Министерство образования, науки и молодежной политики Нижегородской области

Государственное бюджетное профессиональное образовательное учреждение

"Нижегородский Губернский колледж"

ОТЧЕТ ПО ДИСЦИПЛИНЕ

**МДК 01.03 РАЗРАБОТКА МОБИЛЬНЫХ ПРИЛОЖЕНИЙ**

Специальность **09.02.07 Информационные системы и программирование**

|  |  |  |
| --- | --- | --- |
| Выполнила | |  |
| Студентка | | М.А. Трянина |
|  | подпись |  |
| Группа 43П курс 4 | |  |
| Руководитель | | Ю.С. Мамшева |
|  | подпись |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

Нижний Новгород

2024

**СОДЕРЖАНИЕ**

[ВВЕДЕНИЕ 4](#_Toc183645722)

[1. ПОСТАНОВКА ЦЕЛЕЙ И ЗАДАЧ 5](#_Toc183645723)

[1.1 Анализ предметной области 5](#_Toc183645724)

[1.2 Выбор паттерна проектирования 11](#_Toc183645725)

[1.3 Проектирование требований к серверной части приложения 13](#_Toc183645726)

[1.4 Проектирование требований к мобильному приложению 15](#_Toc183645727)

[1.5 Выбор ПО для реализации 16](#_Toc183645728)

[2. РАЗРАБОТКА МАКЕТА ПРИЛОЖЕНИЯ 18](#_Toc183645729)

[3. РАЗРАБОТКА СЕРВЕРНОЙ ЧАСТИ ПРИЛОЖЕНИЯ 20](#_Toc183645730)

[3.1 Создание ER-диаграммы 20](#_Toc183645731)

[3.2 Создание и заполнение таблиц БД 21](#_Toc183645732)

[3.3 Учреждение политик доступа к базе данных 23](#_Toc183645733)

[4. ТЕСТИРОВАНИЕ СЕРВЕРНОЙ ЧАСТИ ПРИЛОЖЕНИЯ 25](#_Toc183645734)

[4.1 Тест-план 25](#_Toc183645735)

[4.2 Тестовый набор 27](#_Toc183645736)

[4.3 Тест-кейсы 29](#_Toc183645737)

[5. РАЗРАБОТКА МОБИЛЬНОГО ПРИЛОЖЕНИЯ 32](#_Toc183645738)

[5.1 Подключение шрифтов и настройка ранее выбранных цветов 32](#_Toc183645739)

[5.2 Верстка макета 33](#_Toc183645740)

[5.3 Подключение библиотек и интернета 36](#_Toc183645741)

[5.4 Подключение к серверной части приложения 37](#_Toc183645742)

[5.5 Навигация по страницам 38](#_Toc183645743)

[5.6 Разработка вспомогательных методов и методов для обработки данных с сервера 39](#_Toc183645744)

[6. ТЕСТИРОВАНИЕ МОБИЛЬНОГО ПРИЛОЖЕНИЯ 40](#_Toc183645745)

[6.1 Тест-план 40](#_Toc183645746)

[6.2 Тестовый набор 42](#_Toc183645747)

[6.2 Юнит-тесты 45](#_Toc183645748)

[ЗАКЛЮЧЕНИЕ 46](#_Toc183645749)

ВВЕДЕНИЕ

В современном мире цифровые платформы играют важную роль в объединении людей, разделяющих общие интересы. Одной из популярных сфер являются платформы, посвящённые фанатскому творчеству, которые дают авторам возможность делиться своими произведениями, а читателям — легко находить интересные материалы.

Целью данного проекта является разработка приложения-пополняемого репозитория фанатского творчества, которое станет удобным инструментом для публикации, поиска и взаимодействия с авторским контентом.

В рамках работы будут рассмотрены ключевые этапы: от постановки целей и задач до создания макетов, разработки серверной части и мобильного интерфейса, а также тестирования обоих компонентов.

1. ПОСТАНОВКА ЦЕЛЕЙ И ЗАДАЧ
   1. Анализ предметной области
   2. Общие требования

1.1 Назначение документа

Настоящее Техническое задание определяет требования и порядок разработки мобильного приложения «Склад» пополняемого репозитория фанатского творчества.

1.2 Наименование Исполнителя и Заказчика

* Заказчик – ИП MAMS; Адрес: с. Ольгино, ул. Лесная, 5;
* Исполнитель – ИП Baimors; Адрес: г. Н. Новгород, Московское ш., 1.

1.3 Информация о компании

1. Название: ИП MAMS;
   1. Отрасль и направления деятельности: развлечения;
   2. Перечень услуг:
      * Хранение репозитория текстовых работ;
      * Организация публикации текстовых работ;
      * Аналитика;
      * Организация подписок.
   3. Публичные контактные данные: +79625062815;

Конкуренты: Breakpoint SIA, Organization for Transformative Works.

* 1. Основание для разработки приложения

Основанием для разработки приложения является Договор № 00000001 от 25.09.2024 между Исполнителем и Заказчиком.

* 1. Назначение приложения

Назначением мобильного приложения «Склад» является организация доступа к ресурсам и услугам Заказчика, поддержание её положительного и современного имиджа, а также создание сообщества единомышленников, где пользователи могут обмениваться идеями и находить вдохновение.

Актуальность разработки приложения:

* приложение может предоставить пользователям инструменты для публикации работ;
* приложение может организовать комфортный доступ к обширному репозиторию работ.
  1. Целевая аудитория
  2. Пол: женский, мужской;
  3. Возраст: от 16 лет;
  4. Занятость: школьники, студенты, заинтересованные в фанатском творчестве;
  5. Цели и задачи приложения
  6. Тип приложения: мобильное приложение.
  7. Цели приложения:
  + Организация доступа к репозиторию работ;
  + Организация возможности добавления, изменения и удаления новых работ.
  1. Целевые действия пользователей приложения:
  + Авторизация пользователей;
  + Добавление/изменение/удаление текстовых работ;
  + Поиск и просмотр работ;
  + Добавление/удаление понравившихся работ и их просмотр;
  + Просмотр и изменение данных пользователя в профиле.
  1. Участники и роли

Таблица 1 – Участники и их роль в проекте

|  |  |
| --- | --- |
| Участники | Роли |
| М.А. Трянина | Разработка дизайна и работа с графическим контентом. Осуществление тестирования приложения. |
| Разработка и администрирование базы данных. Разработка функционала приложения и интерфейсов. |

* 1. Стадии разработки приложения

Таблица 2 – Стадии разработки проекта

|  |  |  |  |
| --- | --- | --- | --- |
| Стадии | Этапы | Ответственность | Сроки |
| Постановка целей и задач | Разработка технического задания | Трянина М.А. | 25.09.2024 – 25.09.2024 |
| Выбора паттерна проектирования |
| Проектирование требований к серверной части приложения |
| Проектирование требований к мобильному приложению |
| Выбор для реализации ПО и обоснование выбора |
| Разработка макета приложения | Выбор цветовой палитры | 09.09.2024 – 09.09.2024 |
| Выбор шрифта для заголовков и основного текста |
| Разработка макета каждого экрана разрабатываемого мобильного приложения |

Продолжение таблицы 2

|  |  |  |  |
| --- | --- | --- | --- |
| Разработка серверной части приложения | Проектирование серверной части |  | 23.10.2024 – 30.10.2024 |
| Разработка серверной части |
| Тестирование серверной части приложения | Разработка документации для тестирования сервера приложения | 13.11.2024 – 20.11.2024 |
| Unit-тесты |
| Разработка мобильного приложения | Подключение шрифтов и настройка ранее выбранных цветов | 20.11.2024  –  18.12.2024 |
| Верстка макета |
| Подключение библиотек и интернета |
| Подключение к серверной части приложения |
| Разработка вспомогательных методов, и методов для обработки данных с сервера |
| Тестирование мобильного приложения | Разработка документации для тестирования мобильного приложения | 18.12.2024 – 25.12.2024 |
| Unit-тесты |
| Мониторинг работоспособности и решение возникающих проблем. |
| Сопровождение и поддержка | Исправление ошибок и внесение изменений по запросам пользователей. | С 25.12.2024 |
| Модернизация, обновление и масштабирование системы. |
| Мониторинг работоспособности и решение возникающих проблем. |

* 1. Ресурсы

1. Человеческие ресурсы: разработчики и пользователи;
2. Инфраструктурные ресурсы: тестовая база данных;
3. Программные ресурсы: ОС Windows 10, инструменты разработки Visual Studio и Android Studio, графический онлайн-редактор Figma, система управления версиями Git, система хранения проектов Gogs, SaaS Supabase;
4. Информационные ресурсы: документация по языкам программирования.
   1. Нефункциональные требования
   2. Требования к персоналу

Технических специалистов узкого профиля для поддержания приложения не требуется, необходимыми и достаточными навыками будет владение общими понятиями работы с персональным компьютером на уровне базового пользователя и выше.

* 1. Требования к разграничению доступа

В соответствии с правами доступа можно выделить 1 категорию. пользователей сайта: Пользователи – доступ ко всей общедоступной информации в приложении, просмотр, редактирование и удаление.

* 1. Предварительная структура приложения

Основная страница приложения должна представлять собой страницу с новостями и событиями платформы, на которую выполняется переход после авторизации пользователя в системе.

Должна быть предусмотрена возможность перехода на следующие страницы с главной:

* Популярные работы;
* Читатели и авторы;
* Поиск работ;
* Фандомы;
* Профиль пользователя.
  1. Элементы макета приложения

Структура страниц приложения должна состоять из следующих элементов:

* «Навигационные элементы» – в данном блоке должны располагаться ссылки на соответствующие функциональные страницы, на которые можно перейти;
* «Шапка» – горизонтальный блок, расположенный наверху приложения – в данном блоке представлены возможности перехода на упомянутые выше страницы;
* «Основная часть» – часть страницы, на которой представлен функционал либо информация, соответствующие данному блоку.
  1. Требования к оформлению и дизайну
  2. Логотип: должен быть отображен;
  3. Предпочтительные цвета: оттенки зеленого, чёрного и коричневого;
  4. Возможные предпочтительные шрифты: Roboto, дополнительные.
  5. Количество вариантов дизайна

Основная тема приложения – светлая, дополнительно по требованию Заказчика может быть разработана темная тема.

