AsNoTracking()
      .Where(q => q.CourseItemId == request.CourseItemId && !q.IsDeleted)
      .Select(q => new QuizQuestionModel
      {
            QuestionId = q.QuestionId,
            QuestionText = q.QuestionText,
            UpdatedAt = q.UpdatedAt,
            QuizQuestionType = q.QuizQuestionType.Name,
            AttachedDataSetId = q.AttachedDataSetId,
            CourseItemId = request.CourseItemId,
            Answers = q.QuizQuestionsAnswers
                  .Select(qa => new QuizQuestionAnswerModel
                  {
                        AnswerId = qa.AnswerId,
                        AnswerText = qa.AnswerText,
                        IsCorrect = qa.IsCorrect
                  })
                  .ToList()
      })
      .ToList();
var allAnswersIds = questions.SelectMany(q => q.Answers.Select(a => a.An-
swerId)).ToList();
var lastUserAnswers = await _dbContext.QuizQuestionUserAnswerLogs
      .AsNoTracking()
      .Where(al =>
            al.UserId == request.UserId
            && allAnswersIds.Contains(al.AnswerId)
            && al.AnsweredAt == _dbContext.QuizQuestionUserAnswerLogs
                  .Where(al2 => al2.UserId == request.UserId)
                  .Max(al2 => al2.AnsweredAt))
      .ToListAsync(cancellationToken);
foreach (var question in questions)
{
      foreach (var answer in question.Answers)
      {
            var lastUserAnswer = lastUserAnswers.FirstOrDefault(l => l.An-
swerId == answer.AnswerId);
            if (lastUserAnswer == null)
```

```
            {
                  continue;
            }
            answer.WasChosenByUser = true;
            answer.UserAnswerText = lastUserAnswer.UserAnswerText;
      }
}
return new GetQuizResponse(questions, null);
```

Полученный запрос преобразовывается в подходящие для мобильного приложения модели в `QuizRepository` и отправляется в `QuizViewModel`. Далее происходит отображение полученных данных. В листинге 9 показан код, отображающий данные вопроса на экране.

Листинг 9 – Код, отображающий данные вопроса на экране

```
questionCounter.text = String.format(context.getString(R.string.ques-
tion_counter), position + 1, models?.size)
questionTitle.text = String.format(context.getString(R.string.question_ti-
tle), position + 1)
questionText.text = question.questionText
if (question.attachedFile != null) {
    image.visibility = View.VISIBLE
    Glide.with(fragmentContext).load(question.attachedFile).into(image)
} else image.visibility = View.GONE
radioGroup.removeAllViews()
for (answer in question.answers) {
    val radioButton = RadioButton(fragmentContext).apply {
        id = answer.answerId
        text = answer.answerText
        textSize = 16f
        isChecked = answer.wasChosenByUser ?: false
        isEnabled = _startQuiz
        layoutParams = RadioGroup.LayoutParams(RadioGroup.Layout-
Params.MATCH_PARENT,RadioGroup.LayoutParams.WRAP_CONTENT).apply {
            weight = 1f
            setMargins(0, 3, 0, 3)
        }
        buttonDrawable = AppCompatResources.getDrawable(context, R.drawa-
ble.btn_radio_padding)
        background = if (_startQuiz) {
            AppCompatResources.getDrawable(context, R.drawable.btn_ra-
dio_background_correct)
        } else if (!answer.isCorrect) {
            AppCompatResources.getDrawable(context, R.drawable.btn_ra-
dio_background_incorrect)
        } else {
            AppCompatResources.getDrawable(context, R.drawable.btn_ra-
dio_background_correct)
        }
    }
    radioButton.setAutoSizeTextTypeUniformWithConfiguration(12, 24, 2,
TypedValue.COMPLEX_UNIT_SP)
    radioGroup.addView(radioButton)
}
```

В конце концов пользователь видит экран структурно такой же, как на рисунке 27.

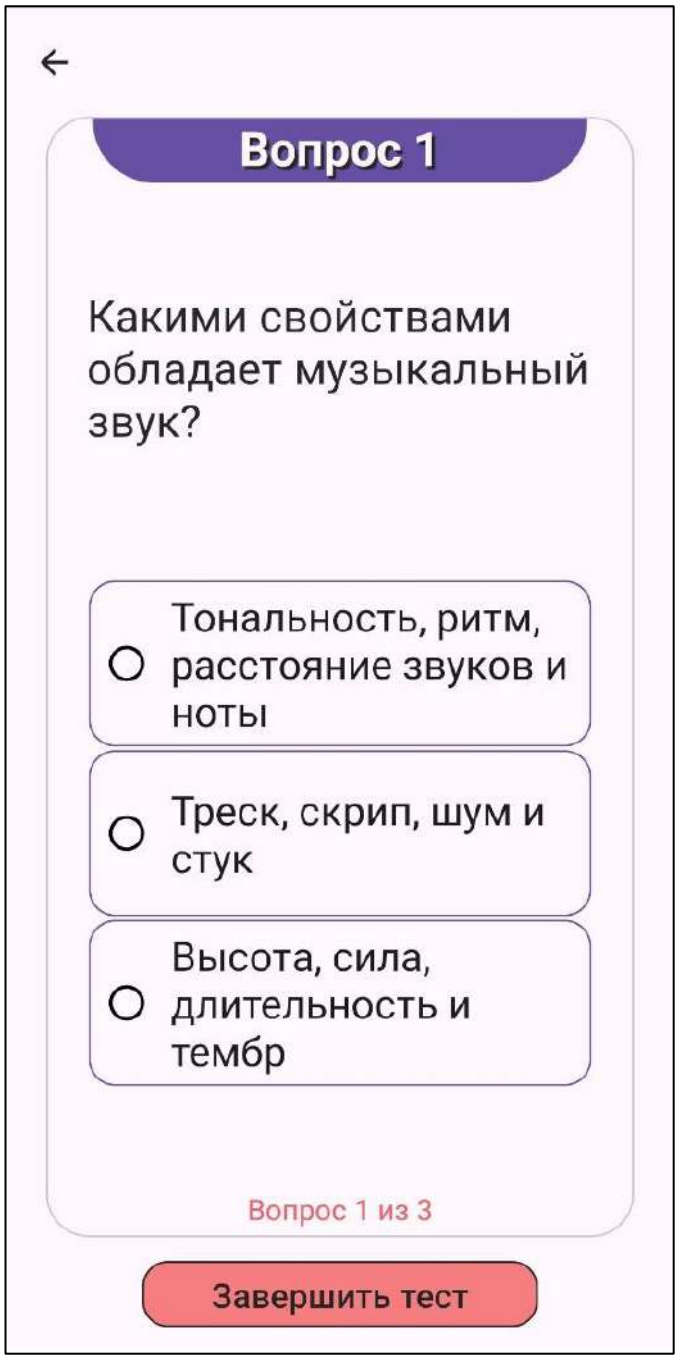

Рисунок 27 – Экран прохождения теста

Вывод по третьей главе

В рамках этой главы были рассмотрены два отдельных функциональных процесса в приложении, подробно описаны алгоритмы, протекающие в рамках этих процессов, освещены особенности реализации разных классов программы. В результате была создана полноценная система, состоящая из мобильного и серверного приложений и базы данных.

