**Введение**

На проект по технологической практике была поставлена задача разработка мобильного приложения для воспроизведения музыки «4Music».

Цель задачи заключается в реализации приложения в среде разработки Android Studio с помощью языка программирования Kotlin.

Первый раздел носит название «Структура и общая технологическая схема производства, характеристики основных видов продукции». Этот раздел позволяет понять, как организовано производство, какие продукты выпускаются и какие требования предъявляются к качеству и безопасности продукции. Это помогает обеспечить прозрачность и открытость информации о предприятии, что важно для потенциальных инвесторов, партнеров и потребителей. Второй раздел носит название

Второй раздел «Описание структуры ВЦ, АСУП и производственного отдела, где проходит практика». Этот раздел технической документации описывает структуру ВЦ, АСУП и производственного отдела. В нем указывается общая структура организации, функции и обязанности каждого отдела, их взаимодействие друг с другом, штатное расписание, процессы, оборудование и программное обеспечение, требования к квалификации персонала и действующие процедуры и регламенты.

В третьем разделе «Должностные обязанности оператора ЭВМ, техника-программиста, инженера-программиста» описаны должностные обязанности специалистов в области информационных технологий: оператора ЭВМ, техника-программиста и инженера-программиста. Для каждого из указанных специалистов перечислены основные задачи, которые могут включать ввод и обработку данных, обеспечение сохранности и резервного копирования, поддержку пользователей, разработку и тестирование программного обеспечения, руководство командой разработчиков и другие.

В четвёртом разделе «Анализ задачи», описан процесс изучения исходных данных и требований к проекту, а также выявления возможных проблем и рисков, так же в данном разделе описаны инструменты разработки, диаграмма вариантов использования, модель данных и постановка задачи.

Пятый раздел «Проектирование задачи» в данном разделе описывается и анализируется проблема или задача, которую необходимо решить в рамках проекта. Здесь определяются цели, требования и ограничения проекта, а также выбираются оптимальные методы и подходы для его выполнения.

Шестой раздел «Реализация» технической документации описывает процесс создания продукта, указывает используемые технологии и методы, а также отслеживает изменения и проблемы в ходе разработки.

Седьмой раздел «Тестирование», описывающий полное и функциональное тестирование программы. В нем тестируется каждый пункт меню и операции, выполняемые приложением. Все возможные действия пользователя при работе с программой моделируются, начиная от запуска и до выхода.

Восьмой раздел «Применение» технической документации, который описывает, как и где использовать продукт или услугу, а также какие меры предосторожности следует соблюдать при их использовании.

В «Списке используемых источников» приведен список используемых источников при разработке программного продукта.

В приложении А к пояснительной записке представлен листинг мобильного приложения.

**1 Структура и общая технологическая схема производства, характеристики основных видов продукции**

Штаб сотрудников состоит из четырёх отделов: фронтенд, бэкенд и отдел по проведению тестирования продукта. Так же на предприятии имеется тим-лидер. Тим- лидер не является отделом.

Должностные обязанности тим-лидера на предприятии заключается в проверки выполнения работ, слежение за качеством работ, выдача поручений и задний для бэкенд и фронтенд отделов. Так же в его задачи входят проведение аудио конференций с заказчиками и уточнение деталей проекта. Так же тим-лидер занимает должность генерального директора компании.

Бэкенд отдел состоит из девяти человек. В должностные обязанности данного отдела входят: создание функциональной части проектов, поддерживание старых проектов компании, доработка существующих проектов.

Фронтенд отдел состоит из одного человека. В его должностные обязанности входят: создание визуальной клиентской части, создание визуальной функциональной составляющей, связывание визуальной составляющей и функциональной части, создание и доработка уже существующих дизайнов.

Отдел по проведению тестирования программного продукта состоит из двух сотрудников. В их должностные обязанности входит проведение тестирования готовых, тестирование продуктов в которые внесли изменения и создание отчётов по проведению тестирования программных продуктов.

**2 Описание структуры ВЦ, АСУП и производственного отдела**

ВЦ (вычислительный центр) представляет собой организацию, которая обладает специальным оборудованием и программными средствами для обработки и хранения данных. ВЦ обеспечивает комплексное решение задач информационных технологий и предоставляет услуги по обработке данных для внутренних или внешних пользователей.

Вычислительный центр данной компании отсутствует. На данном производстве все работники компании приходят со своим оборудованием для реализации проекта. Хранение данных происходит в облачном формате с помощью системы GIT.

АСУП (автоматизированная система управления производством) – это система, основанная на применении компьютерных технологий для управления производственными процессами. Она позволяет автоматизировать и контролировать различные аспекты производства, такие как планирование производства, управление запасами, контроль качества, управление энергоресурсами и другие.

Автоматизированная система управления производством происходит так же в облачном формате, так все данные хранятся на серверах GIT. Контролирование и управление производством происходит с помощью системы GIT.

Производственный отдел производства состоит из бэкенд, фронтенд частей и тестирования программного продукта. Фронтенд отвечает за визуализацию и реализацию фронтовой частей сайта. Бэкенд часть занимаются созданием базы данных и функциональной части проектов. Тестирующий отдел занимается поиском ошибок и некоренной работы продукта для дальнейшего исправления ошибок.

**3 Должностные обязанности оператора ЭВМ, техника-программиста, инженера-программиста**

В должностные обязанности фронтенд разработчика входит: создание визуальной составляющей проекта, создание логики с бэкенд частью и создание дизайна по запросам заказчика.

В обязанности бэкенд разработчика входит: создание функциональной части разрабатываемого продукта и поддержка существующих проектов.

В обязанности тестировщика входит: проведение тестирования готового продукта для выявления ошибок работы продукта.

В обязанности тим-лидера входит: взаимодействие с клиентами, выдача заданий бэкенд, фронтенд разработчикам и контроль за выполнением работы.

