**ПРАВИТЕЛЬСТВО РОССИЙСКОЙ ФЕДЕРАЦИИ**

**НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИ УНИВЕРСИТЕТ**

**«ВЫСШАЯ ШКОЛА ЭКОНОМИКИ»**

Факультет компьютерных наук

Департамент программной инженерии

|  |  |
| --- | --- |
| Подп. и дата |  |
| Инв. № дубл. |  |
| Взам. инв. № |  |
| Подп. и дата |  |
| Инв. №подл |  |

СОГЛАСОВАНО

Старший преподаватель департамента программной инженерии факультета компьютерных наук

Д.В. Пантюхин

«\_\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 2021 г.

УТВЕРЖДАЮ

Академический руководитель образовательной программы «Программная инженерия», к.т.н., профессор ДПИ ФКН

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ В.В. Шилов

«\_\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_\_ 2021 г.

**Клиентская часть приложения для генерации музыки**

**Пояснительная записка**

**ЛИСТ УТВЕРЖДЕНИЯ**

**RU.17701729.05.01-01 81 01-1-ЛУ**

**Исполнитель**

Студент группы БПИ181

\_\_\_\_\_\_\_\_\_\_\_\_ / Е. Д. Плющ /

«\_\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_\_ 2021 г.

**Москва 2021**

**УТВЕРЖДЕН**

**RU.17701729.05.01-01 81 01-1 ЛУ**

|  |  |
| --- | --- |
| Подп. и дата |  |
| Инв. № дубл. |  |
| Взам. инв. № |  |
| Подп. и дата |  |
| Инв. №подл |  |

**Клиентская часть приложения для генерации музыки**

**Пояснительная записка**

**RU.17701729.05.01-01 81 01-1**

**Листов 17**

**Москва 2021**

[1. Введение 3](#_Toc69413602)

[1.1. Наименование программы 3](#_Toc69413603)

[1.2. Документы, на основании которых ведется разработка 3](#_Toc69413604)

[2. НАЗНАЧЕНИЕ И ОБЛАСТЬ ПРИМЕНЕНИЯ 4](#_Toc69413605)

[2.1. Назначение программы 4](#_Toc69413606)

[2.2. Краткая характеристика области применения 4](#_Toc69413607)

[3. ТЕХНИЧЕСКИЕ ХАРАКТЕРИСТИКИ 5](#_Toc69413608)

[3.1. Постановка задачи на разработку программы 5](#_Toc69413609)

[3.2. Описание алгоритма и функционала программы 5](#_Toc69413610)

[3.3. Описание и обоснование выбора технических средств 9](#_Toc69413611)

[4. Ожидаемые технико-экономические показатели 11](#_Toc69413612)

[4.1. Предполагаемая потребность 11](#_Toc69413613)

[4.2. Экономические преимущества разработки по сравнению с отечественными и зарубежными аналогами. 11](#_Toc69413614)

[5. Дальнейшие перспективы развития 12](#_Toc69413615)

[Список использованных источников 13](#_Toc69413616)

[ПРИЛОЖЕНИЕ 1 14](#_Toc69413617)

[ПРИЛОЖЕНИЕ 2 17](#_Toc69413618)

[ПРИЛОЖЕНИЕ 3 18](#_Toc69413619)

# Введение

## Наименование программы

Наименование программы - «Клиентская часть приложения для генерации музыки». Наименование программы на английском языке – «Application for music generation». Пользовательское наименование – «Sonus»

## Документы, на основании которых ведется разработка

Программа выполнена в рамках темы курсового проекта – «Приложение для генерации музыки». Основанием для разработки является учебный план подготовки бакалавров по направлению 09.03.04 «Программная инженерия» и утвержденная академическим руководителем тема курсового проекта.

# НАЗНАЧЕНИЕ И ОБЛАСТЬ ПРИМЕНЕНИЯ

## Назначение программы

* + 1. Функциональное назначение

Клиентская часть приложения для генерации музыки реализует основные функции пользователя, которые должно поддерживать «Приложение для генерации музыки». Клиентская часть представляет из себя десктопное приложение, основные функции которого – предоставить интерфейс, с помощью которого пользователь сможет воспользоваться серверной частью для генерации музыки.