* 1. Информационное наполнение приложения

Требуется. Представление информации, хранящейся в базе данных.

* 1. Функциональные требования
  2. Примерный набор модулей
  3. Регистрация и аутентификация пользователей;
  4. Добавление/изменение/удаление работ;
  5. Поиск и фильтрация работ;
  6. Управление личным кабинетом пользователя.
  7. Дополнительно
  8. Дополнительные пожелания

Дополнительные пожелания диктуются Заказчиком в процессе разработки.

* 1. Выбор паттерна проектирования

Основой дальнейшего проектирования и разработки является выбор паттерна.

1. The Model—View—Controller (MVC) Pattern – шаблон, в которое приложение делится на на три компонента (Model – бизнес-логика, View – видимая область, Controller – основная логика).

Схематичное построение данного шаблона можно увидеть на рисунке 1.

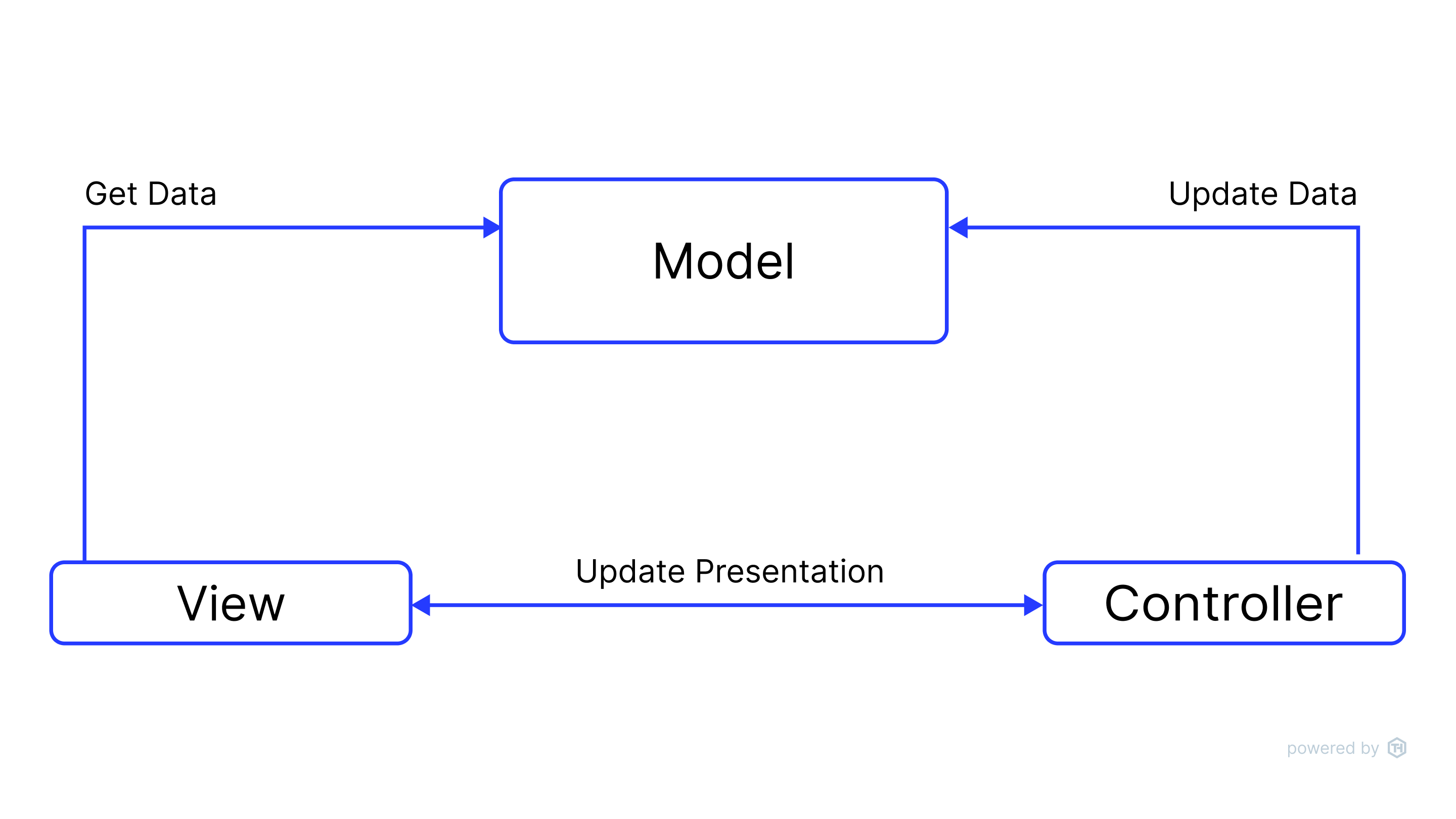


Рисунок 1 – MVC

Данный паттерн подлежит применению из-за реализуемости в нем бизнес-логики, поддержки асинхронных методов, изолированности всей модели от модификаций и в целом довольного быстрого процесса разработки. Однако, MVC может быть достаточно захламленным из-за большого количества кода в контроллере, а также не подвержен модульному тестированию.

1. The Model—View—Presenter (MVP) Pattern – паттерн, изолирующий логику представления от интерфейса для упрощения тестирования и поддержки. Также состоит включает в себя три компонента (Model – бизнес-логика, View – пользовательский интерфейс, Presenter – представление данных из модели).

Схематичное построение данного шаблона можно увидеть на рисунке 2.

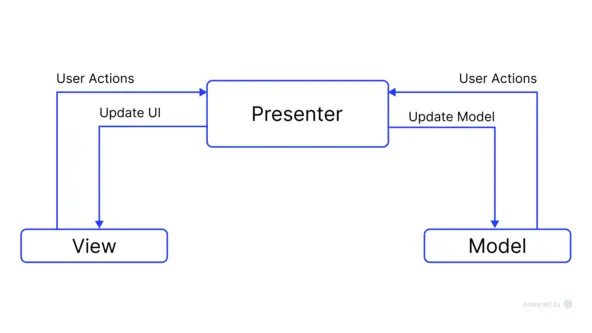


Рисунок 2 – MVP

Среди преимуществ данного шаблона: делает View абстрактной, так, чтобы ее можно было легко поменять; переиспользует View и Presenter, имеет сравнительно более читабельный и удобный в сопровождении код, а также удобен и прост в тестировании. Среди недостатков: имеет тесную связь между View и Presenter, реализует огромное количество интерфейсов между слоями, но также имеет довольно избыточный размер кода.

1. The Model—View—ViewModel (MVVM) Pattern – паттерн, реализующий двустороннюю привязку данных между View и ViewModel для упрощения взаимодействия. Точно так же состоит из 3 компонентов: Model – бизнес-логика, View – пользовательский интерфейс, ViewModel – большая часть логики пользовательского интерфейса. Изменения в ViewModel автоматически отражаются в View, что упрощает управление состоянием.

Схематичное построение данного шаблона можно увидеть на рисунке 3.

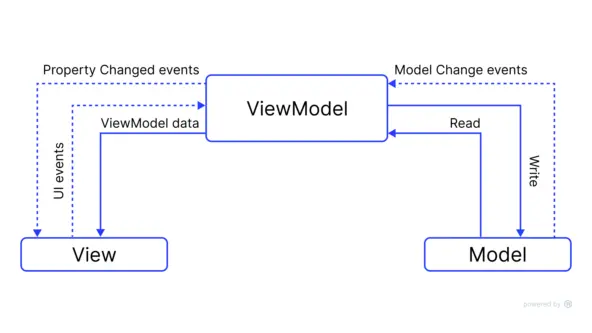


Рисунок 3 – MVVM

Сочетает в себе многие преимущества, в том числе разделение логики приложения и представления, гибкость и переиспользование, прямое связывание данных, а также упрощение тестирования. Среди недостатков – опять же, лишь большой объем кода.

Итак, ввиду некоторых особенностей, включая тематику приложения, язык и среду разработки, а также вышеописанных преимуществ и недостатков в дальнейшем приложение будет основано на паттерне MVVM.

* 1. Проектирование требований к серверной части приложения

Для организации хранения данных приложения необходимо спроектировать основные сущности будущей базы данных. Так как на физическом уровне сущности соответствует таблица, то перечислим основные таблицы БД.

В реляционную модель проектированной БД будут входить следующие таблицы (сущности): данные о пользователях (Authors), данные о работах (Works), смежная таблица для описания связи между пользователями и их любимыми работами (FavWorks), данные о фандомах (Fandoms), данные о жанрах (Genres), смежная таблица для описания связи между жанрами и работами (WorkGenres), смежная таблица для описания связи между фандомами и работами (WorkFandoms), таблица с галереей картинок работы (Gallery).

Назначения всех таблиц (сущностей), имеющихся в базе данных, можно увидеть в таблице 3.

Таблица 3 – Список сущностей

|  |  |  |
| --- | --- | --- |
| № | Название | Данные |
| 1 | Пользователи | Данные о пользователях |
| 2 | Работы | Данные о работах |
| 3 | Жанры | Данные о жанрах |
| 4 | Фандомы | Данные о фандомах |
| 5 | Любимые работы | Смежная таблица для хранения работ, отмеченных пользователем как любимые |
| 6 | Жанры работ | Смежная таблица для описания присущих работам жанров |
| 7 | Фандомы работ | Смежная таблица для описания принадлежности работ фандомам |
| 8 | Галерея | Данные о привязанных к работам картинках |

Данные таблицы связывают определённые отношения, которые отражены в таблице 4.