**4 Анализ задачи**

**4.1 Анализ предметной области**

В понятие разработки мобильных приложений для смартфонов, планшетов и прочих мобильных устройств входит написание программного кода с целью создания программ, которые будут работать на определенных мобильных платформах (на сегодняшний день существует 2 основные платформы мобильных операционных систем — Android и iOS, и менее популярные Windows Phone и Symbian). Эти программы и приложения могут предварительно устанавливаться на мобильные телефоны, персональные цифровые помощники, корпоративные цифровые ассистенты, смартфоны и прочие мобильные устройства до того, как устройства попадут в руки пользователю, либо загружаться пользователями в устройство непосредственно в процессе использования.

«4Music» — это мобильное приложение для Android, которое выполняет роль музыкальной площадки. В данном приложении пользователи могут находить различными способами и прослушивать все песни мира, добавлять их в избранное, тем самым создавая собственный плейлист, а так же пользователь имеет возможность оценивать песни по музыкальной системе оценивания проекта “Риса за творчество” .

Пользоваться приложением представляется возможным сразу после установки приложения.

**4.2 Инструменты разработки**

Для реализации проекта будет выбран фреймворк Android Studio. Так же будут использованы другие инструменты разработки:

* Firebase - это облачная база данных, которая позволяет пользователям хранить и получать сохраненную информацию, а также имеет удобные средства и методы взаимодействия с ней;
* Microsoft Word – программа, используемая для создания документации к программному продукту.

Для оптимального функционирования разрабатываемого приложния необходимо наличие у телефонов и планшетов следующих параметров:

* 1 GB RAM;
* android 6.0 и выше;
* стабильный интернет.

**4.3 Диаграмма вариантов использования**

Диаграмма вариантов использования – это инструмент UML для описания функциональных требований системы и ее взаимодействия с внешними сущностями (актёрами) в возможных сценариях использования.

Диаграмма вариантов использования используется для определения функциональных требований к системе, которые должны быть реализованы. Диаграмма описывает варианты использования системы, которые могут возникнуть при взаимодействии пользователей с ней. В центре диаграммы находится система, а вокруг нее располагаются актеры – пользователи, которые взаимодействуют с системой. Для каждого актера определяются варианты использования системы, в соответствии с его потребностями.

Актёр — это любой человек, группа людей или внешняя система, которые участвуют в определенном процессе или взаимодействии с системой (Рисунок 1).

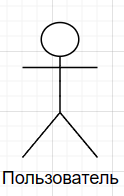


Рисунок 1 – Актёр

Вариант использования – это список конкретных шагов, необходимых для достижения определенной цели, такой как настройка устройства или использование программного обеспечения изображено на рисунке 2. На диаграмме вариантов использования отображаются все возможные варианты использования системы, что позволяет определить и проанализировать функциональные требования.

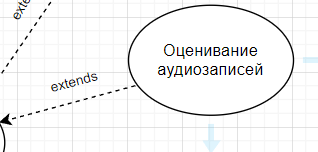


Рисунок 2 - Вариант использования

В данном веб-ресурсе выделены следующие варианты использования для пользователя:

* Просмотр подборок песен;
* Прослушивание и переключение аудиозаписей;
* Оценивание аудиозаписей;
* Скачивание аудиозаписей;
* Просмотр карточки музыканта;
* Просмотр новинок песен;
* Просмотр популярных песен.

Отношение — это описание взаимодействия между актерами и системой, представленное в графическом виде с помощью стрелок. Отношения в диаграмме помогают разработчикам понимать, какие действия могут выполнять пользователи или другие системы в системе, и как система должна реагировать на эти действия.

Отношения ассоциации в диаграмме вариантов использования (Use Case) представляют связи между элементами диаграммы, например, между акторами и вариантами использования или между различными вариантами использования изображено на рисунке 3. Они могут быть двусторонними или односторонними и указывать на связь, обмен информацией или взаимодействие между различными частями системы. Отношения ассоциации могут быть представлены стрелками, которые указывают направление связи или без стрелок, если связь двусторонняя.

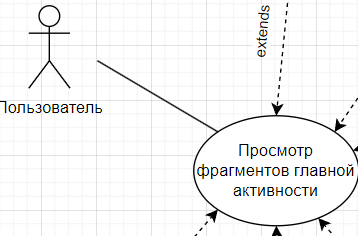


Рисунок 3 - Отношение ассоциации

Отношение дополнения в диаграмме вариантов использования используется для связывания двух взаимоисключающих вариантов использования изображено на рисунке 4. Это означает, что если один из вариантов использования выполняется, то другой не может быть выполнен. Отношение дополнения обозначается открытой стрелкой, указывающей на вариант использования, который является дополнением.

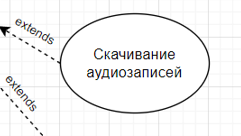


Рисунок 4 – Отношение дополнения

Через анализ вариантов использования и связей между ними для конкретного актера, мы составим полную диаграмму вариантов использования изображено на рисунке 6.