* + 1. Эксплуатациональное назначение

Клиентская часть приложения для генерации музыки создана как интерфейс для работы с серверной частью приложения

## Краткая характеристика области применения

Данное программное обеспечение предназначено для создания музыкальных композиций на основе предложенных треков с использованием технологий искусственного интеллекта и систем машинного обучения. Для сохранения контента и его структуризации используется система регистрации пользователя и авторизации пользователя.

# ТЕХНИЧЕСКИЕ ХАРАКТЕРИСТИКИ

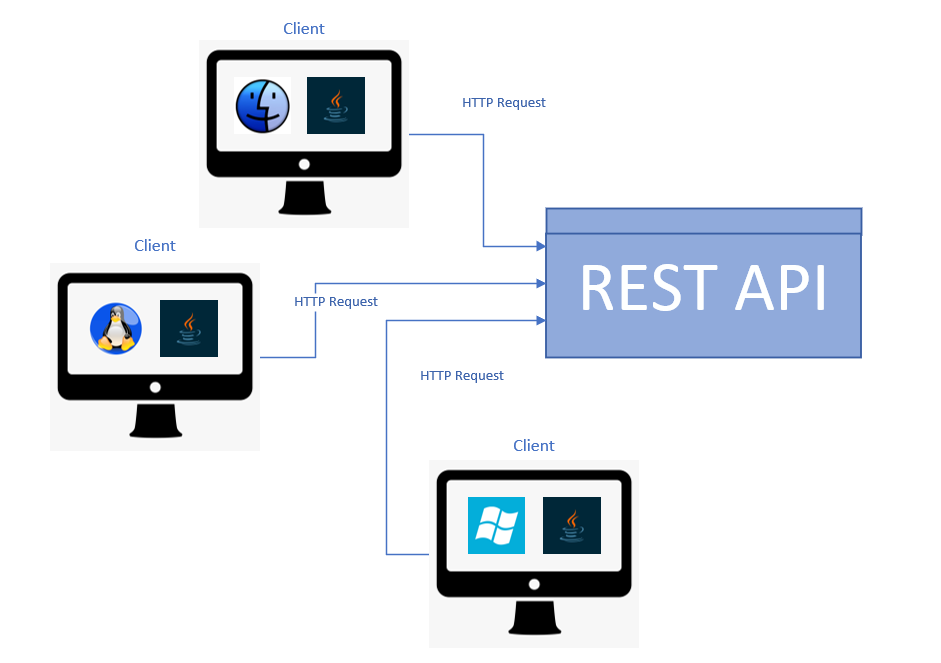
## Постановка задачи на разработку программы

Разрабатываемая программа должна:

1. Позволять зарегистрироваться
2. Позволять авторизоваться
3. Позволять задавать параметры генерации музыки
4. Позволять выбирать музыку как основу для генерации
5. Позволять проигрывать сгенерированные треки
6. Позволять создавать плейлисты
7. Позволять проигрывать плейлисты

## Описание алгоритма и функционала программы

* + 1. Описание общей архитектуры программы

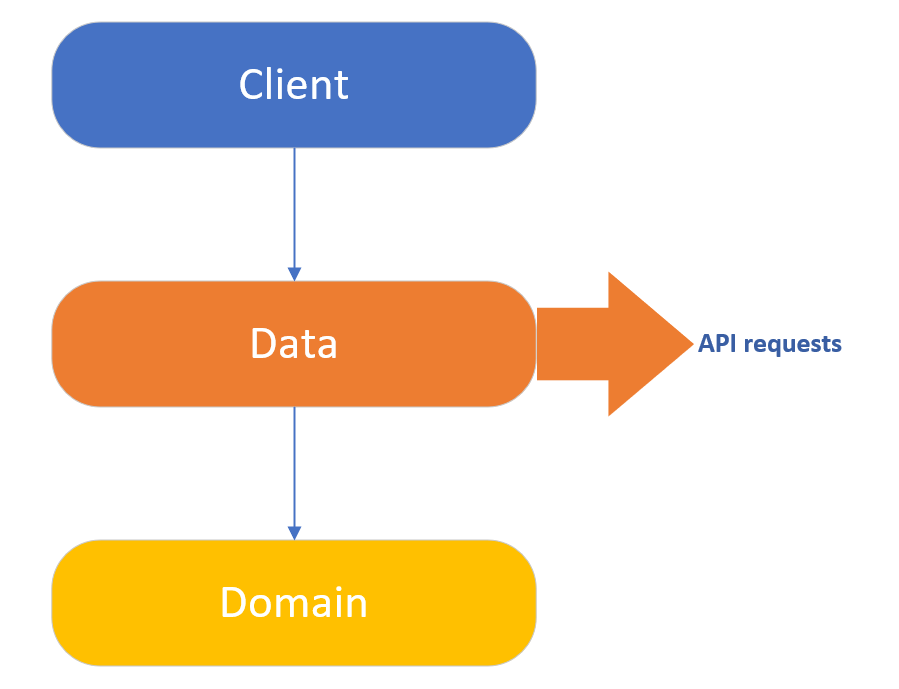
 Общая схема взаимодействия серверной и клиентской части изображена на рисунке 1