Таблица 4 – Список связей

|  |  |  |  |
| --- | --- | --- | --- |
| № | Название связи | Сущности, участвующие в связи | Назначение |
| 1 | 1:М | Пользователь-Работа | У одного пользователя может быть много работ. Одна работа может принадлежать только одному пользователю. |
| 2 | М:М | Пользователь-Работа | У одного пользователя может быть много любимых работ. У одной работы может быть много пользователей, кому она понравилась. |
| 3 | 1:М | Работа-Галерея | У одной работы может быть много картинок. Картинка может принадлежать одной работе. |

Продолжение таблицы 4

|  |  |  |  |
| --- | --- | --- | --- |
| 4 | М:М | Работа-Фандом | У одного фандома может быть много работ. У одной работы может быть много фандомов. |
| 5 | М:М | Работа-Жанр | У одного жанра может быть много работ. У одной работы может быть много жанров. |

Помимо вышеперечисленного, стоит отметить, что, раз база данных будет развернута в облачном инструментальном сервисе Supabase, имеющем собственные механизмы авторизации и хранения медиаданных, данные для регистрации и авторизации (такие, как логин и пароль) будут храниться в таблице с пользователями в соответствующей схеме «auth», с которой и будет производиться взаимодействие в дальнейшем.

Помимо этого, Supabase предоставляет хранилище для разного рода медиаданных для последующей привязки их в базе данных.

* 1. Проектирование требований к мобильному приложению

Ниже приведен ориентировочный перечень общих требований к мобильному приложению, которые настоятельно рекомендованы к соблюдению:

1. Предоставление доступа к интернету. Для корректной работы приложения необходимы внешние подключения;
2. Предоставление доступа к камере и внутреннему хранилищу. Для возможности заполнения галереи работы, необходимо предоставить соответствующие разрешения к медиаресурсам;
3. Версия Android 5.0 (Lollipop) или более поздняя;
4. Создание учетной записи и регистрация. Доступ к контенту должны иметь только авторизованные пользователи;
5. Поиск и фильтрация. Пользователь должен иметь возможность найти нужную ему работу;
6. Достижения. По достижению определенного количественного значения характеристик, пользователь должен получить соответствующее достижение;
7. Возможность добавления работ в избранное;
8. Возможность пользователю редактировать свои личные данные;
9. Возможность автору производить добавление, редактирование и удаление работ.
   1. Выбор ПО для реализации

Определившись с примерным функциональным наполнением приложения и получив ограничение на IDE в виде Android Studio, необходимо задуматься над средствами реализации, а именно над платформой разработки.

Далее рассмотрим преимущества и недостатки нескольких из них.

1. React Native:

Платформа для разработки на языках JavaScript, JSX, TypeScript.

Главные ее особенности: кроссплатформенность, быстрая разработка, простота использования, большое сообщество и множество готовых библиотек.

Из недостатков: ограничения в доступе к нативным функциям устройств, возможная более низкая производительность, есть повышенный шанс возникновения проблем с совместимостью и обновлениями.

1. Xamarin:

Платформа для разработки на C#, F#.

Из преимуществ: быстрая разработка, возможность использования единого кода для всех платформ, полная поддержка .NET и C#, доступ к нативным API платформ, интеграция с Visual Studio.

Недостатки: может быть сложным для начинающих разработчиков, ограниченное количество готовых библиотек по сравнению с React Native и Flutter, размер приложений может быть больше.

1. Kotlin (Java):

Платформа для разработки соответственно на Kotlin (Java).

Положительные стороны: быстрая разработка, лучшая производительность и оптимизация, полный доступ к нативным API и функциональности, поддержка со стороны Apple и Google.

Среди недостатков основные: относительно новый фреймворк, ограниченное количество библиотек и инструментов, не кроссплатформенно (нужно писать отдельный код для каждой платформы, для iOS будет Swift), более длительный процесс разработки и тестирования.

Ввиду вышеописанного, для дальнейшей разработки была выбрана платформа с языком Kotlin для Android. Он официально поддерживается Google, обладает современным и удобным синтаксисом, обеспечивает высокую производительность и совместимость с Java, требует сравнительно меньше шаблонного кода, предоставляет Coroutines для асинхронного программирования, а также гарантирует безопасность и надежность кода, поддерживается активным сообществом и легко интегрируется с другими инструментами.

1. РАЗРАБОТКА МАКЕТА ПРИЛОЖЕНИЯ

Одним из важнейших этапов создания любого приложения является проектирование пользовательского интерфейса, который обеспечивает интуитивное и приятное взаимодействие, а, следовательно, больший охват и продаваемость. Для нашего приложения был разработан макет в онлайн-сервисе Figma, где были проработаны основные экраны и сценарии взаимодействия.

Целью макета стало создание такого интерфейса, который позволит пользователям легко находить, читать и добавлять работы, а авторам – комфортно публиковать и редактировать свои произведения.

Далее, на рисунках 4-7 представлены все разработанные экраны.



Рисунок 4 – Интро, авторизация, регистрация, главная страницы



Рисунок 5 – Профиль, редактирование профиля, избранные работы, собственные работы