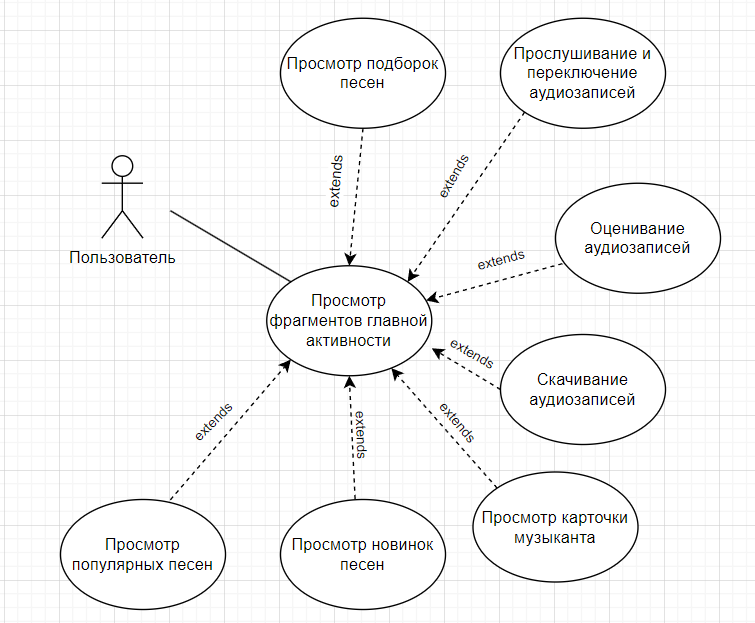


Рисунок 6 – Диаграмма вариантов использования

**4.4 Модель данных**

Модель данных — это абстрактное представление структуры, связей и характеристик данных, используемых в информационной системе или базе данных. Она определяет, как данные будут организованы, храниться и обрабатываться. Модель данных описывает сущности, их атрибуты, связи между ними и ограничения на данные. В зависимости от способа представления структуры данных, существуют различные модели данных, такие как иерархическая, сетевая, реляционная, объектно-ориентированная и другие. Каждая модель данных имеет свои специфические характеристики и применяется в определенных ситуациях в зависимости от требований проекта.

Сущности — это основные объекты данных в модели. Они представляют реальные объекты или концепции, которые хранятся и обрабатываются в системе. Например, в модели данных для интернет-магазина могут быть сущности "пользователь", "товар", "заказ".

Связи — это связи или отношения между сущностями. Они определяют, как сущности взаимодействуют друг с другом. Например, в модели данных для интернет-магазина может быть связь "заказ" между сущностями "пользователь" и "товар", чтобы отразить тот факт, что пользователь делает заказ на определенный товар.

Модель данных изображена на рисунке 7.

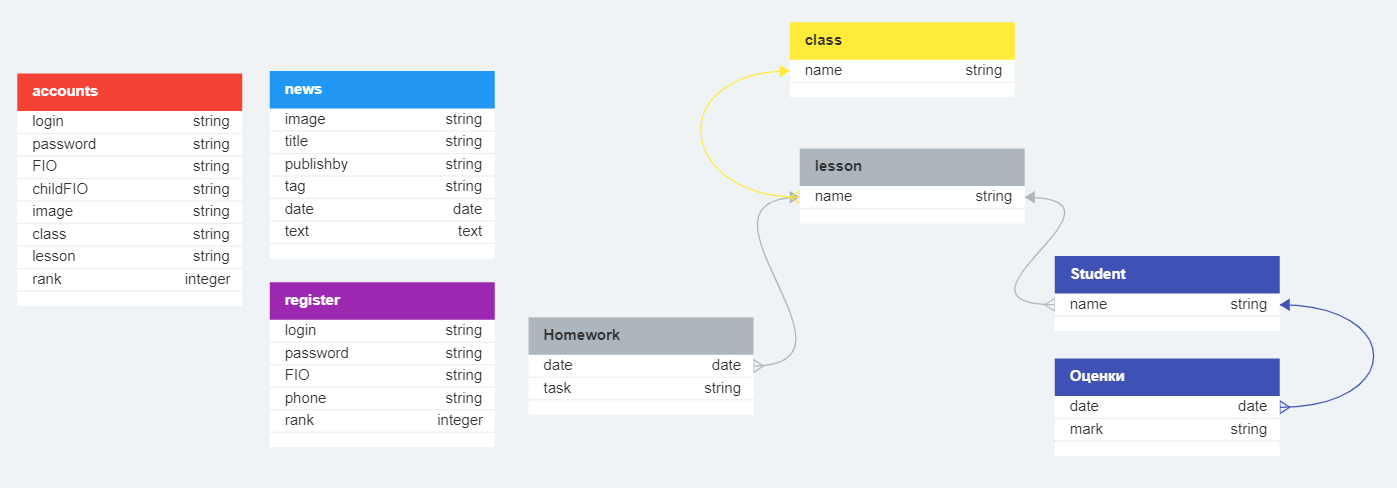


Рисунок 7 - Модель данных

**4.5 Постановка задачи**

Наименование задачи: мобильное приложение для воспроизведения музыки «4Music».

Цель задачи: создание приложения, в котором пользователь может слушать музыку, создавать свой плейлист, оценивать творчество.

Назначение: мобильное приложение предназначено для слушателей музыки, так как оно предоставляет доступ к огромной коллекции музыкальных треков различных жанров, артистов и альбомов. Оно также предлагает создавать собственный плейлист и основе избранных треков пользователя.

Периодичность использования: в любое время;

Источники и способы получения данных: интернет-ресурсы;

Аналогичные программные продукты: MusicMax для системы IOS.

Описание перечня функций и задач, которые должно выполнять приложение:

Пользователь:

- Просмотр и прослушивание последних новинок музыки;

* Просмотр и прослушивание творческих плейлистов;
* Поиск песен в поиске;
* Добавление песен в избранное;
* Оценивание песен по системе оценивания “Риса за творчество”;

- Скачивание песен в формате mp3.

Описание процессов с входной, выходной и условно-постоянной информацией:

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Таблица 1 – Описание входной и выходной информации | | | | | | | |
| Категория пользователей | Наименование процесса | Краткое описание алгоритма выполнения процесса | | Входная информация | Выходная информация | | Условно - постоянная информация |
| Пользователь | Прослушивание аудиозаписи | Включение/выключение песни | | Аудиозапись | Звук | | Название песни, аудиозапись |
| Пользователь | Добавление аудиозаписи в избранное | Создание собственного плейлиста | | Аудиозапись | Плейлист | | Название песни, аудиозапись |
| Пользователь | Поиск песен в поиске | Просмотр песен по поиску | | Название песни | Получение аудиозаписи | | Информация в приложении |
| Пользователь | Оценивание песен | Оценка творчества исполнителей | | Аудиозапись | Оцененное творчество | | Аудиозапись, название песни |
| Пользователь | Скачивание аудиозаписи | Установка мп3 файла | Аудиозапись | | Аудиозапись в формате мп3 | Аудиозапись, название | |

Требования к применению: приложение должно воспроизводить музыку, сохранять ее, скачивать, оценивать.