**Рисунок 1 – общая схема взаимодействия**

* + 1. Описание архитектуры клиентской части программы

Код клиентской части приложения разделен на три слоя: Domain (слой бизнес-логики),

Data (слой получения данных) и Client (слой непосредственно приложения)

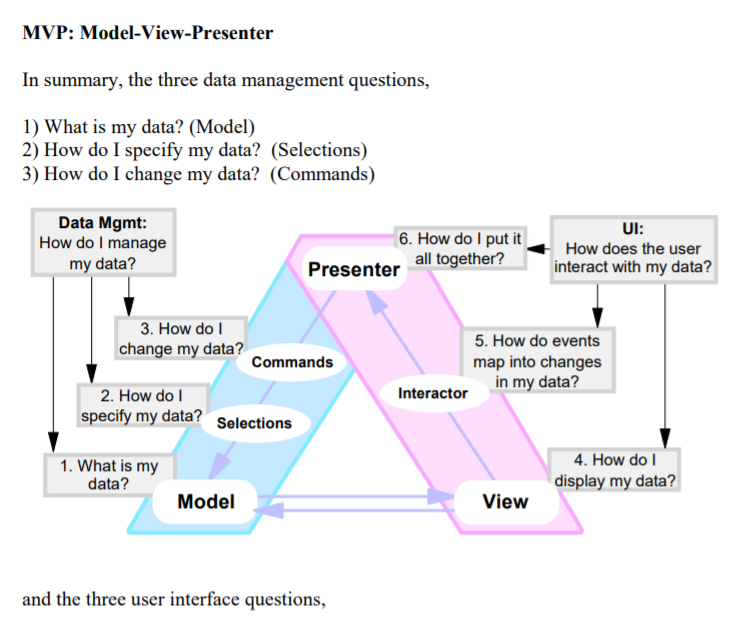


**Рисунок 2 – многослойная архитектура**

Подобное устройство слоев, следующее принципам Clean Architecture [1], помогает добиться простоты в изменениях и возможности переиспользования модулей.

1. Устройство слоя Client

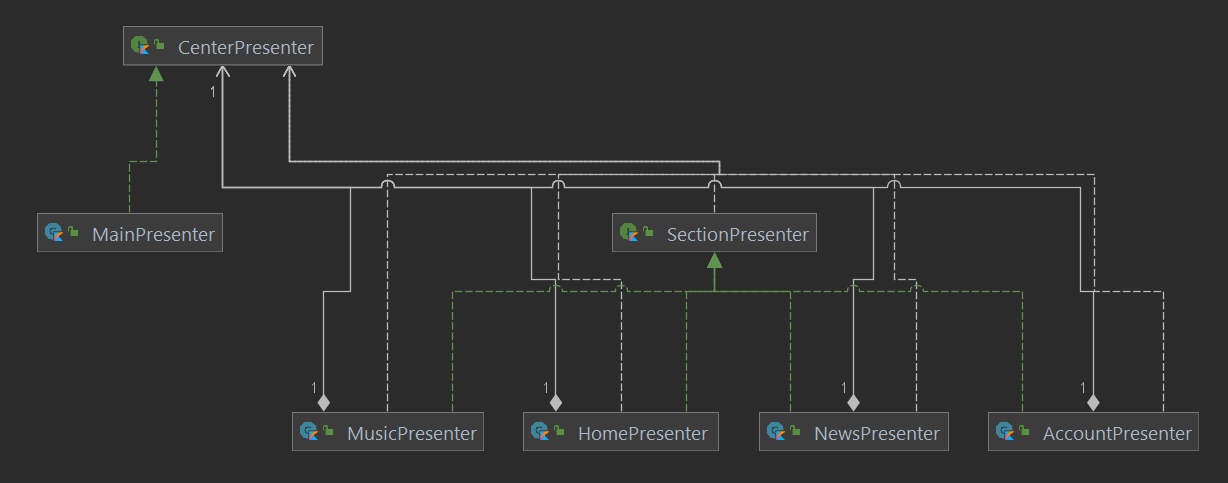
Устройство слоя Client представляет собой некоторую усложненную реализацию классического паттерна MVP (Model-View-Presenter) [2]:



**Рисунок 3 – иллюстрация из источника [2]**

В нашем конкретном случае реализации Client представляет из себя следующую диаграмму:

Слой Presenters:

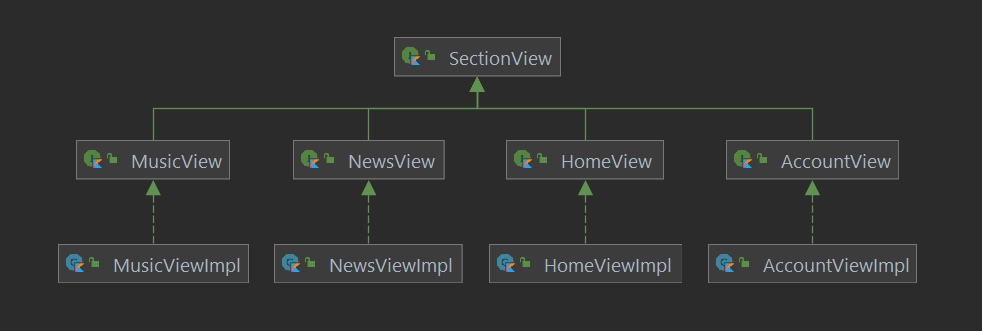


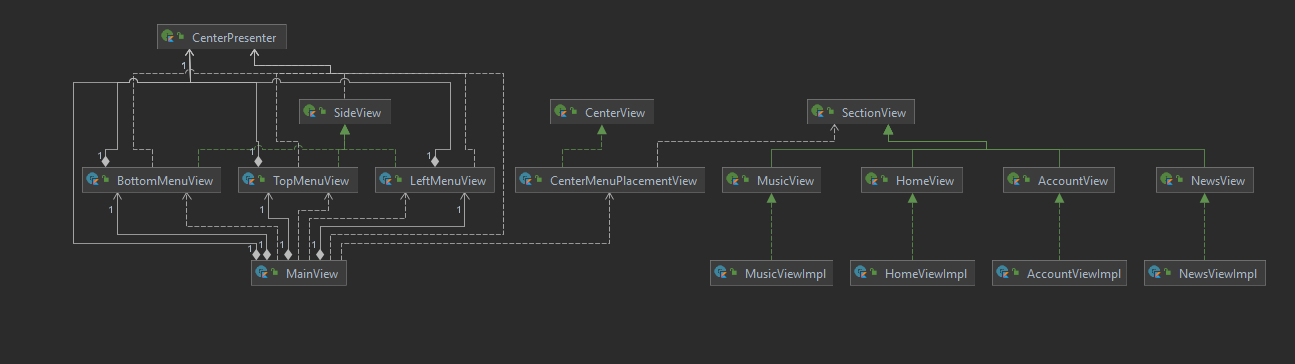
**Рисунок 3 – диаграмма классов Presenters**

Главный презентер – CenterPresenter – контролирует состояние всего окна

Конкретные реализации SectionPresenter имеют ссылку на главный презентер, в то время как он не следит за ними.