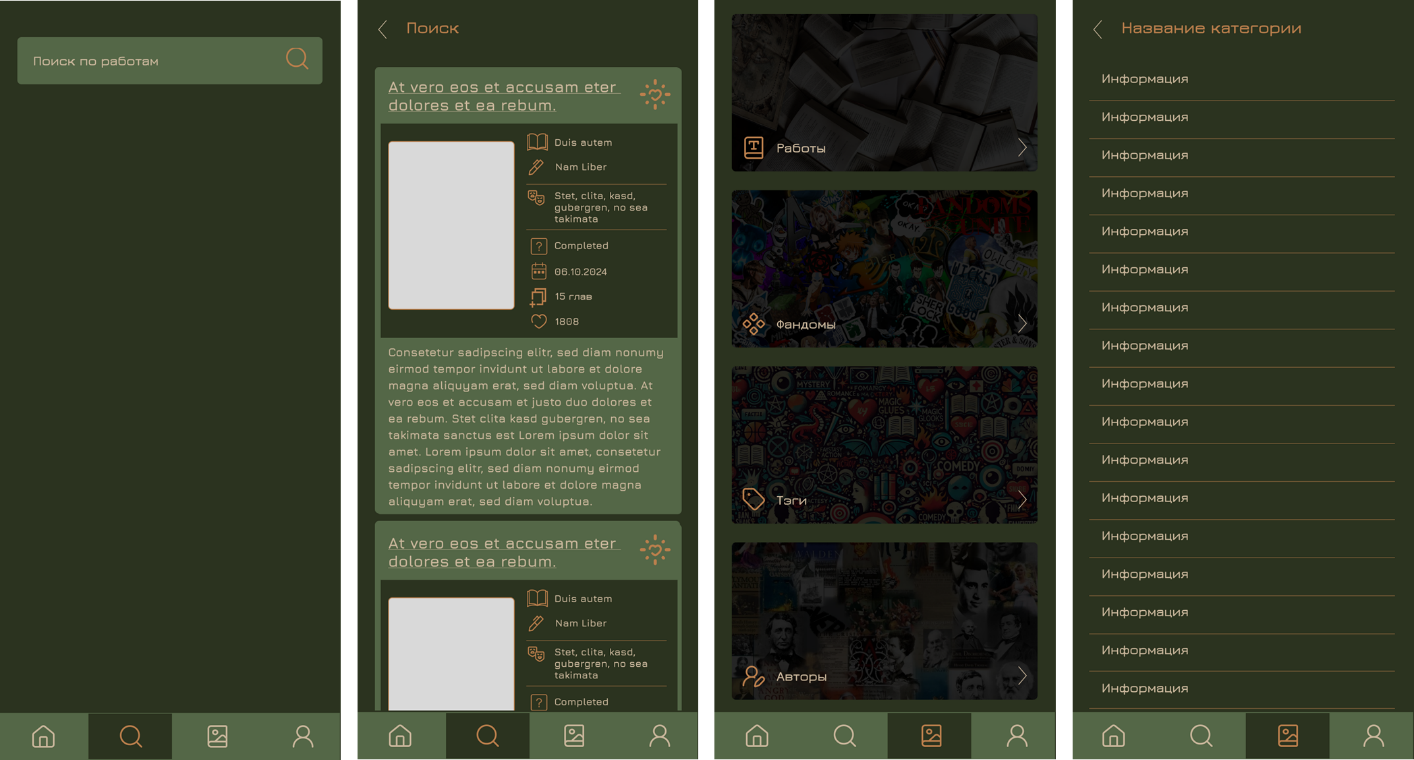


Рисунок 6 – Поиск, результат поиска, каталог, подкатегория

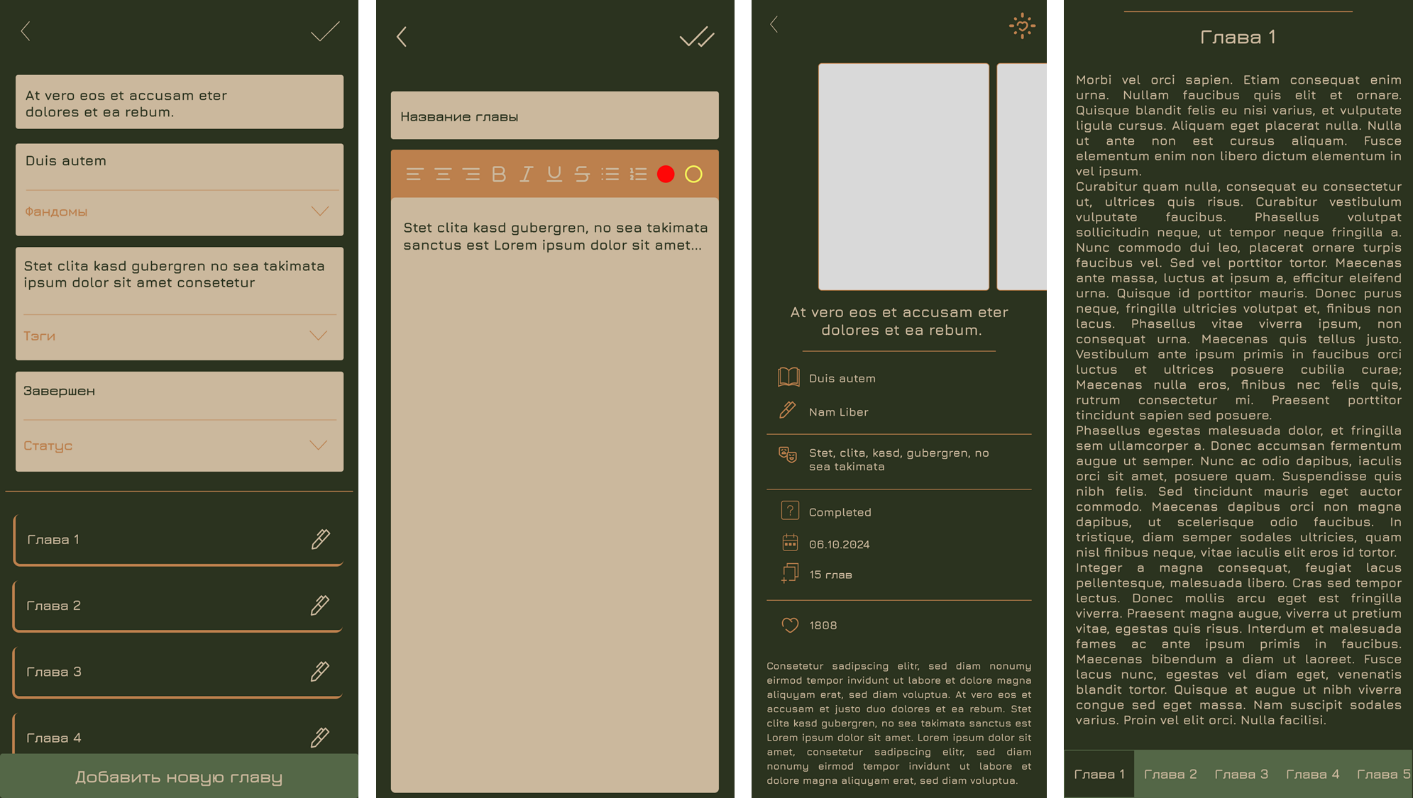


Рисунок 7 – Редактирование работы, редактирование главы, чтение работы

1. РАЗРАБОТКА СЕРВЕРНОЙ ЧАСТИ ПРИЛОЖЕНИЯ
   1. Создание ER-диаграммы

В соответствии с предъявленными ранее требованиями к серверной части приложения была составлена ER-диаграмма будущей базы данных. Рассмотреть ее можно на рисунке 1.

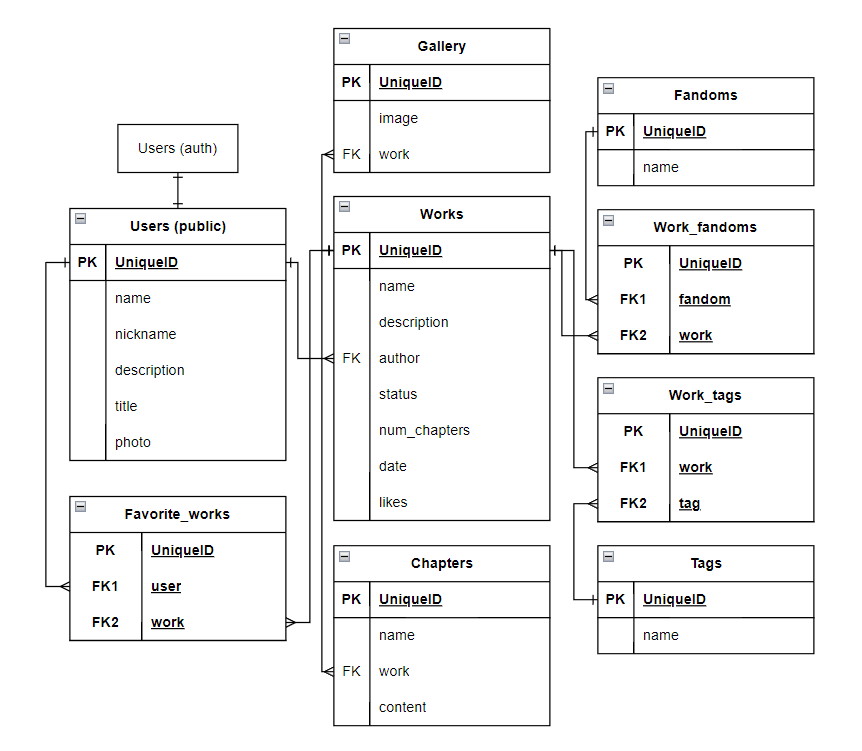


Рисунок 8 – ER-диаграмма БД

* 1. Создание и заполнение таблиц БД

Как говорилось ранее база данных будет развернута в облачном инструментальном сервисе Supabase, предоставляющем удобный интерфейс для создания таблиц реляционной БД, организации связей и заполнения получившегося хранилища комфортным для создателя путем.

Так, необходимые сущности были воссозданы в соответствии с таблицами, отображенными на ER-диаграмме базы данных. Полученная схема представлена на рисунке 2.

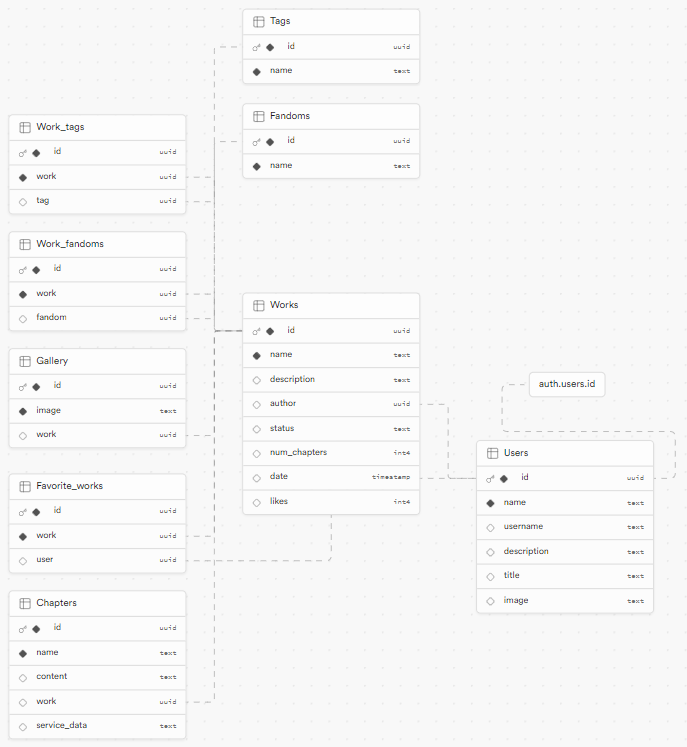


Рисунок 9 – Схема БД

Помимо этого, в большую часть таблиц были помещены тестовые (начальные) наборы данных. Пример таковых с таблицей Works можно увидеть на рисунке 3.

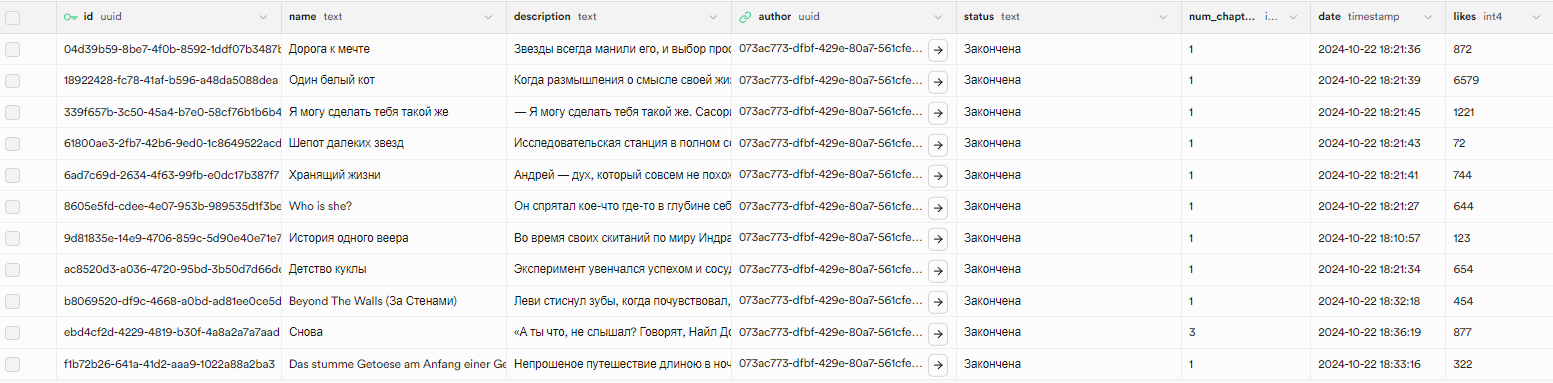


Рисунок 10 – Данные таблицы Works

* 1. Учреждение политик доступа к базе данных

Еще одна особенность сервиса Supabase – возможность простого разграничения прав доступа к таблицам баз данных. Платформа дает возможность создания как своих собственных политик, так и применения типовых, описанных самим Supabase.

Так, к таблицам, данные из которых предстоит лишь считывать и представлять, была применена типовая политика SELECT. Данные таблицы представлены на рисунке 4.

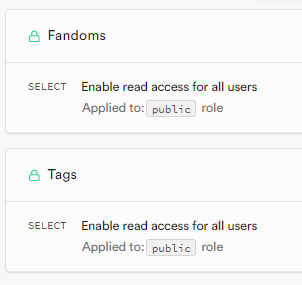


Рисунок 11 – Таблицы политики SELECT

С остальными таблицами несколько сложнее – их данные предстоит представлять, создавать, изменять и удалять, причем специфично по отношению к каждой из таблиц. Для оптимизации дальнейших манипуляций с ними была применена типовая политика ALL, позволяющая выполнять все вышеперечисленные действия. Таблицы, затронутые данным разграничением прав, представлены на рисунке 5.