4. ТЕСТИРОВАНИЕ

Первостепенно было проведено функциональное тестирование, результаты которого приведены в таблице 8. Такой вид тестирования был проведён со стороны мобильного приложения, имеющего соединение с серверным приложением, что позволяет одновременно проверить функционал всей системы.

Таблица 8 – Результаты функционального тестирования

№	Описание	Шаги	Ожидаемый результат	Результат проверки
1	Загрузить фото профиля пользователя	1. Открыть профиль пользователя 2. Загрузить фотографию	Фотография загружена. Профиль отображает загруженную фотографию	Успешно
2	Сменить пароль от учетной записи	1. Открыть профиль пользователя 2. Сменить пароль от учетной записи	Пароль изменен, выведено сообщение об успешной смене пароля	Успешно
3	Зайти в приложение без входа в учетную запись	Войти в приложение	Статистика пуста, выведено соответствующее сообщение	Успешно
4	Просмотреть статистику обучения, войдя в учетную запись	1. Войти в учетную запись 2. Вернуться на главный экран	Актуальная статистика отображается	Успешно
5	Открыть раздел «Курсы»	Выбрать в меню раздел «Курсы»	Экран курсов открылся, отображаются актуальные курсы с актуальный прогрессом прохождения	Успешно
6	Открыть раздел «Лекции по теории»	Выбрать в меню раздел «Лекции по теории»	Отображается список всех лекций по теории из всех курсов	Успешно
7	Открыть раздел «Тесты по теории»	Выбрать в меню раздел «Тесты по теории»	Отображается список всех тестов по теории из всех курсов	Успешно
8	Открыть раздел «Практические занятия»	Выбрать в меню раздел «Практические занятия»	Отображается список всех практических занятий из всех курсов	Успешно
9	Открыть лекцию	1. Открыть раздел «Курсы» 2. Выбрать курс 3. Выбрать лекцию	Экран лекции открылся, лекция загрузилась	Успешно

Продолжение таблицы 8

№	Описание	Шаги	Ожидаемый результат	Результат проверки
10	Открыть тест, предварительно войдя в учетную запись	1. Войти в учетную запись 2. Открыть раздел «Курсы» 3. Выбрать курс 4. Выбрать тест	Экран теста открылся, вопросы загрузились	Успешно
11	Открыть тест, не входя в учетную запись	1. Открыть раздел «Курсы» 2. Выбрать курс 3. Выбрать тест	Экран теста не открывается, выводится сообщение о необходимости войти в учетную запись	Успешно
12	После прочтения лекции сохраняется прогресс	1. Открыть раздел «Курсы» 2. Выбрать курс 3. Выбрать лекцию 4. Прочитать лекцию	Прогресс прочтения лекции сохранен	Успешно
13	Решить тест неправильно	1. Открыть раздел «Курсы» 2. Выбрать курс 3. Выбрать тест 4. Пройти тест с ошибками	Открывается экран, сообщающий о провале теста. Сохраняется результат провала теста	Успешно
14	Решить тест правильно	1. Открыть раздел «Курсы» 2. Выбрать курс 3. Выбрать тест 4. Пройти тест с ошибками	Открывается экран, сообщающий об успешном прохождении теста. Сохраняется результат успеха	Успешно
15	Выйти из теста, не завершая его	1. Открыть раздел «Курсы» 2. Выбрать курс 3. Выбрать тест 4. Выбрать некоторые ответы 5. Нажать кнопку «Назад»	Промежуточные ответы сохранены, прогресс прохождения теста сохранен	Успешно
16	Пройти практическое занятие с ошибками	1. Открыть раздел «Курсы» 2. Выбрать курс 3. Выбрать практическое занятие 4. Пройти занятие с ошибками	Отображается экран провала практического занятия. Сохраняется статистика провала занятия	Успешно

Окончание таблицы 8

№	Описание	Шаги	Ожидаемый результат	Результат проверки
17	Пройти практическое занятие без ошибок	1. Открыть раздел «Курсы» 2. Выбрать курс 3. Выбрать практическое занятие 4. Пройти занятие без ошибок	Отображается экран, сообщающий об успешном прохождении занятия. Сохраняется результат успеха	Успешно
18	Аутентифицироваться под существующим пользователем	1. Открыть приложение 2. Открыть экран входа в учетную запись 3. Ввести корректные учетные 4. Нажать «Войти»	Пользователь успешно вошел в учетную запись. Отображается главный экран с актуальной статистикой	Успешно
19	Аутентифицироваться под несуществующим пользователем	1. Открыть приложение 2. Открыть экран входа в учетную запись 3. Ввести некорректные данные 4. Нажать «Войти»	Пользователь не входит в учетную запись, отображается ошибка входа	Успешно
20	Зарегистрироваться, используя email, ранее не занятый другим пользователем	1. Открыть приложение 2. Открыть экран регистрации 3. Ввести корректные данные 4. Нажать «Зарегистрироваться»	Новая учетная запись создана, пользователь автоматически входит в новую учетную запись	Успешно
21	Зарегистрироваться, используя email, ранее занятый иным пользователем	1. Открыть приложение 2. Открыть экран регистрации 3. Ввести некорректные данные 4. Нажать «Зарегистрироваться»	Новая учетная запись не создана, выводится ошибка создания учетной записи	Успешно

Помимо функционального тестирования было проведено тестирование эргономичности мобильного приложения (юзабилити-тестирование). Такая проверка применятся для оценки удобства использования той или

иной программы. В качестве тестировщиков набирается фокус-группа, каждому члену которой даётся несколько заданий. Модератор тестирования в свою очередь собирает обратный отзыв от тестировщиков.

Для проведения такого рода тестирования были подобраны участники, каждому из которых поочерёдно был предоставлен смартфон с предустановленной программой PianoMentor. Важно заметить, что ранее ни один участник не видел данного приложения и не был знаком с его спецификой.