Требования к реализации: для реализации статических страниц и шаблонов должен использоваться язык XML. Для реализации интерактивных элементов клиентской части должен использоваться язык Kotlin. Для реализации базы данных должен быть использован язык программирования SQL.

Описание пользовательского интерфейса: при разработке веб-ресурса должны быть использовано несколько цветовых оттенков для грамотной палитры цветов. Основные разделы сайта должны быть доступны с первой страницы. Грамотный пользовательский интерфейс.

Требования к хостингу:

* Поддерживает любые CMS;
* Круглосуточный мониторинг серверов 24/7: сбои устраняются раньше, чем их можно заметить;
* Объем дискового пространство 200Мб;
* Эффективная защита от спама и взлома;
* Базы данных хранятся на SSD-дисках;
* Максимальный размер базы данных - 2 Гб;
* Все данные проходят через процедуру ежедневного резервного копирования;
* Копии хранятся в течение 20 дней;

**5 Проектирование задачи**

Для проектирования задачи выбрана среда разработки Android Studio. Данная среда разработки позволит организовать логическую структуру приложения. Пользовательский интерфейс будет реализован просто и понятно для пользователя любого возраста.

Создание самого приложения осуществляется с помощью файлов с расширением .kt и .xml для реализации интерфейса. При связывании интерфейса и функциональной части на выходе выходит готовое приложение.

При первом входе в приложение пользователю откроется основная страница, а именно просмотр последних новинок музыки, где он может сразу включить желаемую композицию и прослушать ее. (Рисунок 8.)

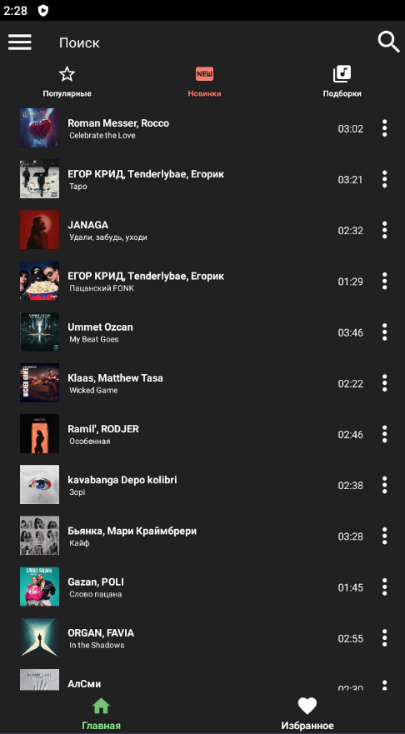


Рисунок 8 – Основная страница

После мы можем перейти во вкладку популярное и просмотреть и прослушать популярные на данный момент песни. (Рисунок 9.)

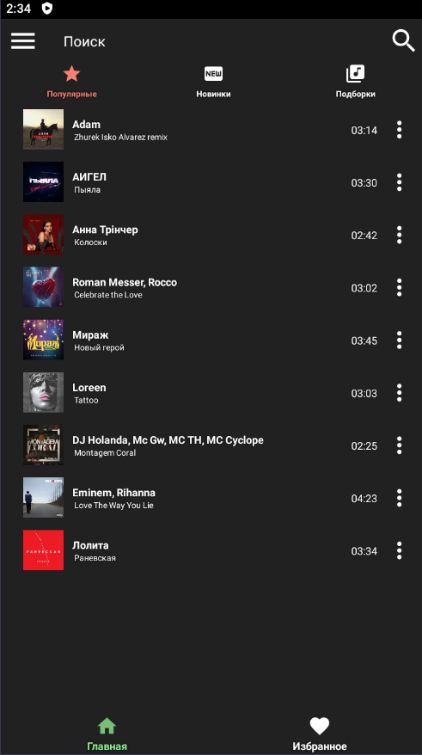
****

Рисунок 9 – Вкладка “Популярные”

Вкладка “Подборки” представляет собой плейлисты подобранные под пользователей на выбор. (Рисунок 10.)

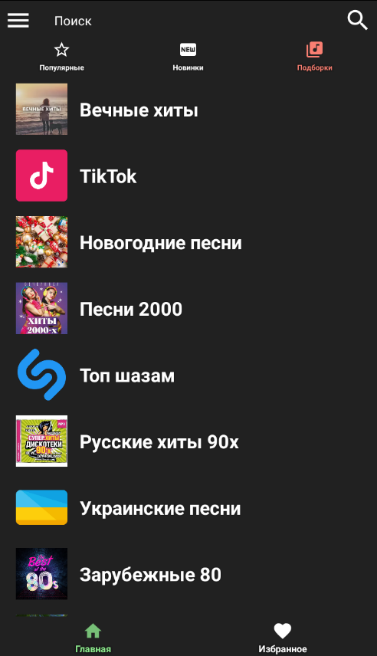


Рисунок 10 – Вкладка “Подборки”

**6 Реализация**

Для реализации проекта выбрана среда разработки Android Studio. С помощью данного инструмента реализованы пользовательский интерфейс и функциональная часть приложения. Данная среда разработки позволяет удобно редактировать функциональную и визуальную часть приложения.

Модули веб-ресурса:

* activity\_artist.xml – файл разметки активности с информацией об артисте;
* activity\_main.xml – файл разметки основной активности;
* AristActivity.kt – файл для реализации функциональной составляющей активности с информацией об артисте;
* MainAcitivty.kt – файл для реализации функциональной составляющей основной активности;
* MusicBdHelper.kt – класс, который создает, пересоздает или удаляет локальную базу данных, хранящую в себе « куки » файлы;
* MusicBdNames.kt – класс, который хранит в себе команды создания и удаления;
* MusicBdManager.kt – класс в котором реализованы пользовательские функции для упрощения работы с локальной базы данных;
* MusicAdapter.kt – класс в котором реализованы пользовательские настройки адаптера для RecyclerView;
* PlaylistAdapter.kt – класс в котором реализованы пользовательские настройки адаптера для RecyclerView;

Спецификация файлов с расширением .xml является тем, что в данном файле разметка активностей и фрагментов.