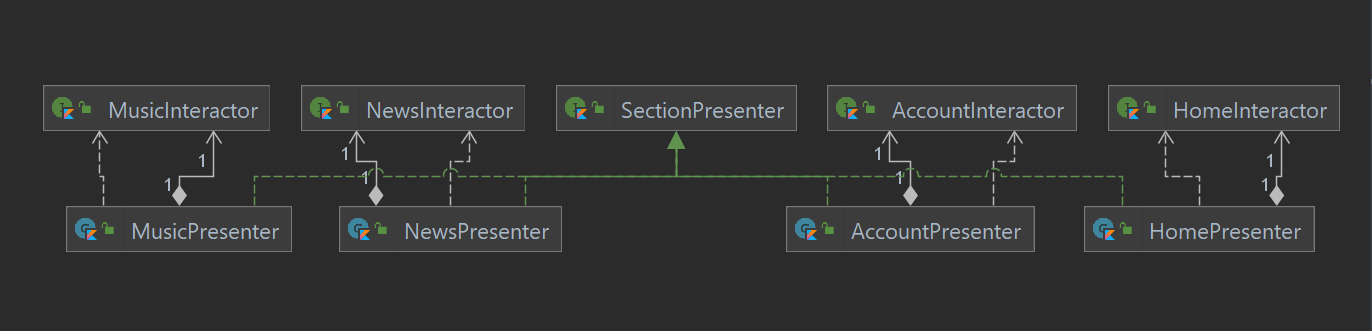
Слой Views:



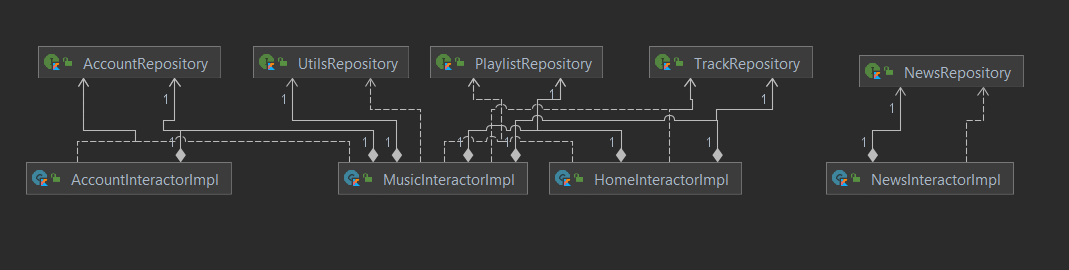
**Рисунок 4 – диаграмма классов Views**

***Рисунок 5 – диаграмма классов Views(полная)***

MainView – это основной View (корень для остальных). Он является родителем для меню по сторонам и CenterMenuPlacementView, который подставляет конкретную реализацию секции в зависимости от выбора пользователя

Конкретные SectionView также содержат ссылки на конкретные презентеры (см Приложение 3) по секциям

***Рисунок 6 – диаграмма классов Presenters и Interactors***

В данном случае моделями из слоя Domain управляют интеракторы (см Приложение 3), которые в конкретной своей реализации включают в себя зависимости от репозиториев и слоя Data

***Рисунок 7 – диаграмма классов Interactors и Repository***

Благодаря использованию Dependency Injection [3] с использованием библиотеки Dagger [4] достигается минимальная зависимость между компонентами.

Также стоит отметить, что слой Client реализован при помощи compose-стиля (см Приложение 3).

Данный слой зависит от двух нижних слоев – Data и Domain.

1. **Устройство слоя Data**

Слой Data реализует по своей сути паттерн Repository [5], организуя доступ к данным с помощью реализации интерфейсов репозиториев из слоя Domain, а также делая внутри запросы к серверной части с помощью Retrofit [6].

Данный слой зависит только от слоя Domain.

1. **Устройство слоя Domain**

Слой Domain состоит из трех основных компонентов: пакет моделей, пакет интерфейсов репозиториев и пакет интерфейсов интеракторов. Этот модуль является основой бизнес-логики приложения и не зависит от остальных

## Описание и обоснование выбора технических средств

* + 1. Состав технических и программных средств

Для нормального функционирования клиентской части необходим персональный компьютер c операционной системой Windows 10, Ubuntu, Fedora, Debian или Mac OS 9 и выше, с установленной JVM версии не ниже 15 и следующими техническими характеристиками:

- процессор не ниже Intel Core I5 generation 8 или аналогичных ему с тактовой частотой не ниже 1.8 ГГц и количеством ядер не менее 4;

- не менее 4096 Мб ОЗУ;

- хранилище памяти с объемом свободной памяти не менее 64 Гб;

- доступ к сети Интернет;

- поддержка формата файлов \*.wav.