После изложения общей направленности приложения и тематики исследования, каждому участнику были выданы задания и выслушаны вопросы участников, касающиеся этих заданий. Каждое из них представляло собой просьбу сделать некоторое действие в приложении, оно должно быть выполнено без использования сторонних источников информации, перед пользователем должен быть лишь интерфейс приложения.

В процессе тестирования были собраны заметки фокус-группы, а по окончанию был проведён устный блиц-опрос для закрепления тезисов, высказанных тестировщиками. Общий отзыв участников тестирования был положительным, они подметили приятную и необычную цветовую гамму приложения, интуитивно понятный интерфейс, большие шрифты, наличие анимаций, позволяющих достигнуть лучшего опыта использования. Полный отчет об юзабилити-тестировании можно увидеть в приложении Б.

Вывод по четвертой главе

По окончанию функционального тестирования можно сделать вывод о том, что система работает устойчиво, не было обнаружено ошибок, функции отрабатывают корректно за приемлемое время. Такого результата удалось достичь благодаря использованию многоуровневой архитектуры, различных паттернов проектирования. Юзабилити-тестирование завершено удачно, фокус-группа дала общий положительный отзыв об удобстве пользования мобильным приложением. Этого удалось достичь благодаря предварительному определению принципов построения UI/UX-дизайна, продуманным заранее макетам основных экранов мобильного приложения.

ЗАКЛЮЧЕНИЕ

В рамках выпускной квалификационной работы было создано мобильное приложение для платформы Android под названием PianoMentor, которое предназначено для обучения людей игре на пианино по средствам как теоретических, так и практических занятий. При этом были решены следующие задачи:

1) произведен обзор литературы и существующих приложений в рамках предметной области;

2) проведен анализ как функциональных, так и нефункциональных требований к системе;

3) выбрана архитектура приложения согласно передовым наработкам в этой области, применены популярные паттерны проектирования, доказавшие свою жизнеспособность с течением времени;

4) разработана структура базы данных, которая позволяет удобно ее масштабировать;

5) разработан UI/UX-дизайн мобильного приложения;

6) реализованы мобильное приложение, серверное приложение и база данных;

7) проведено функциональное тестирование и юзабилити-тестирование системы.

В будущем планируется продолжать наращивать функционал приложения, одной из особенностей, которая будет добавлена первостепенно, является возможность анализа звука с помощью микрофона телефона, чтобы ученик мог проходить практические занятия, используя не виртуальное пианино, а настоящее пианино. После будет добавлен функционал обмена файлами между пользователями, в том числе между учениками и преподавателями, использующими это приложение, что позволит вывести приложение на новый уровень обучения. Помимо этого, планируется изменение профиля пользователя с целью обновления как внешнего вида личного кабинета, так и возможности создать доску достижений пользователя.

ЛИТЕРАТУРА

1. Мартин Р. Чистая архитектура. Искусство разработки программного обеспечения. // СПб.: Питер, 2021. – 352 с.

2. Что такое Interactor | Stack Overflow на русском. [Электронный ресурс] URL: https://ru.stackoverflow.com/questions/674745/Что-такое-interactor (дата обращения: 25.02.2024 г.).

3. MediatR | GitHub. [Электронный ресурс] URL: https://github.com/jbogard/MediatR (дата обращения 28.05.2024 г.)

4. Вам нужен Mediator | Habr. [Электронный ресурс] URL: https://habr.com/ru/articles/734302/ (дата обращения 28.05.2024 г.)

5. Вероятно, вам не нужен MediatR | Habr. {Электронный ресурс] URL: https://habr.com/ru/articles/686278/ (дата обращения 28.05.2024 г.)

6. What is Message Oriented Middleware | GeeksforGeeks. [Электронный ресурс] URL: https://www.geeksforgeeks.org/what-is-message-oriented-middleware-mom/ (дата обращения 28.05.2024 г.)

7. MVVM: полное понимание (+WPF) Часть 1 | Habr. [Электронный ресурс] URL: https://habr.com/ru/articles/338518/ (дата обращения 28.05.2024 г.)

8. The MVVM Pattern | Microsoft Learn. [Электронный ресурс] URL: https://learn.microsoft.com/ru-ru/previous-versions/msp-n-p /hh848246(v=pandp.10) (дата обращения: 28.05.2024 г.)

9. Гамма Э., Хелм Р., Джонсон Р., Влиссидес Дж. Паттерны объектно-ориентированного проектирования. // СПб.: Питер, 2020. – 448 с.

10. Model-View-ViewModel (MVVM) | Microsoft Learn. [Электронный ресурс] URL: https://learn.microsoft.com/ru-ru/dotnet/architecture/maui/mvvm (дата обращения 28.05.2024 г.)

11. Overview of ASP.NET Core MVC | Microsoft Learn. [Электронный ресурс] URL: https://learn.microsoft.com/en-us/aspnet/core/mvc/overview?view=aspnetcore-8.0 (дата обращения: 28.05.2024 г.)

12. Difference Between MVC and MVVM Architecture Pattern | GeeksforGeeks. [Электронный ресурс] URL: https://www.geeksforgeeks.org/difference-between-mvc-and-mvvm-architecture-pattern-in-android/ (дата обращения: 28.05.2024 г.)

13. The Difference Between MVVM And MVC (With Pros and Cons) | Indeed. [Электронный ресурс] URL: https://in.indeed.com/career-advice/career-development/difference-between-mvvm-and-mvc (дата обращения: 28.05.2024 г.)

14. Форматы файлов | Open Data Handbook. [Электронный ресурс] URL: https://opendatahandbook.org/guide/ru/appendices/file-formats/ (дата обращения: 28.05.2024 г.)

15. Архитектура REST | Habr. [Электронный ресурс] URL: https://habr.com/ru/articles/38730/ (дата обращения: 01.06.2024 г.)

16. Что такое протокол HTTPS и как он работает | Timeweb. [Электронный ресурс] URL: https://timeweb.com/ru/community/articles/chto-takoe-protokol-https-i-princip-ego-raboty (дата обращения: 01.06.2024 г.)

17. Рекомендации Google по созданию структуры URL | Центр Google Поиска. [Электронный ресурс] URL: https://developers.google.com/search/docs/crawling-indexing/url-structure?hl=ru (дата обращения: 01.06.2024 г.)

18. Обзор ASP.NET | Microsoft Learn. [Электронный ресурс] URL: https://learn.microsoft.com/ru-ru/aspnet/overview (дата обращения: 01.06.2024 г.)