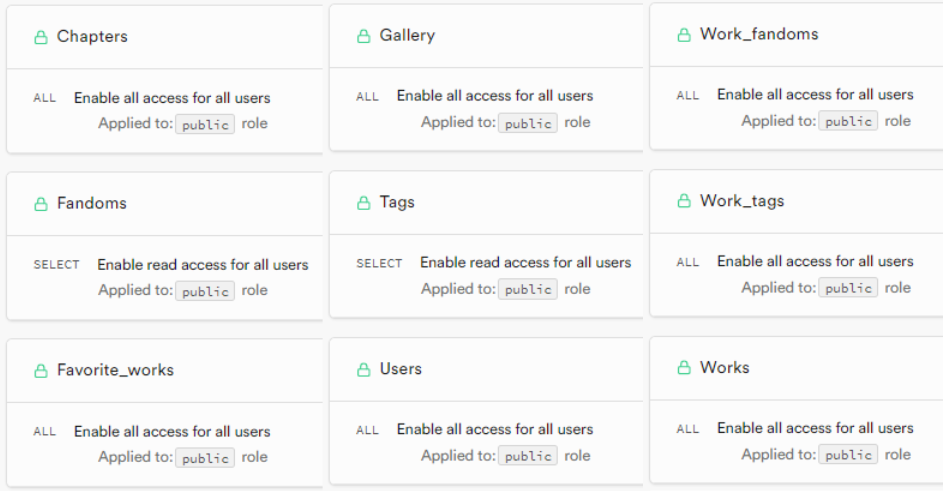


Рисунок 12 – Таблицы политики ALL

В последнюю очередь стоит сказать, что все примененные выше политики распространяются по умолчанию на все общедоступные в сервисе Supabase роли. Авторизованные пользователи, получившие доступ к Приложению, также входят в эту категорию.

1. ТЕСТИРОВАНИЕ СЕРВЕРНОЙ ЧАСТИ ПРИЛОЖЕНИЯ
   1. Тест-план
      1. Перечень работ
         1. Таблицы базы данных – тестирование безопасности и защищенности, производительности, эффективности, стрессов, компонентов, регрессионное, функциональное.
      2. Критерии качества
         1. Безопасность и защищенность – стойкость к внешним и внутренним угрозам, контроль доступа.
         2. Функциональность – выполнение в полной мере всех предполагаемых функций.
         3. Компонентов – структурная целостность и корректность.
         4. Обратная совместимость – стабильная работа «старого» функционала после обновления.
         5. Производительность – скорость выполнения запросов, время выполнения запросов в заданных условиях.
         6. Эффективность – максимальная производительность сервера баз данных при минимальном использовании ресурсов.
         7. Стрессы – исследование работы сервера БД под нагрузкой.
      3. Оценка рисков

Таблица 5 – Оценка рисков

|  |  |  |
| --- | --- | --- |
| Риск | Вероятность появления | Варианты выхода из ситуации |
| Потеря данных | Очень редко | Повышение безопасности и защищенности системы |
| Низкая производительность | Средне | Оптимизация алгоритмов для работы с большим количеством запросов от пользователей |
| Недоступность БД по причине сбоя на платформе | Редко | Реализация механизмов оповещения пользователей о проблемах с соединением |
| Нарушение конфиденциальности данных пользователей | Очень редко | Оптимизация алгоритмов шифрования персональных данных |

* + 1. Документация
       1. Тест-план;
       2. Тестовый набор;
       3. Тест-кейсы.
    2. Стратегия тестирования

Таблица 6 – Стратегия тестирования

|  |  |
| --- | --- |
| Вид теста | Тесты |
| Функциональное тестирование | Возможность получения данных из БД |
| Возможность вставки данных в БД |
| Возможность удаления данных из БД |
| Возможность обновления данных в БД |
| Возможность получения данных по запросу с фильтрацией |
| Интеграционное тестирование | Проверка работоспособности связей между таблицами |
| Проверка реализации каскадного удаления данных |
| Стрессовые тесты | Работа при запросах заданной сложности |
| Работа при большом количестве авторизованных пользователей |
| Работа при большом количестве запросов от пользователей |
| Работа при большом количестве некорректных запросов от пользователей |
| Тестирование производительности | Время отклика при сложных запросах |
| Время отклика при посылании одновременных запросов |
| Тестирование компонентов | Проверка корректности сущностей таблиц БД |
| Проверка корректности связей между сущностями |
| Тестирование безопасности | Проверка разграничения доступа к данным |
| Проверка шифрования персональных данных |
| Регрессионное тестирование | Проверка работоспособности при изменениях в БД |

* + 1. Ресурсы
       1. Человеческие – человек или команда, проводящие тестирование;
       2. Временные – затраченное на все основные и вспомогательные процессы тестирования время;
       3. Программные – программные средства для автоматизации проведения и результатов процесса тестирования;
       4. Аппаратные – основа для работы программных средств.
    2. Метрики
       1. Время отклика – среднее время выполнения запроса по заданным параметрам.
       2. Пропускная способность – количество запросов по заданным параметрам, которые сервер БД способен обработать в единицу времени.
       3. Использование ресурсов – нагрузка на сеть, память, процессор и диск.
    3. Расписание и ключевые точки
       1. Планирование тестирования – 2 ч.
       2. Подготовка тестовых ресурсов – 2 ч.
       3. Выполнение функционального и интеграционного тестирования, тестирования компонентов – 10 ч.
       4. Выполнение стресс тестирования, тестирования безопасности и производительности – 8 ч.
       5. Выполнение регрессионного тестирования – 4 ч.
       6. Анализ результатов и составление отчетности – 6 ч.
  1. Тестовый набор

Таблица 7 – Тестовый набор

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Вид теста | Название теста | Предусловие | Шаги | Ожидаемый результат |
| Функциональное тестирование | Возможность получения данных из БД | Сервер с доступом к базе данных | 1. Составить запрос на выборку данных из таблицы БД 2. Выполнить запрос | Получение всех данных из запрошенной таблицы |
| Возможность вставки данных в БД | 1. Составить запрос на вставку данных в таблицу БД 2. Выполнить запрос | Добавление новых данных в запрошенную таблицу |
| Возможность удаления данных из БД | 1. Составить запрос на удаление данных из таблицы БД 2. Выполнить запрос | Удаление данных из запрошенной таблицы |
| Возможность обновления данных в БД | 1. Составить запрос на редактирование существующих данных в таблице БД 2. Выполнить запрос | Обновление необходимых данных из запрошенной таблицы |
| Возможность получения данных по запросу с фильтрацией | 1. Составить запрос на выборку определенных данных из таблицы БД 2. Выполнить запрос | Получение определенной выборки данных из запрошенной таблицы |
| Интеграционное тестирование | Проверка работоспособности связей между таблицами | 1. Составить запрос на выборку данных из таблицы БД через связь с другой таблицей 2. Выполнить запрос | Получение необходимых данных по связи из запрошенных таблиц |
| Проверка реализации каскадного удаления данных | 1. Составить запрос на удаление данных из таблицы БД, связанной с другой таблицей 2. Выполнить запрос | Удаление необходимых данных из запрошенной таблицы, а также всех связанных с ними данных других таблиц |
| Стрессовые тесты | Работа при запросах заданной сложности | 1. Составить сложный запрос с большим количеством параметров 2. Выполнить запрос | Получение необходимых данных из запрошенной таблицы |
| Работа при большом количестве авторизованных пользователей | 1. Реализовать авторизацию большого числа пользователей | Правильная, безотказная работа приложения |
| Работа при большом количестве запросов от пользователей | 1. Реализовать авторизацию большого числа пользователей 2. Реализовать одновременное большое число запросов к серверу | Правильная, безотказная работа приложения |
| Работа при большом количестве некорректных запросов от пользователей | 1. Реализовать авторизацию большого числа пользователей 2. Реализовать одновременное большое число некорректных запросов к серверу | Правильная, безотказная работа приложения |
| Тестирование производительности | Время отклика при сложных запросах | 1. Составить сложный запрос к таблице БД с большим количеством параметров 2. Выполнить запрос | Корректное установленное среднее время отклика |
| Время отклика при посылании одновременных запросов | 1. Отправить 100 запросов к таблицам БД с регулярными интервалами | Корректное установленное среднее время отклика |
| Тестирование компонентов | Проверка корректности сущностей таблиц БД | 1. Составить запрос, задействующий все существующие поля сущностей БД 2. Выполнить запрос | Корректное представление и содержание всех сущностей БД |
| Проверка корректности связей между сущностями | 1. Составить запрос, задействующий все существующие связи между сущностями БД 2. Выполнить запрос | Корректное выполнение запроса с определенными заранее представлениями данных |
| Тестирование безопасности | Проверка разграничения доступа к данным | 1. Составить запрос на получение данных из БД без наличия необходимых прав 2. Выполнить запрос | Ограничение в выполнении запроса из-за отсутствия надлежащих прав |
| Проверка шифрования персональных данных | 1. Произвести регистрацию нового пользователя 2. Составить запрос на определение типа аутентификационных данных | Данные для авторизации хранятся в зашифрованном виде |
| Регрессионное тестирование | Проверка работоспособности при изменениях в БД | 1. Произвести обновление сервера БД 2. Воспроизвести механизм авторизации | Успешная реализация механизма авторизации |