Спецификация файлов с расширением .kt является тем, что в данном файле находится код для реализации функциональной части приложения. Данный файл позволяет связать все фрагменты приложения и позволяет корректно работать приложению.

**7 Тестирование**

Таблица 2 – отчёт о дефектах и проделанных тест-кейсах.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Иде-нтификатор | Приоритет | Модуль ресурса | Название | Шаги | Ожидаемый результат | Результ-ат |
| TС1 | Высокий | Основная активность, фрагмент новинок | Просмотр новинок | 1 Открыть приложение  2 Выбрать вкладку “Новинки” | Отображение музыкальных новинок | Выполнено |
| TС2 | Высокий | Основная активность,  фрагмент популярных песен | Просмотр популярных песен | Выбрать вкладку “Популярные” | Отображение популярной музыки | Выполнено |
| ТС3 | Высокий | Прослушивание аудиозаписей | Прослушивание песен | Нажать на песню | Воспроизведение музыки | Выполнено |
| TС4 | Высокий | Переключение аудиозаписей | Переключение песен | Нажать на кнопки переключения | Воспроизведение следующей/предыдущей песни | Выполнено |
| TС5 | Высокий | Поиск аудиозаписей | Проверка поиска песен | Нажатие на строку ввода | Успешный поиск песни | Выполнено |
| TC6 | Высокий | Добавление аудиозаписей в избранное | Проверка добавления песен в свой плейлист | 1 выбрать песню  2 нажать на кнопку выбора  3 нажать на кнопку “Добавить в избранное” | Осуществляется добавление песен в плейлист | Выполнено |
|  |  |  |  |  |  |  |
| Продолжение - отчёт о дефектах и проделанных тест-кейсах. | | | | | | |
| ТС7 | Высокий | Оценивание песен | Проверка оценивания песен | 1 Выбрать песню  2 Нажать на кнопку выбора действия  3 Нажать на кнопку “Посмотреть аналитику РЗТ” | Вывод оценки пользователей о песне | Выполнено |
| ТС8 | Высокий | Переход в карточку музыканта | Просмотр песен выбранного исполнителя | 1 Выбрать песню  2 Нажать на кнопку выбора действия  3 Нажать на “Перейти к исполнителю” | Осуществляется взаимодействие с базой данных классов | Выполнено |

**8 Применение**

Руководство пользователя включает в себя следующие характеристики.

Данное приложение разработано под любое Android устройство.

Наименование приложения: « 4Music ». Приложение разработано в среде разработки Android Studio с помощью языков программирования Kotlin и расширяемого языка программирования XML.

Назначение выполняемого приложения:

- Просмотр и прослушивание последних новинок музыки;

* Просмотр и прослушивание творческих плейлистов;
* Поиск песен в поиске;
* Добавление песен в избранное;
* Оценивание песен по системе оценивания “Риса за творчество”;

- Скачивание песен в формате mp3.

На этапе тестирования программного продукта были выявлены минимальные системные требования, которые позволят работать приложению без задержек.

Быстродействие данного приложения во многом зависит от скорости

подключенного к выбранному для эксплуатации мобильному устройству интернета. Несмотря на все реализованные задачи, сайт легко запускается и функционирует на любых устройствах.

Минимальные системные требования для других гаджетов:

* + Разрешение экрана от 320 пикселей;
  + 100MHz процессор;
  + 512 RAM.

Руководство программиста включается в себя следующие характеристики.

Входными данными приложения являются файлы составляющие структуру ресурса, база данных, хранящая данные пользователей, информации о товарах.

Выходными данными являются визуальная составляющая приложения.

XML разметка фрагментов и активностей располагается в файлах acitivity\_artist.xml, activity\_main.xml.

Kotlin код реализующий функционал располагается в файлах ArtistActivity.kt, MainActivity.kt, MusicBdNames.kt, MusicBdHelper.kt, MusicBdManager.kt, MusicAdapter.kt , PlaylistAdapter.kt

Спецификация файлов приведена в таблице 3.

|  |  |
| --- | --- |
| Таблица 3 – Спецификация файлов | |
| Имя файла | Назначение |
| acitivity\_artist.xml | В данном файле находится разметка активности об исполнителе |
| activity\_main.xml | В данном файле находится разметка основной активности |
| ArtistActivity.kt | В данном файле реализована функциональная составляющая активности с авторизацией |
| MainActivity.kt | В данном файле реализована функциональная составляющая основной активности |
| MusicBdNames.kt | В данном файле хранится класс, который хранит в себе команды создания и удаления таблиц |
| MusicBdHelper.kt | В данном файле хранится класс, который создает, пересоздает или удаляет локальную базу данных, хранящую в себе « куки » файлы, которые определяют, вошли вы в аккаунт или нет |
| MusicBdManager.kt | В данном файле хранится класс, который хранит в себе команды создания и удаления таблиц; |
| MusicAdapter.kt, PlaylistAdapter.kt | В данном файле хранится класс в котором реализованы пользовательские настройки адаптера для RecyclerView; |

**Заключение**

Целью проекта по технологической практике была разработка мобильного приложения для воспроизведения музыки «4Music».

Особенность данного задания заключалась в том, чтобы, реализовать мобильное приложение с помощью Android Studio.

В поставленной задаче был реализован простой и понятный пользовательский интерфейс.

В ходе реализации поставленной задачи был укреплён пройденный курс программирования в среде разработки Android Studio, а также получено много дополнительной информации о Kotlin, XML, SqlLite.