19. Что такое SQL-инъекция? | Kaspersky. [Электронный ресурс] URL: https://www.kaspersky.ru/resource-center/definitions/sql-injection (дата обращения: 17.05.2024 г.)

20. Вопросы безопасности (Entity Framework) | Microsoft Learn. [Электронный ресурс] URL: https://learn.microsoft.com/ru-ru/dotnet/framework/data/adonet/ef/security-considerations (дата обращения: 01.06.2024 г.)

21. Пакеты SDK для проектов .NET | Microsoft Learn. [Электронный ресурс] URL: https://learn.microsoft.com/ru-ru/dotnet/core/project-sdk/overview (дата обращения: 01.06.2024 г.)

22. Обзор ORM для C#: что подойдет для проекта | Habr. [Электронный ресурс] URL: https://habr.com/ru/companies/simbirsoft/articles/659841/ (дата обращения 25.05.2024 г.)

23. Kotlin Programming Language. [Электронный ресурс] URL: https://kotlinlang.org/ (дата обращения: 01.06.2024 г.)

24. View | Android Developers. [Электронный ресурс] URL: https://developer.android.com/reference/android/view/View (дата обращения: 01.06.2024 г.)

25. Jetpack Compose – сила Android-разработки | Habr. [Электронный ресурс] URL: https://habr.com/ru/articles/722040/ (дата обращения: 01.06.2024 г.)

26. Нормализация отношений. Шесть нормальных форм. | Habr [Электронный ресурс] URL: https://habr.com/ru/articles/254773/ (дата обращения: 02.06.2024 г.)

ПРИЛОЖЕНИЯ

Приложение А. Спецификация вариантов использования

Спецификация вариантов использования мобильного приложения приведена в таблицах 1–8.

Таблица 1 – Спецификация прецедента «Управлять профилем пользователя»

Прецедент: Управлять профилем пользователя
ID: 1
Краткое описание: Изменить имя пользователя, загрузить фото профиля, изменить пароль от учётной записи
Главные актеры: Аутентифицированный пользователь
Второстепенные актеры: Нет
Предусловия: Пользователь аутентифицирован
Основной поток: 1. Прецедент начинается, когда пользователь открывает профиль 2. Пользователь изменяет имя, пользователь загружает фото профиля, изменяет пароль от учётной записи
Постусловия: Имя профиля изменено, фотография загружена, пароль изменён
Альтернативные потоки: I. Ошибка при загрузке фотографии 1. Приложение выводит сообщение об ошибке

Таблица 2 – Спецификация прецедента «Просмотреть статистику обучения»

Прецедент: Просмотреть статистику обучения
ID: 2
Краткое описание: Просматривать статистику прогресса выполнения заданий
Главные актеры: Аутентифицированный пользователь
Второстепенные актеры: Нет
Предусловия: Пользователь аутентифицирован
Основной поток: 1. Прецедент начинается, когда пользователь открывает экран статистики 2. Приложение отображает диаграммы с актуальными данными
Постусловия: Пользователь видит актуальную статистику
Альтернативные потоки: I. Ошибка при загрузке данных 1. Приложение выводит сообщение об ошибке

Таблица 3 – Спецификация прецедента «Читать лекции по теории»

Прецедент: Читать лекции по теории
ID: 3
Краткое описание: Читать лекцию по теории
Главные актеры: Аутентифицированный пользователь, неаутентифицированный пользователь
Второстепенные актеры: Нет
Предусловия: Отсутствуют
Основной поток: Прецедент начинается, когда пользователь открывает лекцию
Постусловия: Лекция открывается для чтения.
Альтернативные потоки: I. Ошибка при загрузке лекции 1. Приложение выводит сообщение об ошибке

Таблица 4 – Спецификация прецедента «Проходить тесты по теории»

Прецедент: Проходить тесты по теории
ID: 4
Краткое описание: Решать тесты по теории с целью закрепления знаний, полученных на лекции
Главные актеры: Аутентифицированный пользователь
Второстепенные актеры: Нет
Предусловия: Пользователь аутентифицирован
Основной поток: 1. Прецедент начинается, когда пользователь открывает тест 2. Пользователь проходит тест 3. Приложение отображает результат теста
Постусловия: 1. Тест открывается 2. Вопросы отображаются корректно 3. После прохождения теста отображается результат
Альтернативные потоки: I. Ошибка при загрузке теста 1. Приложение выводит сообщение об ошибке

Таблица 5 – Спецификация прецедента «Проходить практические занятия»

Прецедент: Проходить практические занятия
ID: 5
Краткое описание: Заниматься с помощью практических занятий
Главные актеры: Аутентифицированный пользователь
Второстепенные актеры: Нет
Предусловия: Пользователь аутентифицирован
Основной поток: 1. Прецедент начинается, когда пользователь открывает практическое занятие 2. Пользователь выполняет задание 3. Приложение оценивает выполнение задания
Постусловия: 1. Практическое занятие открывается 2. Задание выполняется пользователем 3. Выполнение задания оценивается
Альтернативные потоки: I. Ошибка при загрузке задания 1. Приложение выводит сообщение об ошибке

Таблица 6 – Спецификация прецедента «Сыграть на виртуальном пианино»

Прецедент: Сыграть на виртуальном пианино
ID: 6
Краткое описание: Использовать виртуальное пианино без прохождения каких-нибудь занятий
Главные актеры: Аутентифицированный пользователь, неаутентифицированный пользователь
Второстепенные актеры: Нет
Предусловия: Отсутствуют
Основной поток: 1. Прецедент начинается, когда пользователь открывает виртуальное пианино 2. Пользователь играет на виртуальном пианино
Постусловия: Виртуальное пианино доступно для игры
Альтернативные потоки: I. Ошибка при загрузке виртуального пианино 1. Приложение выводит сообщение об ошибке

Таблица 7 – Спецификация прецедента «Аутентифицироваться»

Прецедент: Аутентифицироваться
ID: 7
Краткое описание: Возможность войти в учетную запись пользователя
Главные актеры: Неаутентифицированный пользователь
Второстепенные актеры: Нет
Предусловия: Отсутствуют
Основной поток: 1. Прецедент начинается, когда пользователь открывает страницу входа в учетную запись 2. Пользователь вводит учетные данные 3. Пользователь нажимает кнопку «Войти»
Постусловия: Пользователь успешно вошел в учетную запись
Альтернативные потоки: I. Неверные учетные данные 1. Приложение выводит сообщение об ошибке

Таблица 8 – Спецификация прецедента «Зарегистрироваться»

Прецедент: Зарегистрироваться
ID: 8
Краткое описание: Возможность создать новую учетную запись пользователя
Главные актеры: Неаутентифицированный пользователь
Второстепенные актеры: Нет
Предусловия: Отсутствуют
Основной поток: 1. Прецедент начинается, когда пользователь открывает страницу регистрации новой учетной записи 2. Пользователь вводит необходимые данные 3. Пользователь нажимает кнопку «Зарегистрироваться»
Постусловия: 1. Новая учетная запись создана 2. Пользователь автоматически входит в новую учетную запись
Альтернативные потоки: I. Пользователь с такими данными существует 1. Приложение выводит сообщение об ошибке

Спецификация вариантов использования серверного приложения приведена в таблицах 9–20.