* 1. Тест-кейсы

Таблица 8 – Тест-кейсы

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| № | Название | Предусловие | Шаги | Ожидаемый результат |
| 1 | Получение данных из БД | Сервер с доступом к базе данных | 1. Ввод запроса «select \* from "Works"» 2. Выполнение запроса | Вывод всех данных из таблицы Works |
| 2 | Возможность вставки данных в БД | 1. Ввод запроса «insert into "Tags" (name) values ('Emotional hurt/comfort')» 2. Выполнение запроса | Успешная вставка данных в таблицу Tags |
| 3 | Возможность удаления данных из БД | 1. Ввод запроса «delete from "Tags" where name = 'Emotional hurt/comfort'» 2. Выполнение запроса | Успешное удаление запрошенной строки из таблицы Tags |
| 4 | Возможность обновления данных в БД | 1. Ввод запроса «update "Users" set title = 'Новичок' where username = 'y\_k\_e\_s'» 2. Выполнение запроса | Успешное обновление данных в запрошенной строке таблицы Users |
| 5 | Возможность получения данных по запросу с фильтрацией | 1. Ввод запроса «select \* from "Works" where num\_chapters > 1» 2. Выполнение запроса | Успешный поиск и отображение запрошенных данных из таблицы Works |
| 6 | Проверка работоспособности связей между таблицами | 1. Ввод запроса «select \*   from "Works"  join "Work\_fandoms" on "Works".id = "Work\_fandoms".work  join "Fandoms" on "Fandoms".id = "Work\_fandoms".fandom»   1. Выполнение запроса | Успешное соединение и отображение данных из запрошенных таблиц |
| 7 | Проверка реализации каскадного удаления данных | 1. Ввод запроса «delete from "Works" where name = 'Снова'» 2. Выполнение запроса | Успешное удаление запрошенных данных из таблицы Works, а также из всех связанных таблиц – Work\_tags, Work\_fandoms, Chapters, Gallery, Favorite\_works |
| 8 | Работа при большом количестве авторизованных пользователей | 1. Произвести авторизацию 100 пользователей 2. Выполнить запрос «select \* from "Works"» | Успешное представление запрошенных данных для всех пользователей |
| 9 | Работа при большом количестве запросов от пользователей | 1. Произвести авторизацию 100 пользователей 2. Выполнить 100 запросов от каждого пользователя | Успешное представление всех запрошенных данных для всех пользователей |
| 10 | Работа при большом количестве некорректных запросов от пользователей | 1. Произвести авторизацию 100 пользователей 2. Выполнить 100 запросов с ошибками от каждого пользователя | Успешная обработка каждой ошибки в каждом запросе от каждого пользователя, вывод сообщений об ошибках |
| 11 | Время отклика при сложных запросах | 1. Ввести сложный запрос «select "Works".name as work, "Users".name from "Works"   join "Users" on "Works".author = "Users".id  where "Works".name = 'Детство куклы'»   1. Выполнить запрос | Среднее время отклика не превышает 200 мс |
| 12 | Время отклика при посылании одновременных запросов | 1. Составить 1000 запросов 2. Выполнить запросы одновременно | Среднее время отклика не превышает 500 мс |
| 13 | Проверка корректности сущностей таблиц БД | 1. Выполнить select запрос для всех таблиц БД 2. Проверить корректность типов полей и значений | Таблицы имеют соответствующие документации типы данных и значений, значения корректны |
| 14 | Проверка корректности связей между сущностями | 1. Ввод запроса «select \*   from "Works"  join "Work\_tags" on "Works".id = "Work\_tags".work  join "Tags" on "Tags".id = "Work\_tags".tag»   1. Выполнение запроса | Успешное соединение и отображение данных из запрошенных таблиц |
| 15 | Проверка шифрования персональных данных | 1. Ввод запроса «select auth.users.encrypted\_password from auth.users» 2. Выполнение запроса | Тип полученных паролей - зашифрованный |
| 16 | Проверка работоспособности при изменениях в БД | 1. Добавить новый столбец «description» в таблицу «Tags» 2. Выполнить запросы select, insert, delete и update к таблице Tags | Успешное, безошибочное выполнение всех необходимых запросов |

1. РАЗРАБОТКА МОБИЛЬНОГО ПРИЛОЖЕНИЯ
   1. Подключение шрифтов и настройка ранее выбранных цветов

Одной из самых важных составляющих мобильного приложения в принципе является приятный глазу пользователя графический интерфейс, благодаря которому пользователь гораздо в большей мере заинтересовывается в использовании функционала. Грамотная цветовая схема и красивый, но доступный шрифт – один из «кирпичиков», строящих заинтересованность пользователя.

В соответствие с составленным выше макетом в приложении должен применяться шрифт «Jura», являющийся частью Google Fonts.

Чтобы подключить необходимый нам шрифт к приложению на Jetpack Compose, первым делом необходимо подключить соответствующую библиотеку (таблица 9).

Таблица 9 – Библиотека для подключения Google Fonts

|  |
| --- |
| *implementation*(*libs*.*androidx*.*ui*.*text*.*google*.*fonts*) |

После этого в системной папке проекта ui.theme в файле Type необходимо создать Google Font провайдера, имя шрифта и в конце концов собственную Font Family. Как именно это сделать, можно увидеть в репозитории по ссылке[[1]](#footnote-1). После всех вышеперечисленных действий необходимый нам шрифт можно привязать к тексту привычным нам образом.

Касаемо шрифтов, Jetpack Compose «из коробки» предоставляет некоторый набор цветов (цветовую тему), которую можно редактировать под потребности своего приложения. Такая ситуация характерна и для нашего проекта – у нас есть 4 основных цвета (HEX #546747, #BC804D, #CBB89D, #2B331F), которые мы просто прописываем в системном файле с цветами и подключаем в системном файле с цветовыми темами. Код подключения цветов также можно увидеть в файлах Type и Color папки ui по ссылке ниже.

* 1. Верстка макета

В соответствии с макетом были сверстаны вводной экран приложения и страница с популярными работами (рисунок 13); страница поиска по работам и экран с результатами поиска (рисунок 14); раздел с каталогом и подразделы для каждой из категорий меню (рисунок 15); профиль пользователя, редактирования профиля, экраны с любимыми работами и работами авторства пользователя (рисунок 16); страницы чтения и редактирования работы (рисунок 17); и, естественно, экраны авторизации и регистрации (рисунок 18).