После тестирования мобильного приложения были выявлены недоработки, которые были по большей части исправлены на стадии проектирования, и полностью исключены на стадии тестирования приложения. При реализации приложения, были выполнены все условия, перечисленные в предыдущих разделах пояснительной записки.

# Список использованных источников

* Android Studio [Электронный ресурс]. – Электронные данные. Режим доступа: https://developer.android.com/studio;
* Firebase [Электронный ресурс]: – Электронные данные. Режим доступа: https://firebase.google.com/docs/database?hl=ru.

**Приложение А**

**Листинг программы**

package com.holysource.myschool.Activities  
  
import android.annotation.SuppressLint  
import android.app.Activity  
import android.content.Context  
import android.content.Intent  
import android.os.Bundle  
import android.text.InputType  
import android.util.Log  
import android.view.View  
import android.view.animation.Animation  
import android.view.animation.AnimationUtils  
import android.view.inputmethod.InputMethodManager  
import android.widget.Button  
import android.widget.EditText  
import android.widget.ImageButton  
import android.widget.LinearLayout  
import android.widget.Spinner  
import android.widget.TextView  
import android.widget.Toast  
import androidx.appcompat.app.AppCompatActivity  
import com.google.firebase.firestore.ktx.firestore  
import com.google.firebase.ktx.Firebase  
import com.holysource.myschool.Adapters.CustomToast  
import com.holysource.myschool.R  
import com.skydoves.powerspinner.PowerSpinnerView  
import com.squareup.picasso.Picasso  
import db.SchoolDbManager  
import kotlin.math.log  
  
class LoginActivity : AppCompatActivity() {  
  
 var count = 0  
 val fdb = Firebase.*firestore* val db = SchoolDbManager(this)  
   
 private lateinit var mLoginMenu: LinearLayout  
 private lateinit var mLoginField: EditText  
 private lateinit var mPasswordField: EditText  
 private lateinit var mPasswordVisiblityButton: ImageButton  
 private lateinit var mTryToLoginButton: Button  
 private lateinit var mToRegisterMenuButton: TextView  
 private lateinit var mRegisterMenu: LinearLayout  
 private lateinit var mRegisterLoginField: EditText  
 private lateinit var mRegisterPasswordField: EditText  
 private lateinit var mRegisterPasswordVisiblityButton: ImageButton  
 private lateinit var mRegisterAgainPasswordField: EditText  
 private lateinit var mRegisterFIOField: EditText  
 private lateinit var mRegisterRankSpinner: PowerSpinnerView  
 private lateinit var mRegisterPhoneField: EditText  
 private lateinit var mTryToRegisterButton: Button  
 private lateinit var mToLoginMenuButton: TextView  
  
 @SuppressLint("MissingInflatedId")  
 override fun onCreate(savedInstanceState: Bundle?) {  
 super.onCreate(savedInstanceState)  
 db.openSchoolDb()  
 val cookies = db.Cookies()  
 db.closeSchoolDb()  
 if(cookies.get("IsAuth")=="true") {  
 checkAuth(login = cookies.get("Login")!!, password = cookies.get("Password")!!)  
 } else {  
 setContentView(R.layout.*activity\_login*)  
  
val animAlpha: Animation = AnimationUtils.loadAnimation(*applicationContext*, R.anim.*alpha*)  
 val animScale: Animation = AnimationUtils.loadAnimation(*applicationContext*, R.anim.*scale*)  
  