Таблица 9 – Спецификация прецедента «Зарегистрировать пользователя»

Прецедент: Зарегистрировать пользователя
ID: 9
Краткое описание: Создание учетной записи нового пользователя
Главные актеры: Клиент
Второстепенные актеры: Нет
Предусловия: Отсутствуют
Основной поток: 1. Прецедент начинается, когда клиент присылает запрос на эндпоинт регистрации 2. Серверное приложение проверяет корректность присланных данных 3. Серверное приложение создает новую запись в БД для нового пользователя
Постусловия: Новая учетная запись пользователя успешно создана
Альтернативные потоки: I. Неверный формат данных при регистрации 1. Возврат сообщения об ошибке клиенту II. Логин или email уже используются другим пользователем 1. Возврат сообщения об ошибке клиенту

Таблица 10 – Спецификация прецедента «Аутентифицировать пользователя»

Прецедент: Аутентифицировать пользователя
ID: 10
Краткое описание: Получение данных доступа к учетной записи пользователя
Главные актеры: Клиент
Второстепенные актеры: Нет
Предусловия: Отсутствуют
Основной поток: 1. Прецедент начинается, когда клиент присылает запрос на эндпоинт аутентификации 2. Серверное приложение проверяет корректность присланных данных 3. Серверное приложение отправляет данные доступа к учетной записи
Постусловия: Данные сформированы и отправлены

Окончание таблицы 10 приложения А

Альтернативные потоки: I. Неверный формат данных при аутентификации 1. Возврат сообщения об ошибке клиенту II. Email или пароль не верные 1. Возврат сообщения об ошибке клиенту

Таблица 11 –Спецификация прецедента «Восстановить access & refresh tokens»

Прецедент: Восстановить access & refresh tokens
ID: 11
Краткое описание: Отзыв старых access & refresh tokens, последующее создание новых с восстановлением доступа к сессии пользователя
Главные актеры: Клиент
Второстепенные актеры: Нет
Предусловия: Отсутствуют
Основной поток: 1. Прецедент начинается, когда клиент присылает запрос на соответствующий эндпоинт 2. Серверное приложение отзывает старые access & refresh tokens пользователя 3. Серверное приложение создает новые access & refresh tokens 4. Клиент получает новые токены
Постусловия: У пользователя восстановлен доступ к сессии с использованием новых access & refresh tokens
Альтернативные потоки: I. Ошибка в процессе обработки запроса 1. Серверное приложение возвращает сообщение об ошибке

Таблица 12 – Спецификация прецедента «Отозвать refresh token»

Прецедент: Отозвать refresh token
ID: 12
Краткое описание: Отзыв refresh token(-s) без восстановления доступа к сессии(-ям) пользователя(-ей)
Главные актеры: Клиент
Второстепенные актеры: Нет
Предусловия: Отсутствуют

Основной поток: 1. Прецедент начинается, когда клиент отправляет запрос на отзыв refresh token(-s) пользователя(-ей) 2. Серверное приложение отзывает refresh token(-s)
Постусловия: У пользователя(-ей) отозваны refresh token(-s), и доступ к текущим сессии(-ям) невозможно восстановить
Альтернативные потоки: I. Не удалось отозвать refresh token(-s) 1. Серверное приложение возвращает сообщение об ошибке

Таблица 13 – Спецификация прецедента «Получить данные о курсах»

Прецедент: Получить данные о курсах
ID: 13
Краткое описание: Получение данных о доступных курсах
Главные актеры: Клиент
Второстепенные актеры: Нет
Предусловия: Отсутствуют
Основной поток: 1. Прецедент начинается, когда клиент запрашивает данные о курсах 2. Серверное приложение извлекает данные о доступных курсах из базы данных 3. Серверное приложение отправляет данные о курсах клиентскому приложению
Постусловия: Клиентское приложение получает данные о доступных курсах и может их отобразить пользователю
Альтернативные потоки: I. Ошибка при обработке запроса на сервере 1. Серверное приложение возвращает сообщение об ошибке

Таблица 14 – Спецификация прецендента «Получить данные об элементе курса»

Прецедент: Получить данные об элементе курса
ID: 14
Краткое описание: Получение данных, описывающих элемент курса
Главные актеры: Клиент
Второстепенные актеры: Нет

Предусловия: Отсутствуют
Основной поток: 1. Прецедент начинается, когда клиент отправляет запрос на серверное приложение о получении данных об определенном элементе курса 2. Серверное приложение обрабатывает запрос и извлекает данные об элементе курса из базы данных 3. Серверное приложение отправляет данные об элементе курса клиенту
Постусловия: Клиент получает данные об элементе курса
Альтернативные потоки: I. Не удалось получить данные об элементе курса 1. Серверное приложение возвращает сообщение об ошибке

Таблица 15 – Спецификация прецедента «Получить файл с лекцией»

Прецедент: Получить файл с лекцией
ID: 15
Краткое описание: Получение PDF-документа с лекцией по той или иной теме
Главные актеры: Клиент
Второстепенные актеры: Нет
Предусловия: Отсутствуют
Основной поток: 1. Прецедент начинается, когда клиент отправляет запрос на получение файла лекции 2. Серверное приложение обрабатывает запрос и отправляет файл с лекцией
Постусловия: Клиент получает файл с лекцией
Альтернативные потоки: I. Файл с лекцией не найден 1. Серверное приложение возвращает сообщение об ошибке

Таблица 16 – Спецификация прецедента «Получить картинку, привязанную к вопросу теста»

Прецедент: Получить картинку, привязанную к вопросу теста
ID: 16
Краткое описание: Получение растрового изображения, привязанного к вопросу теста
Главные актеры: Клиент
Второстепенные актеры: Нет

Окончание таблицы 16 приложения А

Предусловия: Отсутствуют
Основной поток: 1. Прецедент начинается, когда клиент отправляет запрос на получение изображения, связанного с определенным вопросом теста 2. Серверное приложение обрабатывает запрос и предоставляет нужное растровое изображение
Постусловия: Клиентское приложение получает растровое изображение
Альтернативные потоки: I. Изображение не найдено 1. Серверное приложение возвращает сообщение об ошибке