Рисунок 13 – Экран интро и главная страница

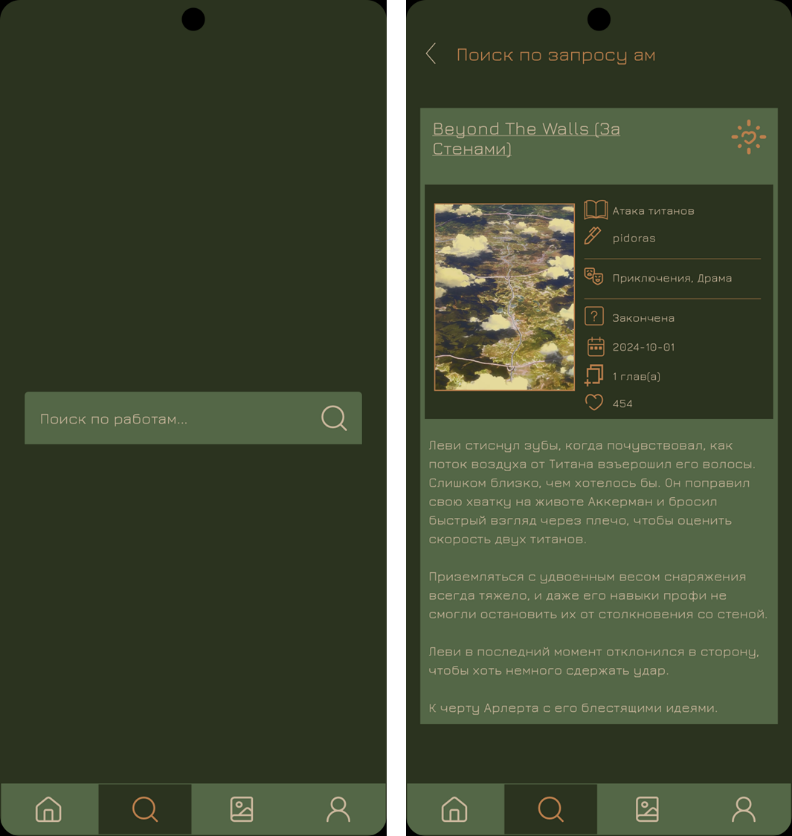


Рисунок 14 – Поиск и результат поиска

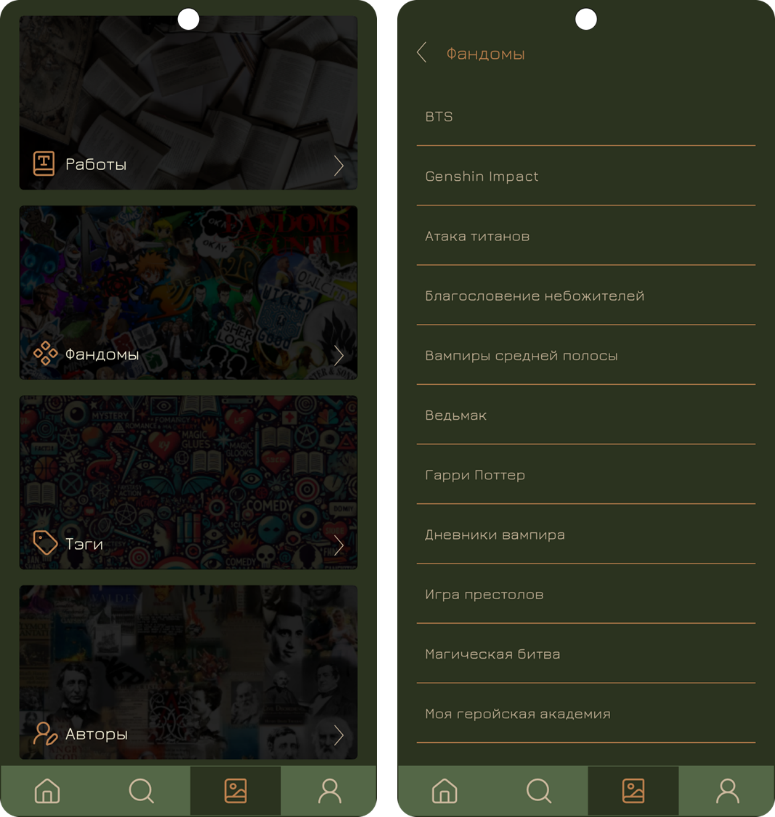


Рисунок 15 – Каталог и подкатегория каталога

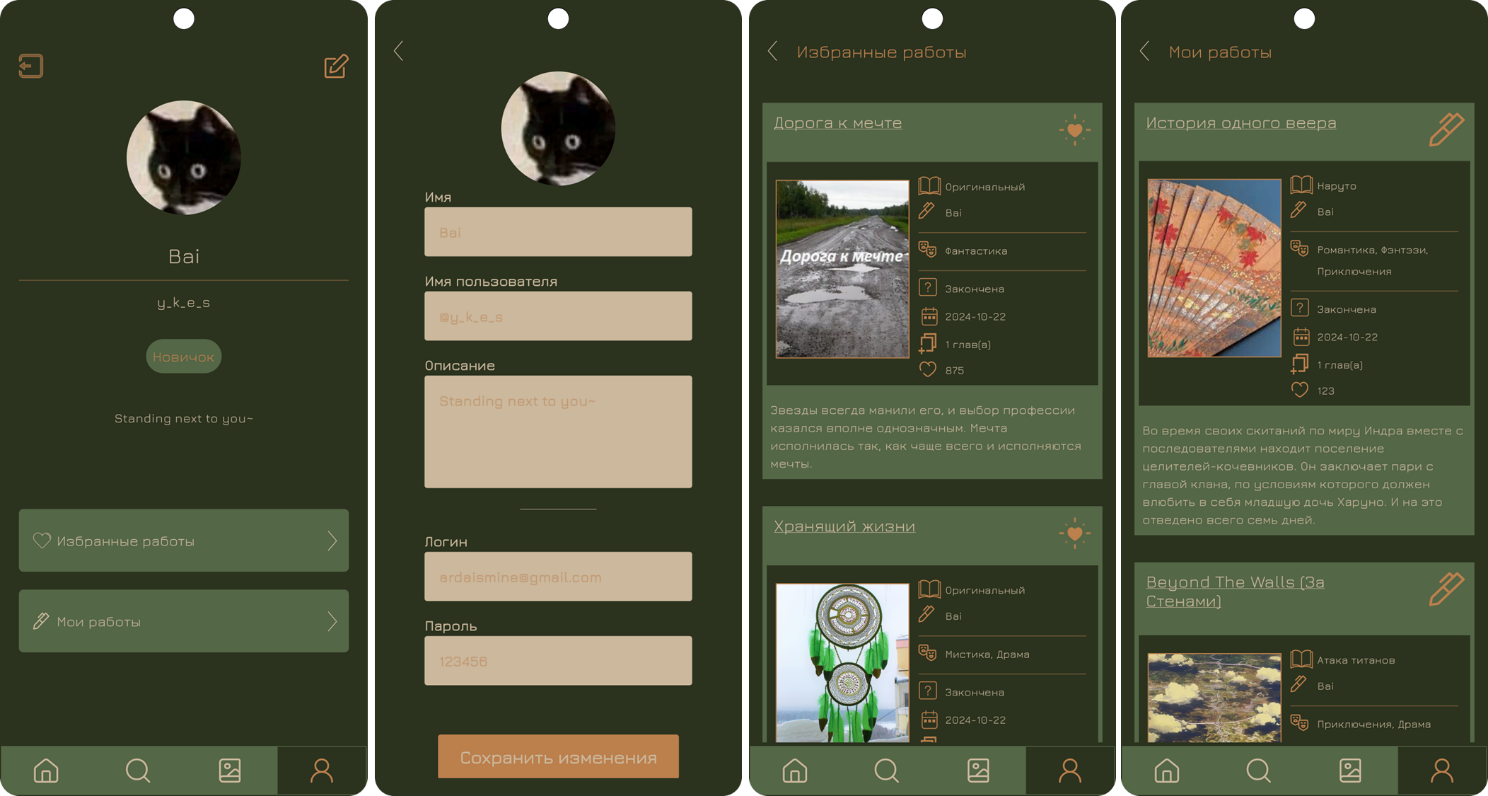


Рисунок 16 – Профиль пользователя, редактирование, избранные работы и работы пользователя

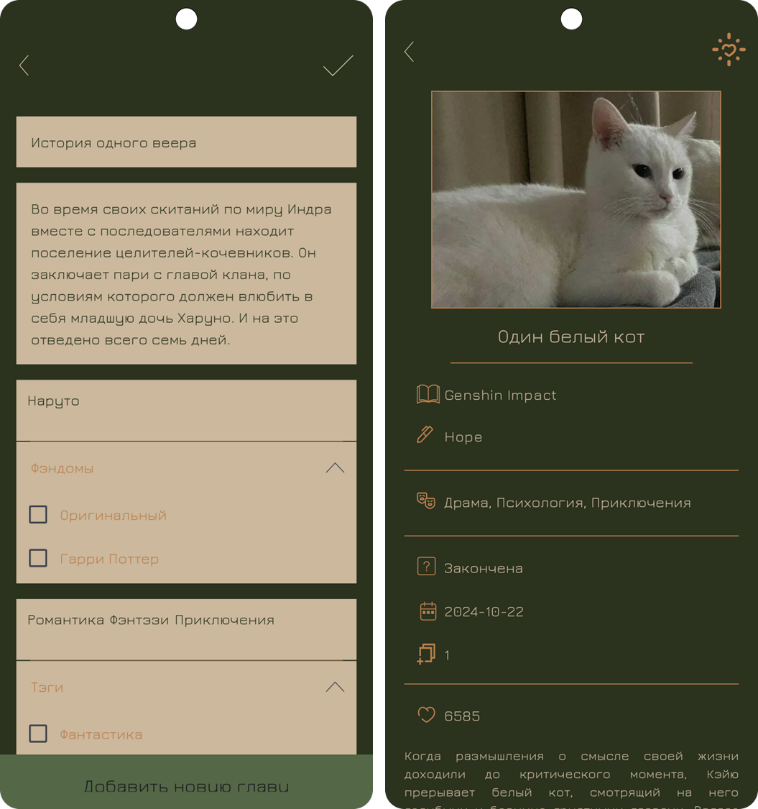


Рисунок 17 – Редактирование работы и чтение работы



Рисунок 18 – Экраны авторизации и регистрации

Каждый из экранов был разработан с учетом требуемой адаптивности приложения, из-за чего размеры и положение элементов на экране преимущественно не являются фиксированными. Для большинства страниц присутствует возможность прокрутки, в том числе с учетом системных функций телефона.

Помимо вышеперечисленного, для приложения была организована собственная иконка (логотип), отражающая тематику проекта.

* 1. Подключение библиотек и интернета

Так как приложение активно взаимодействует с базой данных, наличие на носителе интернета, безусловно, является фундаментом для дальнейших действий.

Для обеспечения доступа приложения к интернету в основополагающем системном файле AndroidManifest была прописана команда (таблица 10), по умолчанию при установке приложения запрашивающая и получающая соединение с сетью.

Таблица 10 – Команда получения доступа к интернету

|  |
| --- |
| <uses-permission android:name="android.permission.INTERNET" /> |

Помимо этого, для обеспечения корректной работы и отображения всех элементов и функций в системном файле build.gradle.kts сборки были подключены некоторые библиотеки и плагины, код которых можно найти в репозитории по ссылке[[2]](#footnote-2).

* 1. Подключение к серверной части приложения

Следующим этапом являлось создание подключения к базе данных и моделей таблиц.

Первое было обеспечено благодаря созданию статического подключения к базе данных посредством специального метода Supabase по созданию клиента createSupabaseClient(). В метод были переданы реальные URL адрес базы данных и API Key для подключения к ней. На завершающем этапе к созданному нами клиенту Supabase устанавливались необходимые расширения для работы с БД. В нашем случае были установлены Auth для реализации внутреннего механизма аутентификации Supabase, Postgrest собственно для манипуляций с базой данных и Storage для работы с внутренним хранилищем Supabase.

Далее необходимо было создать модели (Data Class) существующих в базе данных таблиц для получения и идентификации данных из нее. Стоит упомянуть, что, помимо полей, отражающих в точности реально существующие колонки таблиц БД для некоторых из них были созданы аналоги так называемых виртуальных полей в Avalonia .NET, содержащие в себе списки с данными, полученными по внешним ключам данной сущности. Так, например, для каждой из работ, полученной из базы данных, создавался список всех тэгов, присвоенных данной работе, полученных по внешнему ключу из смежной таблицы с тэгами. Код создания всех моделей сущностей размещен в папке Models репозитории по ссылке[[3]](#footnote-3).

* 1. Навигация по страницам

Так как разрабатываемое приложение – многостраничное, то наличие навигации между ними определяет работоспособность проекта в целом.

Все события происходят в функции Navigation, содержащей в себе специальный объект NavHost, строящий навигационный граф между прописанными в нем Composable функциями. Код данного метода также можно найти в репозитории по ссылке ниже.

* 1. Разработка вспомогательных методов и методов для обработки данных с сервера
     1. Сохранение пользовательской сессии – самый фундаментальный метод приложения, вызывающийся на этапе загрузки («Интро»). Наличие сессии определялось по наличию почты пользователя, сохраненной при помощи SharedPreferences и полученной getString() метода getSharedPreferences, на основе же полученных данных определялась дальнейшая навигация по приложению.
     2. Методы авторизации и регистрации пользователя – функции, реализующие подключаемые ранее внутренние механизмы аутентификации Supabase. На основе передаваемых аргументов методы onSignInEmailPassword и onSignUpEmail делают запрос к Supabase и обрабатывают успешность ответа.
     3. Метод получения работ – один из самых объемных и значимых методов в приложении, получающий самые многочисленные данные проекта: работы. При помощи запросов к Supabase (опять же, возможных благодаря расширениям) данные из всех таблиц базы данных получаются в первоначальные листы, после чего обрабатываются, надлежащим образом заполняя все существующие и «виртуальные» поля каждой из работ из листа работ.
     4. Метод добавления лайка – метод, реализующий как изменения в интерфейс при проставлении лайка, так и добавление соответствующей работы в избранное для текущего пользователя. Метод манипулирует состоянием иконки лайка, после чего вызывает хранящуюся в Supabase собственную функцию увеличения числа лайков, а также добавления работы в таблицу с избранными на текущем читателе.