 mLoginMenu = findViewById(R.id.*login\_menu*)  
 mRegisterMenu = findViewById(R.id.*register\_menu*)  
 mLoginField = findViewById(R.id.*login\_field*)  
 mRegisterLoginField = findViewById(R.id.*register\_login\_field*)  
 mPasswordField = findViewById(R.id.*password\_field*)  
 mPasswordVisiblityButton = findViewById(R.id.*password\_visiblity\_button*)  
 mPasswordVisiblityButton.setOnClickListener **{  
 it**.startAnimation(animScale)  
 if(mPasswordField.*inputType*!= InputType.*TYPE\_CLASS\_TEXT* or InputType.*TYPE\_TEXT\_VARIATION\_PASSWORD*) {  
 mPasswordField.*inputType* = InputType.*TYPE\_CLASS\_TEXT* or InputType.*TYPE\_TEXT\_VARIATION\_PASSWORD* } else {  
 mPasswordField.*inputType* = InputType.*TYPE\_TEXT\_VARIATION\_VISIBLE\_PASSWORD* }  
 mPasswordField.clearFocus()  
 **}** mRegisterPasswordField = findViewById(R.id.*register\_password\_field*)  
 mRegisterAgainPasswordField = findViewById(R.id.*register\_password\_again\_field*)  
 mRegisterPasswordVisiblityButton = findViewById(R.id.*register\_password\_visiblity\_button*)  
 mRegisterPasswordVisiblityButton.setOnClickListener **{  
 it**.startAnimation(animScale)  
 if(mRegisterPasswordField.*inputType*!= InputType.*TYPE\_CLASS\_TEXT* or InputType.*TYPE\_TEXT\_VARIATION\_PASSWORD*) {  
 mRegisterPasswordField.*inputType* = InputType.*TYPE\_CLASS\_TEXT* or InputType.*TYPE\_TEXT\_VARIATION\_PASSWORD* mRegisterAgainPasswordField.*inputType* = InputType.*TYPE\_CLASS\_TEXT* or InputType.*TYPE\_TEXT\_VARIATION\_PASSWORD* } else {  
 mRegisterPasswordField.*inputType* = InputType.*TYPE\_TEXT\_VARIATION\_VISIBLE\_PASSWORD* mRegisterAgainPasswordField.*inputType* =  
 InputType.*TYPE\_TEXT\_VARIATION\_VISIBLE\_PASSWORD* }  
 mRegisterPasswordField.clearFocus()  
 mRegisterAgainPasswordField.clearFocus()  
 **}** mRegisterFIOField = findViewById(R.id.*register\_fio\_field*)  
 mRegisterRankSpinner = findViewById(R.id.*register\_rank\_spinner*)  
 mRegisterPhoneField = findViewById(R.id.*register\_return\_call\_field*)  
 mTryToLoginButton = findViewById(R.id.*try\_to\_login\_button*)  
 mTryToLoginButton.setOnClickListener **{  
 it**.startAnimation(animAlpha)  
 if(mLoginField.*text*.*isNotEmpty*() && mPasswordField.*text*.*isNotEmpty*()) {  
 if(mLoginField.*text*.length>=4 && mPasswordField.*text*.length>=6) {  
 login(mLoginField.*text*.toString(),mPasswordField.*text*.toString())  
 mPasswordField.clearFocus()  
 mPasswordField.*text*.clear()  
 mLoginField.clearFocus()  
 mLoginField.*text*.clear()  
 } else {  
 if(mLoginField.*text*.length<4) {  
 toast("Упс... Логин состоит минимум из 4 символов!",null)  
 mLoginField.startAnimation(animScale)  
 } else {  
 toast("Упс... Пароль состоит минимум из 6 символов!",null)  
 mPasswordField.startAnimation(animScale)  
 }  
 }  
 } else {  
 if(mLoginField.*text*.*isEmpty*()) {  
 mLoginField.startAnimation(animScale)  
 } else {  
 mPasswordField.startAnimation(animScale)  
 }  
 toast("Упс... Кажется не все поля заполены!",null)  
 }  
 hideKeyboardFrom(*applicationContext*,mTryToLoginButton)  
 **}** mTryToRegisterButton = findViewById(R.id.*try\_to\_register\_button*)  
 mTryToRegisterButton.setOnClickListener **{  
 it**.startAnimation(animAlpha)  
 if(mRegisterLoginField.*text*.*isNotEmpty*() && mRegisterPasswordField.*text*.*isNotEmpty*() && mRegisterAgainPasswordField.*text*.*isNotEmpty*() && mRegisterFIOField.*text*.*isNotEmpty*() && mRegisterRankSpinner.selectedIndex>=0 && mRegisterPhoneField.*text*.*isNotEmpty*()) {  
 if(mRegisterPasswordField.*text*.toString()==mRegisterAgainPasswordField.*text*.toString()) {  
 if(mRegisterLoginField.*text*.length>=4) {  
 if(mRegisterPasswordField.*text*.length>=6) {  
 register(mRegisterLoginField.*text*.toString(),mRegisterPasswordField.*text*.toString(),mRegisterFIOField.*text*.toString(),(mRegisterRankSpinner.selectedIndex+1).toString(),mRegisterPhoneField.*text*.toString())  
 } else {  
 toast("Пароль должен состоять минимум из 6 символов!",null)  
 }  
  
 } else {  
 toast("Логин должен состоять минимум из 4 символов",null)  
 }  
 } else {  
 toast("Пароли не совпадают. Попробуйте еще раз.",null)  
 }  
 } else {  
 toast("Упс... Кажется не все поля заполены!",null)  
 }  
 hideKeyboardFrom(*applicationContext*,mTryToRegisterButton)  
 **}** mToRegisterMenuButton = findViewById(R.id.*to\_register\_menu\_button*)  
 mToRegisterMenuButton.setOnClickListener **{  
 it**.startAnimation(animScale)  
 mLoginMenu.*visibility* = View.*GONE* mRegisterMenu.*visibility* = View.*VISIBLE* **}** mToLoginMenuButton = findViewById(R.id.*to\_login\_menu\_button*)  
 mToLoginMenuButton.setOnClickListener **{  
 it**.startAnimation(animScale)  
 mLoginMenu.*visibility* = View.*VISIBLE* mRegisterMenu.*visibility* = View.*GONE* **}** }  
 }  
  
 fun toast(text: String, isDone: Boolean?) {  
 CustomToast.makeText(*applicationContext*, text, isDone).show()  
 }  
  
 fun checkAuth(login: String,password: String) {  
 var ErrorCode = 1  
 var FIO = ""  
 var childFIO = ""  
 var rank = ""  
 var clas = ""  
 var lesson = ""  
 var image = "null"  
 fdb.collection("accounts").document(login).get().addOnSuccessListener **{** result **->** if(password==result.*data*?.get("password")) {  
 ErrorCode = 0  
 FIO = result.*data*!!.get("FIO").*toString*()  
 rank = result.*data*!!.get("rank").*toString*()  
 clas = result.*data*!!.get("class").*toString*()  
 lesson = result.*data*!!.get("lesson").*toString*()  
 image = result.*data*!!.get("image").*toString*()  
 childFIO = result.*data*!!.get("childFIO").*toString*()  
 }  
 db.openSchoolDb()  
 if(ErrorCode==0) {  
 db.updateCookies(login,password,FIO,childFIO,rank,clas,lesson,image,"true")  
 db.closeSchoolDb()  
 val intent = Intent(this@LoginActivity, MainActivity::class.*java*)  
 startActivity(intent)  
 } else {  
 db.updateCookies("","","","","","","","null","false")  
 db.closeSchoolDb()  
 super.onCreate(null)  
 }  
 **}**.addOnFailureListener **{** Log.e("TAG", "Ошибка: ${**it**}")  
 **}** }  
  
 fun register(login: String,password: String,fio: String,rank: String,phone: String) {  
  
 val reg = *hashMapOf*(  
 "login" *to* login,  
 "password" *to* password,  
 "FIO" *to* fio,  
 "rank" *to* rank,  
 "phone" *to* phone,  
 )  
 fdb.collection("accounts").whereEqualTo("login",login).get().addOnSuccessListener **{** result **->** if(result.size()==0) {  
 fdb.collection("register").whereEqualTo("login",login).get().addOnSuccessListener **{** res **->** if(res.size()==0) {  
  
 fdb.collection("register").document(login).set(reg)  
 .addOnSuccessListener **{** toast("Вы оставили заявку. Ждите, пока администратор рассмотрит ее.",true)  
 **}** .addOnFailureListener **{** toast("Произошла неизвестная ошибка. Попробуйте еще раз.",false)  
 **}** } else {  
 toast("Заявка на аккаунт с таким логином уже существует",false)  
 }  
  