Таблица 17 – Спецификация прецедента «Управлять данными статистики»

Прецедент: Управлять данными статистики
ID: 17
Краткое описание: Получение данных о статистике прохождения обучения пользователем и сохранения оных
Главные актеры: Клиент
Второстепенные актеры: Нет
Предусловия: Отсутствуют
Основной поток: 1. Прецедент начинается, когда клиентское приложение отправляет запрос на серверное приложение о получении данных статистики пользователя или для сохранения новых данных статистики пользователя 2. Серверное приложение обрабатывает запрос, либо извлекает данные о статистике прохождения обучения из базы данных, либо сохраняет присланные 3. Серверное приложение отправляет извлечённые данные статистики пользователя клиентскому приложению или же отправляет сообщение об успехе сохранения новых данных
Постусловия: Клиентское приложение получает данные о статистике прохождения обучения пользователем и может их отобразить
Альтернативные потоки: I. Не удалось получить данные статистики пользователя 1. Серверное приложение возвращает сообщение об ошибке

Таблица 18 – Спецификация прецедента «Управлять файловым менеджером»

Прецедент: Управлять файловым менеджером
ID: 18
Краткое описание: Позволяет загружать и скачивать пользовательские файлы, создавать одноразовую ссылку для скачивания файлов, которую может использовать любой пользователь, получивший эту одноразовую ссылку
Главные актеры: Клиент
Второстепенные актеры: Нет
Предусловия: Отсутствуют
Основной поток: 1. Прецедент начинается, когда пользователь загружает файл в приложение или запрашивает создание одноразовой ссылки для скачивания файла 2. Серверное приложение обрабатывает запрос и осуществляет загрузку файла или создание одноразовой ссылки для скачивания 3. При создании одноразовой ссылки серверное приложение генерирует уникальный идентификатор ссылки и сохраняет его в базе данных вместе с информацией о файле и временем истечения ссылки 4. Серверное приложение отправляет созданную ссылку пользователю 5. Пользователь может использовать полученную одноразовую ссылку для скачивания файла
Постусловия: Пользователь загружает файл в приложение или может скачать файл, используя одноразовую ссылку
Альтернативные потоки: I. Не удалось загрузить файл 1. Серверное приложение вернет сообщение об ошибке II. Не удалось найти файл для скачивания 1. Серверное приложение вернет сообщение об отсутствии файла

Таблица 19 – Спецификация прецедента «Управлять курсами»

Прецедент: Управлять курсами
ID: 19
Краткое описание: Возможность добавлять новые и редактировать старые курсы
Главные актеры: Клиент
Второстепенные актеры: Нет
Предусловия: Запрос должен происходить от клиента, через которого аутентифицирован администратор

Основной поток: 1. Прецедент начинается, когда клиент отправляет запрос в систему управления курсами 2. Администратор выбирает опцию добавления нового курса или редактирования существующего 3. При добавлении нового курса администратор заполняет необходимую информацию о курсе, такую как название, описание, уровень сложности и другие характеристики 4. При редактировании существующего курса администратор вносит изменения в соответствующие поля 5. Система управления курсами сохраняет внесенные изменения
Постусловия: Новый курс добавлен или существующий курс отредактирован с учетом внесенных изменений
Альтернативные потоки: I. Не удалось добавить новый курс 1. Серверное приложение вернет сообщение об ошибке II. Не удалось найти курс для редактирования 1. Серверное приложение вернет сообщение об отсутствии искомого курса III. Не удалось сохранить данные в редактируемый курс 1. Серверное приложение вернет сообщение об ошибке

Таблица 20 – Спецификация прецедента «Добавить файл к элементу курса»

Прецедент: Добавить файл к элементу курса
ID: 20
Краткое описание: Возможность прикрепить файл к элементу курса
Главные актеры: Клиент
Второстепенные актеры: Нет
Предусловия: Запрос приходит от клиента, через которого аутентифицирован администратор
Основной поток: 1. Прецедент начинается, когда администратор выбирает элемент курса, к которому нужно добавить файл 2. Администратор загружает файл с компьютера 3. Система привязывает файл к выбранному элементу курса 4. Система сохраняет информацию о привязанном файле к элементу курса
Постусловия: Файл успешно прикреплен к элементу курса
Альтернативные потоки: I. Не удалось загрузить файл 1. Серверное приложение вернет сообщение об ошибке II. Не удалось найти искомый курс 1. Серверное приложение вернет сообщение об отсутствии курса

Приложение Б. Отчет об юзабилити-тестировании

Объект исследования

Объектом исследования является мобильное приложение для обучения игре на фортепиано по средствам теоретических лекций и практических заданий.

Метод исследования

Использован метод фокус-группы, заключающийся в привлечении к тестированию некоторой группы человек, которые могут быть заинтересованы в тестируемом продукте, им выдаются задания, которые нужно выполнить, используя исключительно интерфейс приложения.

План проведения тестирования

Для поэтапной характеристики процесса был составлен план, включающий следующие пункты:

1) разработка заданий для тестировщиков;
2) подбор фокус-группы;
3) объяснение каждому участнику всех аспектов исследования;
4) проведение тестирования;
5) сбор отзывов участников об их опыте использования приложения;
6) анализ собранных данных.

Описание процесса тестирования

Фокус-группа состояла из четырех человек разных возрастов и интересов, с каждым из которых была проведена личная беседа модератором тестирования с объяснением заданий. Каждый член фокус-группы, выполняя задание, может говорить сразу свои замечания, предложения и одобрения. Все подобные комментария записываются модератором исследования. По окончанию также даётся возможность фокус-группе ответить на блиц-опрос, нацеленный на получение структурированных ответов.

Протокол заданий

При проведении юзабилити-тестирования респондентам были поставлены следующие задачи:

1) открыть меню приложения;
2) зарегистрировать новую учётную запись;
3) загрузить картинку профиля;
4) изменить пароль от учётной записи;
5) открыть самую первую лекцию;
6) прочитать две страницы первой лекции и выйти на страницу выбора элементов курса;
7) прочитать полностью уже начатую лекцию;
8) определить, что значат индикаторы у каждого элемента курса;
9) просмотреть изменилась ли статистика прогресса обучения у данной учётной записи;
10) открыть первый тест;
11) не завершая тест, выйти на экран статистики, после вернуться к тесту;
12) решить тест с любым результатом;
13) открыть первое практическое занятие;
14) в процессе прохождения занятия, использовать повторение проигрывания интервала(-ов) и пропуска интервала(-ов);
15) пройти практическое занятие с любым результатом;
16) просмотреть прогресс обучения.

Полученные результаты

По окончанию юзабилити-тестирования были собраны отзывы участников фокус-группы. Анализ этих отзывов показал, что навигация в приложении понятна и проста, практические задания описаны лаконично, понимание теоретического материала не вызвало трудностей, тесты имеют умеренный уровень сложности, а редактирование профиля пользователя не вызвало затруднений. Тестирование пройдено успешно.

Приложение В. Диаграмма деятельности процесса аутентификации на стороне мобильного приложения

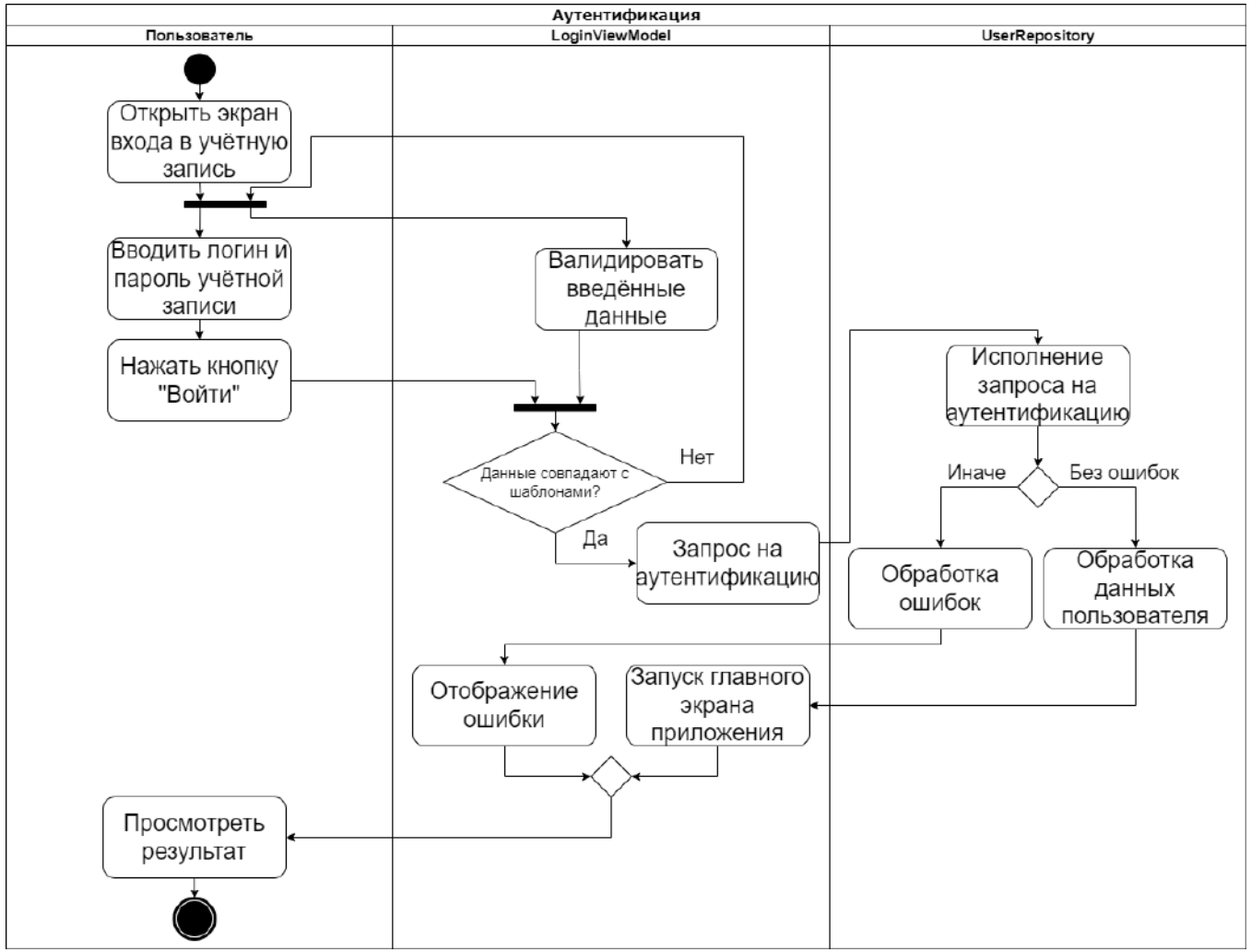

Рисунок 1 – Диаграмма деятельности «Аутентификация» в рамках мобильного приложения

Приложение Г. Диаграмма деятельности процесса аутентификации на стороне серверного приложения

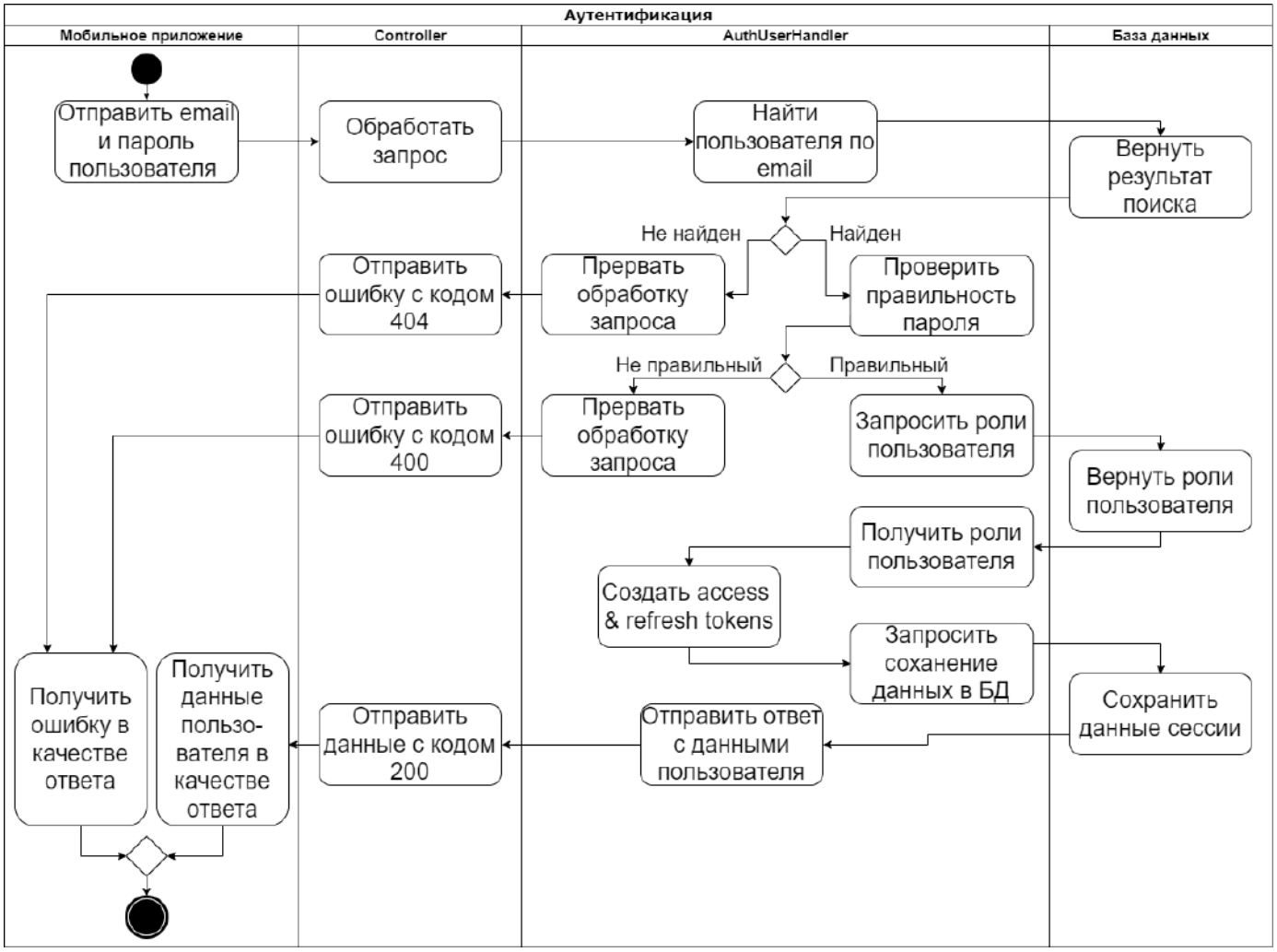

Рисунок 2 – Диаграмма деятельности «Аутентификация» в рамках серверного приложения

Приложение Д. Диаграмма последовательности процесса отображения вопросов теста

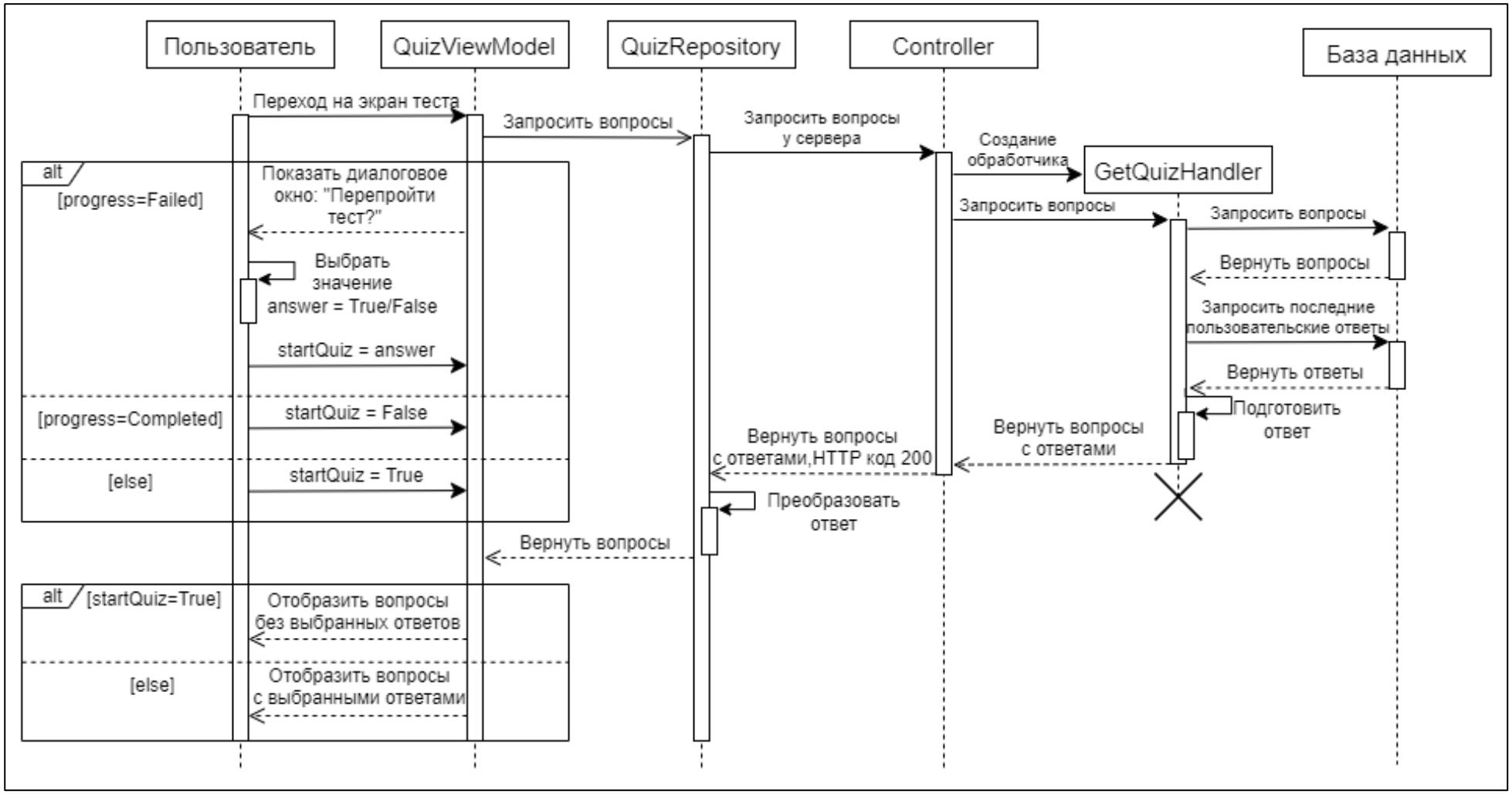

Рисунок 3 – Диаграмма последовательности «Получение вопросов теста»