1. РАЗРАБОТКА ИНФОРМАЦИОННОЙ СИСТЕМЫ
   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.

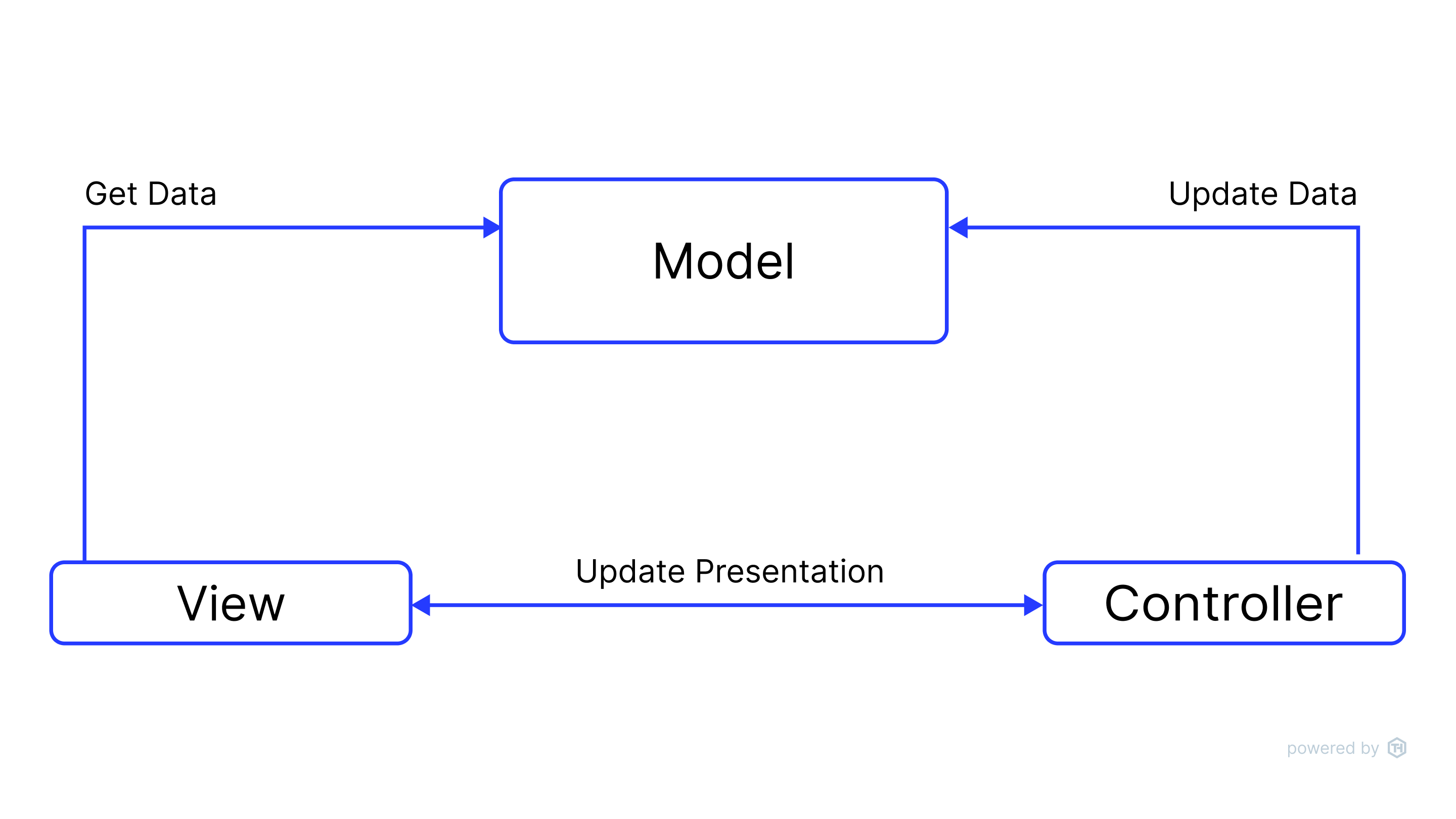


Рисунок 1 – MVC

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.

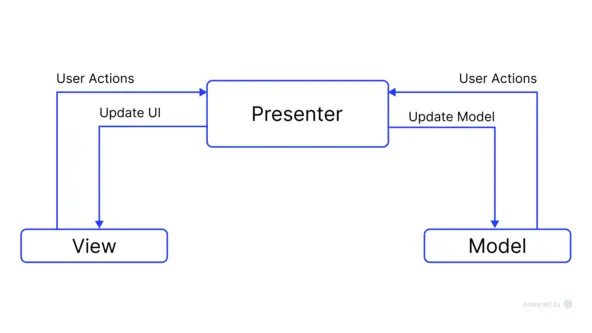


Рисунок 2 – MVP

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.

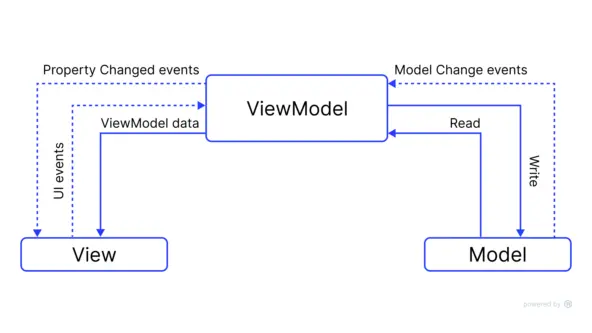


Рисунок 3 – MVVM

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.