* + 1. Обоснование выбора технических и программных средств

При реализации клиентской части был использован язык Kotlin версии 1.4.19 и Javа версии 15 с использованием TornadoFx 1.7.20 (использующий внутри себя JavaFX [6]), поэтому для нормального функционирования требования к системе взяты из [6]

В качестве реализации были выбраны именно языки Kotlin и Java, поскольку они поддерживают кроссплатформенность и в том числе полученный код будет легко портировать на мобильные устройства

# Ожидаемые технико-экономические показатели

## Предполагаемая потребность

В рамках данной работы оценка предполагаемой потребности не предусмотрена.

## Экономические преимущества разработки по сравнению с отечественными и зарубежными аналогами.

В рамках данной работы оценка экономических показателей не предусмотрена.

# Дальнейшие перспективы развития

«Клиентскую часть приложения для генерации музыки» можно продолжить развивать в следующих направлениях:

1. Создание Android-приложения на основе кодовой базы на Kotlin
2. Расширение и увеличение количества особенностей для привлечения пользователей
3. Создать Web-версию с помощью Kotlin Web, используя уже готовую кодовую базу

# Список использованных источников

1. **Мартин Роберт С.** Чистая архитектура. Искусство разработки программного обеспечения // Питер, 2018 г. – С. 10-400
2. **Mike Potel** MVP: Model-View-Presenter The Taligent Programming Model for C++ and Java // Taligent, Inc, 1996 – С. 6 - 8
3. Dependency Injection [Электронный ресурс]// URL: <https://habr.com/ru/post/350068/> (Дата обращения: 10.01.2021, режим доступа: свободный)
4. Dagger [Электронный ресурс]// URL: <https://dagger.dev/dev-guide/> (Дата обращения: 10.01.2021, режим доступа: свободный)
5. Repository [Электронный ресурс]// URL <https://www.cosmicpython.com/book/chapter_02_repository.html> (Дата обращения: 10.01.2021, режим доступа: свободный)
6. Retrofit [Электронный ресурс]// URL <https://square.github.io/retrofit/> (Дата обращения: 10.01.2021, режим доступа: свободный)
7. JavaFx [Электронный ресурс]// URL: <https://www.oracle.com/java/technologies/javafx/system-requirements.html> (Дата обращения: 10.01.2021, режим доступа: свободный)