 **}**.addOnFailureListener **{** toast("Произошла неизвестная ошибка. Попробуйте еще раз.",false)  
 **}** }  
  
 **}**.addOnFailureListener **{** toast("Аккаунт с таким логином уже существует.",false)  
 **}** }  
  
 fun login(login: String,password: String) {  
 val animScale: Animation = AnimationUtils.loadAnimation(*applicationContext*, R.anim.*scale*)  
 var ErrorCode = 2  
 var FIO = ""  
 var childFIO = ""  
 var rank = 0  
 var clas = ""  
 var lesson = ""  
 var image = "null"  
 fdb.collection("accounts").document(login).get().addOnSuccessListener **{** result **->** if(login==result.*data*?.get("login")) {  
 if(password==result.*data*?.get("password")) {  
 ErrorCode = 0  
 FIO = result.*data*!!.get("FIO").*toString*()  
 rank = result.*data*!!.get("rank").*toString*().*toInt*()  
 clas = result.*data*!!.get("class").*toString*()  
 lesson = result.*data*!!.get("lesson").*toString*()  
 image = result.*data*!!.get("image").*toString*()  
 childFIO = result.*data*!!.get("childFIO").*toString*()  
 } else {  
 ErrorCode = 1  
 }  
 }  
 when(ErrorCode) {  
 0 -> {  
 db.openSchoolDb()  
 db.updateCookies(login,password,FIO,childFIO,rank.toString(),clas,lesson,image,"true")  
 db.closeSchoolDb()  
 when(rank) {  
 1 -> {  
 toast("Вы авторизовались как Ученик",true)  
 }  
 2 -> {  
 toast("Вы авторизовались как Родитель",true)  
 }  
 3 -> {  
 toast("Вы авторизовались как Преподаватель",true)  
 }  
 4 -> {  
 toast("Вы авторизовались как Администратор",true)  
 }  
 }  
 val intent = Intent(this@LoginActivity, MainActivity::class.*java*)  
 startActivity(intent)  
 }  
 1 -> {  
 toast("Упс... Вы ввели неправильный пароль! Попробуйте еще раз.",false)  
 mPasswordField.startAnimation(animScale)  
 }  
 2 -> {  
 toast("Такого аккаунта не существует. Подайте заявку, чтобы продолжить.",false)  
 mToRegisterMenuButton.startAnimation(animScale)  
 }  
 }  
 **}**.addOnFailureListener **{** Log.e("TAG", "Ошибка: ${**it**}")  
 **}** }  
  
 override fun onResume() {  
 count = 0  
 super.onResume()  
 }  
  
 @Deprecated("Deprecated in Java")  
 @SuppressLint("MissingSuperCall")  
 override fun onBackPressed() {  
 if(count==0) {  
 toast("Нажмите еще раз, чтобы свернуть",null)  
 count = 1  
 } else {  
 val startMain = Intent(Intent.*ACTION\_MAIN*)  
 startMain.addCategory(Intent.*CATEGORY\_HOME*)  
 startMain.*flags* = Intent.*FLAG\_ACTIVITY\_NEW\_TASK* startActivity(startMain)  
 }  
 }  
  
 fun hideKeyboardFrom(context: Context, view: View?) {  
 val imm = context.getSystemService(Activity.*INPUT\_METHOD\_SERVICE*) as InputMethodManager  
 imm.hideSoftInputFromWindow(view?.*windowToken*, 0)  
 }  
}

package com.holysource.myschool.Activities  
  
import android.annotation.SuppressLint  
import android.content.Intent  
import android.os.Bundle  
import androidx.appcompat.app.AppCompatActivity  
import androidx.navigation.findNavController  
import androidx.navigation.ui.AppBarConfiguration  
import androidx.navigation.ui.setupWithNavController  
import com.google.android.material.bottomnavigation.BottomNavigationView  
import com.holysource.myschool.Adapters.CustomToast  
import com.holysource.myschool.R  
import com.holysource.myschool.databinding.ActivityMainBinding  
  
  
class MainActivity : AppCompatActivity() {  
  
 var count = 0  
 private lateinit var binding: ActivityMainBinding  
  
 override fun onCreate(savedInstanceState: Bundle?) {  
 super.onCreate(savedInstanceState)  
  
 binding = ActivityMainBinding.inflate(*layoutInflater*)  
 setContentView(binding.*root*)  
  
 val navView: BottomNavigationView = binding.navView  
  
 val navController = *findNavController*(R.id.*nav\_host\_fragment\_activity\_main*)  
 *// Passing each menu ID as a set of Ids because each  
 // menu should be considered as top level destinations.* val appBarConfiguration = *AppBarConfiguration*(  
 *setOf*(  
 R.id.*navigation\_news*, R.id.*navigation\_study*, R.id.*navigation\_profile* )  
 )  
 navView.*setupWithNavController*(navController)  
 }  
  
 fun toast(text: String, isDone: Boolean?) {  
 CustomToast.makeText(*applicationContext*, text, isDone).show()  
 }  
  
 override fun onResume() {  
 count = 0  
 super.onResume()  
 }  
  
 @SuppressLint("MissingSuperCall")  
 override fun onBackPressed() {  
 if(count==0) {  
 toast("Нажмите еще раз, чтобы свернуть",null)  
 count = 1  
 } else {  
 val startMain = Intent(Intent.*ACTION\_MAIN*)  
 startMain.addCategory(Intent.*CATEGORY\_HOME*)  
 startMain.*flags* = Intent.*FLAG\_ACTIVITY\_NEW\_TASK* startActivity(startMain)  
 }  
 }  
}