1. ТЕСТИРОВАНИЕ МОБИЛЬНОГО ПРИЛОЖЕНИЯ
   1. Тест-план
      1. Перечень работ
         1. Приложение – функциональное, UI/UX тестирование, тестирование производительности, безопасности, стрессов, компонентов, регрессионное.
      2. Критерии качества
         1. Функциональность – все заявленные функции выполняются корректно.
         2. Удобство использования – гармоничное расположение и вид графических элементов, простая навигация, отсутствие сложных механизмов взаимодействия.
         3. Производительность – адекватное время отклика интерфейса (не превышающее 2 секунд) на всех экранах.
         4. Безопасность – стойкость к угрозам, контроль доступа, защита данных от утечки.
         5. Надежность – приложение не вылетает при интенсивной нагрузке.
         6. Компонентов – структурная целостность и корректность.
         7. Обратная совместимость – стабильная работа «старого» функционала после обновления.
      3. Оценка рисков

Таблица 11 – Оценка рисков

|  |  |  |
| --- | --- | --- |
| Риск | Вероятность появления | Варианты выхода из ситуации |
| Потеря данных в процессе работы приложения | Средняя | Регулярное тестирование с реальной базой данных |
| Некорректное отображение контента на разных устройствах | Высокая | Тестирование на устройствах с различными разрешениями экранов |
| Высокая нагрузка вызывает сбой приложения | Низкая | Нагрузочное тестирование, оптимизация запросов к серверу |
| Утечка пользовательских данных | Средняя | Внедрение методов шифрования, проверка уязвимостей |
| Проблемы с Supabase | Низкая | Тестирование соединения с Supabase, оптимизация тарифа |

* + 1. Документация
       1. Тест-план;
       2. Тестовый набор;
       3. Тест-кейсы.
    2. Стратегия тестирования

Таблица 12 – Стратегия тестирования

|  |  |
| --- | --- |
| Вид теста | Тесты |
| Функциональное тестирование | Авторизация через email и пароль |
| Отображение списка работ |
| Добавление работы в избранное |
| Добавление работы |
| Просмотр глав и переключение между ними |
| UI/UX тестирование | Корректное отображение текста на страницах |
| Верная цветовая схема, шрифты |
| Кнопки нажимаются и ведут к правильным действиям |
| Стрессовые тесты | Создание 1000 пользователей одновременно |
| Проверка работы приложения при загрузке списка из 10 000 фанфиков |
| Тестирование производительности | Время отклика основных интерфейсов |
| Время отклика побочных интерфейсов |
| Тестирование компонентов | Наличие всех заявленных элементов |
| Наличие всех заявленных связей между элементами |
| Тестирование безопасности | Наличие доступа к профилю без авторизации |
| Регрессионное тестирование | Воспроизведение основного сценария взаимодействия с приложением после изменений в коде |

* + 1. Ресурсы
       1. Человеческие – человек или команда, проводящие тестирование;
       2. Временные – затраченное на все основные и вспомогательные процессы тестирования время;
       3. Программные – программные средства для автоматизации проведения и результатов процесса тестирования;
       4. Аппаратные – основа для работы программных средств.
    2. Метрики
       1. Время отклика – среднее время отклика интерфейса.
       2. Использование ресурсов – нагрузка на сеть, память, процессор и диск.
       3. Количество найденных дефектов.
       4. Покрытие тестами.
    3. Расписание и ключевые точки
       1. Планирование тестирования – 2 ч.
       2. Подготовка тестовых ресурсов – 2 ч.
       3. Выполнение функционального и UI/UX тестирования, тестирования компонентов – 10 ч.
       4. Выполнение стресс тестирования, тестирования безопасности и производительности – 8 ч.
       5. Выполнение регрессионного тестирования – 4 ч.
       6. Анализ результатов и составление отчетности – 6 ч.
  1. Тестовый набор

Таблица 13 – Тестовый набор

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Вид теста | Название теста | Предусловие | Шаги | Ожидаемый результат |
| Функциональное тестирование | Авторизация через email и пароль | Функционирующее приложение | 1. Ввести логин [ardaismine@gmail.com](mailto:ardaismine@gmail.com) и пароль 111111 в поля ввода 2. Нажать на кнопку авторизации | Переход на главную страницу |
| Отображение списка работ | 1. Ввод данных для авторизации 2. Нажатие кнопки авторизации 3. Переход на главную страницу | Отображение списка из 5 самых популярных работ |
| Добавление работы в избранное | Функционирующее приложение с аутентифицированным пользователем и открытой главной страницей | 1. Нажатие на кнопку лайка 2. Переход на страницу с профилем пользователя 3. Переход на страницу с любимыми работами пользователя | Количество лайков на избранной работе увеличилось на 1, в любимых работах пользователя появилась избранная ранее работа |
| Добавление работы | 1. Переход на страницу с профилем пользователя 2. Переход на страницу с работами пользователя 3. Нажатие на кнопку Создать 4. Заполнение одного и более полей формы 5. Нажатие на кнопку Сохранить | Переход на экран с работами пользователя, среди которых только что добавленная. |
| Просмотр глав и переключение между ними | 1. Нажатие на работу 2. Нажатие на кнопку следующей главы | Открытие первой главы работы и ее текста. По нажатию на главу переход на требуемую главу работы |
| UI/UX тестирование | Корректное отображение текста на страницах | 1. Проверка гармоничности расположения элементов и текстов на странице | Элементы и текста расположены на странице согласно макету |
| Верная цветовая схема, шрифты | 1. Сверка используемых на главной странице цветов 2. Сверка используемого шрифта | Цвета и шрифт использованы в соответствии с макетом |
| Кнопки нажимаются и ведут к правильным действиям | 1. Нажатие на работу 2. Нажатие на кнопку Назад | Последовательный переход на первую главу выбранной работы и обратно на главную страницу |
| Стрессовые тесты | Создание 1000 пользователей одновременно | 1. Обеспечение одновременной регистрации 1000 пользователей | Бесперебойная работа приложения со средним временем отклика |
| Проверка работы приложения при загрузке списка из 10 000 работ | 1. Переход на страницу каталога, выбор категории «Работы» | Бесперебойная работа приложения со средним временем отклика |
| Тестирование производительности | Время отклика основных интерфейсов | 1. Переход на страницу поиска | Время перехода на страницу не превышает 2 секунд |
| Время отклика побочных интерфейсов | 1. Переход на страницу профиля пользователя 2. Переход на страницу с работами пользователя 3. Нажатие кнопки Создать 4. Выбор значения из списка тэгов на экране добавления работы | Время отклика списка не превышает 2 секунд |
| Тестирование компонентов | Наличие всех заявленных элементов | 1. Последовательный переход по всем имеющимся экранам приложения | Все заявленные экраны существуют |
| Наличие всех заявленных связей между элементами | 1. Последовательный переход по всем имеющимся экранам приложения | Все существующие экраны связаны и дают возможность переходов |
| Тестирование безопасности | Наличие доступа к профилю без авторизации | Функционирующее приложение | 1. Оставление пустых значений в полях авторизации 2. Нажатие на кнопку Авторизация 3. Аварийное завершение работы приложения 4. Запуск приложения 5. Переход на страницу профиля | Уведомление о некорректных входных данных и выход из некорректного профиля на экран авторизации |
| Регрессионное тестирование | Воспроизведение основного сценария взаимодействия с приложением после изменений в коде | 1. Внедрение нового модуля в приложение 2. Воспроизведение всех основных переходов между страницами 3. Воспроизведение всех основных механизмов работы доступных кнопок | Бесперебойная работа приложения со средним временем отклика |

* 1. Юнит-тесты

Помимо вышеперечисленных ручных тестов, средствами Jetpack Compose были выполнены 5 юнит-тестов (код которых можно найти в репозитории по ссылке[[4]](#footnote-4)), механически проверяющих следующие аспекты:

* Тест на правильность установки состояния Success при успешном входе в систему.
* Тест на правильность установки состояния Success при успешной регистрации.
* Тест на проверку правильности установки состояния Error после неудачной попытки входа в систему.
* Тест на проверку правильности установки состояния Error после неудачной попытки регистрации.
* Тест на проверку корректности общего объема полученных данных.

ЗАКЛЮЧЕНИЕ

В ходе разработки приложения был достигнут ряд целей, направленных на создание удобной платформы для взаимодействия пользователей с фанатским контентом. Разработка мобильного приложения предоставила пользователям интуитивно понятный интерфейс и быстрый доступ для взаимодействия с репозиторием.

Проведённые тестирования подтвердили корректность работы приложения, его соответствие поставленным задачам и удобство использования. Успешная реализация проекта показала, что интеграция современных технологий и грамотный подход к проектированию позволяют эффективно решать задачи, направленные на самые разнообразные цели.

Результатом работы стало приложение, которое объединяет авторов и читателей, предоставляя каждому пользователю удобный инструмент для публикации и поиска фанатских произведений. В дальнейшем возможна доработка и масштабирование функционала для ещё большего удобства и привлечения широкой аудитории.

1. http://gogs.ngknn.local:3000/bai/Warehouse.git [↑](#footnote-ref-1)
2. http://gogs.ngknn.local:3000/bai/Warehouse.git [↑](#footnote-ref-2)
3. http://gogs.ngknn.local:3000/bai/Warehouse.git [↑](#footnote-ref-3)
4. http://gogs.ngknn.local:3000/bai/Warehouse.git [↑](#footnote-ref-4)