# ПРИЛОЖЕНИЕ 1

**Описание используемых классов**

|  |  |
| --- | --- |
| **Класс** | **Описание класса** |
| Config.kt | Конфигурация |
| SonusApplication.kt | Класс приложения |
| ApplicationComponent.kt | Компонента для DI |
| InteractorsModule.kt | Модуль интеракторов |
| MenuModule.kt | Модуль меню |
| NavigationModule.kt | Модуль навигации |
| NetworkModule.kt | Модуль для api |
| RepoModule.kt | Модуль репозиториев |
| ViewModule.kt | Модуль view |
| AccountInteractorImpl.kt | Соответствующая реализация |
| HomeInteractorImpl.kt | Соответствующая реализация |
| MusicInteractorImpl.kt | Соответствующая реализация |
| NewsInteractorImpl.kt | Соответствующая реализация |
| Launch.kt | Стартовый класс |
| CenterMenuPlacementView.kt | Центральное меню |
| CenterView.kt | Центральное меню (интерфейс) |
| MainView.kt | Хост окна |
| CustomState.kt | Состояние окна |
| InitialState.kt | Состояние окна |
| MaximizedState.kt | Состояние окна |
| State.kt | Состояние окна |
| WindowState.kt | Состояние окна |
| BottomMenuView.kt | Часть меню |
| LeftMenuView.kt | Часть меню |
| SideView.kt | Часть меню |
| TopMenuView.kt | Часть меню |
| MenuItem.kt | Элемент меню |
| MenuItemImpl.kt | Соответствующая реализация |
| Menu.kt | Интерфейс меню |
| MenuList.kt | Лист меню |
| Navigator.kt | Навигатор |
| NavigatorImpl.kt | Реализация навигатора |
| CenterPresenter.kt | Презентеры |
| MainPresenter.kt | Презентеры |
| AccountPresenter.kt | Презентеры |
| HomePresenter.kt | Презентеры |
| MusicPresenter.kt | Презентеры |
| NewsPresenter.kt | Презентеры |
| SectionPresenter.kt | Презентеры |
| Sections.kt | Секции |
| SectionView.kt | View секции |
| AccountView.kt | Соответствующее VIew |
| AccountViewImpl.kt | Соответствующая реализация |
| MessageWindow.kt | Соответствующее View |
| HomeView.kt | Соответствующее View |
| HomeViewImpl.kt | Соответствующее View |
| MusicView.kt | Соответствующее View |
| MusicViewImpl.kt | Соответствующая реализация |
| NetworkConverter.kt | Конвертер для combobox |
| NewsView.kt | Соответствующее View |
| NewsViewImpl.kt | Соответствующая реализация |
| Colors.kt | Объект с цветами |
| AccountViewStyles.kt | Стили |
| HomeViewStyles.kt | Стили |
| MusicViewStyles.kt | Стили |
| NewsViewStyles.kt | Стили |
| BottomViewStyles.kt | Стили |
| LeftMenuStyles.kt | Стили |
| MainWindowStyles.kt | Стили |
| TopViewStyles.kt | Стили |
| IconsProvider.kt | Провайдер ресурсов |
| ImageProvider.kt | Провайдер ресурсов |
| ResizeHelper.kt | Провайдер ресурсов |
| AccountInfo.kt | Обертки для моделей |
| AccountListWrapper.kt | Обертки для моделей |
| LoginBody.kt | Обертки для моделей |
| RegistrationResult.kt | Обертки для моделей |
| UserInfo.kt | Обертки для моделей |
| NewsList.kt | Обертки для моделей |
| ListOfPlaylists.kt | Обертки для моделей |
| PlaylistImageWrapper.kt | Обертки для моделей |
| PlaylistUpdateResult.kt | Обертки для моделей |
| TrackList.kt | Обертки для моделей |
| ApiService.kt | Сервис API |
| AccountRepositoryImpl.kt | Соответствующая реализация |
| NewsRepositoryImpl.kt | Соответствующая реализация |
| PlaylistRepositoryImpl.kt | Соответствующая реализация |
| TrackRepositoryImpl.kt | Соответствующая реализация |
| UtilsRepositoryImpl.kt | Соответствующая реализация |
| AccountInteractor.kt | Интерфейс интерактора |
| HomeInteractor.kt | Интерфейс интерактора |
| MusicInteractor.kt | Интерфейс интерактора |
| NewsInteractor.kt | Интерфейс интерактора |
| Account.kt | Модель |
| AccountRegistration.kt | Модель |
| GenerationParameter.kt | Модель |
| GenerationParams.kt | Модель |
| Network.kt | Модель |
| News.kt | Модель |
| Playlist.kt | Модель |
| Track.kt | Модель |
| UserInfo.kt | Модель |
| PlaylistImage.kt | Модель |
| AccountRepository.kt | Интерфейс репозитория |
| NewsRepository.kt | Интерфейс репозитория |
| PlaylistRepository.kt | Интерфейс репозитория |
| TrackRepository.kt | Интерфейс репозитория |
| UtilsRepository.kt | Интерфейс репозитория |

# ПРИЛОЖЕНИЕ 2

**Терминология**

**Интерактор (Interactor) –** единица (в том числе класс), реализующая бизнес-логику приложения, которая является связующей частью между слоями данных и представления

**Презентер (Presenter) –** единица (в том числе класс), реализующая связь (логику) между моделью и представлением

**Модель (Model) –** единица (в том числе класс), реализующая представление о мире в виде данных

**Представление (View) –** единица (в том числе класс), реализующая отображение данных из модели

**Репозиторий (Repository) –** единица (в том числе класс), служащая промежуточным звеном между различными источниками данных и слоями, использующими эти данные

**Плейлист (Playlist) –** некоторый упорядоченный набор треков (мелодий)

**Compose-стиль –** декларативный стиль создания интерфейсов, особенно широко развивающийся в наше время. Основное преимущество – композиции функций, создающих элементы интерфейса и удобство контроля над данными.

ЛИСТ РЕГИСТРАЦИИ ИЗМЕНЕНИЙ

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Изм. | Номера листов (страниц) | | | | Всего листов (страниц) в документе | № документа | Входящий № сопроводительного документа и дата | Подпись | Дата |
| измененных | замененных | новых | аннулированных |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |