**4.2.1. Проектирование архитектуры и описание состояний программного средства**

Большая часть приложения, которые используются для аудиоплееров, состоят из множества компонентов. Структура окон приложения описывается в отдельных файлах, отвечающих за графическую часть приложения. Такие файлы создаются с помощью специальных языков программирования, например XML. Это позволяет создавать графические интерфейсы адаптируемые под различные устройства.

Среднестатистический пользователь часто взаимодействует с графическим интерфейсов программы. Принимая это во внимание, нужно построить для приложения со сложной структурой адаптивный интерфейс, который сможет подстраиваться под запросы конечного пользователя.

В связи с этим важно создать архитектуру, которая позволит масштабировать приложение, сделает его надёжнее и повысит его производительность для слабых устройств. В связи с этим, выделю следующие важные аспекты архитектуры приложения:

* 1. Разделение задач и потребностей приложения;
* 2. Построение адаптивного UI
* 3. Построение архитектуры приложения, которая будет мало зависеть от конкретного устройства

На основе выше изложенных фактов, было решено использовать паттерн проектирования MVC. Model-View-Controller – шаблон проектирования, ключевой особенностей которого является разделение данных приложения и управляющей логики на следующие компоненты: модель, представление и контроллер. Данный шаблон проектирования позволяет производить изменение отдельных компоненты архитектуры без влияние на остальные. Диаграмма, на которой отображён принцип работы MVC, представлена на рисунке 4.1.

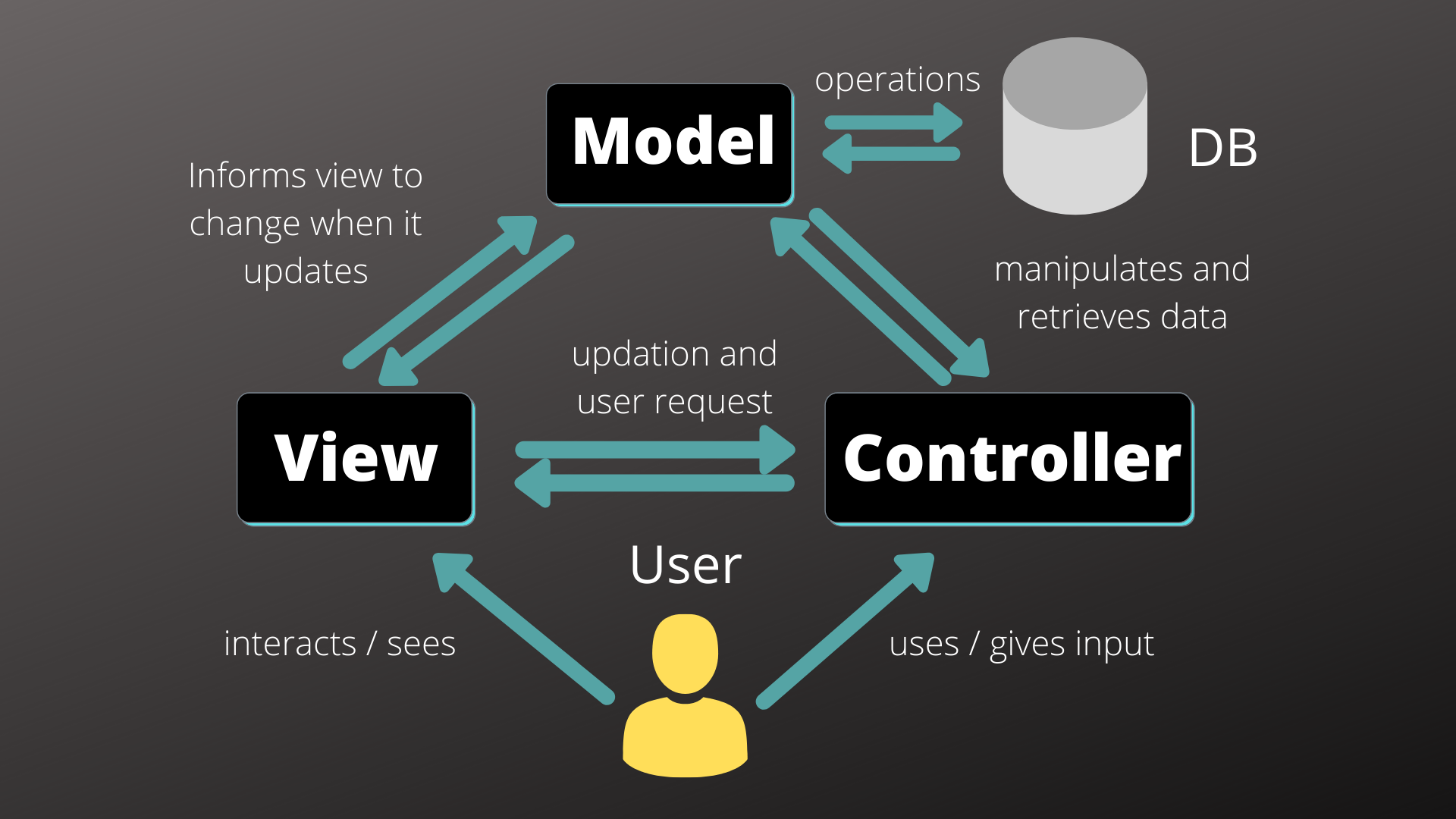


Рисунок 4.1

Как отображено на диаграмме, шаблон MVC состоит из следующих трёх слоёв: представление (View), модель (View), контроллер (Controller).

Модель хранит в себе данные, позволяет делать запросы в базу данных, позволяет обрабатывать эти данные. Тут находится основная логика нашей программы. Модель не знает, что происходит в контроллере и для чего оно обрабатывает этот запрос, который поступил к ней.

Контроллер принимает на себя роль посредника между представление и моделью. Контроллер, можно сказать, реагирует на действия пользователя с графическим интерфейсом, отправляя нужный запрос в модель.

Представление отвечает за отображение информации (визуализацию). Одни и те же данные могут представляться различными способами и в различных форматах. Контроллер направляет данные в представление, которое в свою очередь преобразует их в нужный нам графический формат.

Задача применения паттерна MVC состоит в разделении данных и логики программы от визуализации. За счет такого разделения повышается возможность повторного использования программного кода. Также упрощается и поддержка приложения: внесение изменений в логику приложения не отражаются на графическом интерфейсе, а изменения визуализации, например изменение фона главного экрана, не затрагивают логику программы.

**Диаграмма состояний**

Диаграмма состояний – поведенческая диаграмма, отображающая все возможные состояния в которых может находится программное средство, система или все объекты этой системы.

Опишем состояния программного средства «Trackr App» при помощи нотации UML.

На диаграмме состояний программного средства отображены следующие ключевые состояни:

- отображение экрана авторизации;

- отображение главного экрана;

- отображение окна выбранного проекта.

Прохождение данных состояний пользователями будет осуществляться практически при каждом использовании приложения, остальных – реже.

Рассмотрим все отображённые состояния последовательно в соответствии с их положением на диаграмме. Начало диаграммы отображается в виде закрашенного круга. Первым состоянием является отображение экрана авторизации.

Экран авторизации предлагает пользователю несколько опций. Из данного состояния, в зависимости от выбора пользователя программным средством может быть проведена авторизация пользователя или загружен экран с формой регистрации. Фрагмент диаграммы, на котором отображены данные состояния представлен на рисунке 4.2.

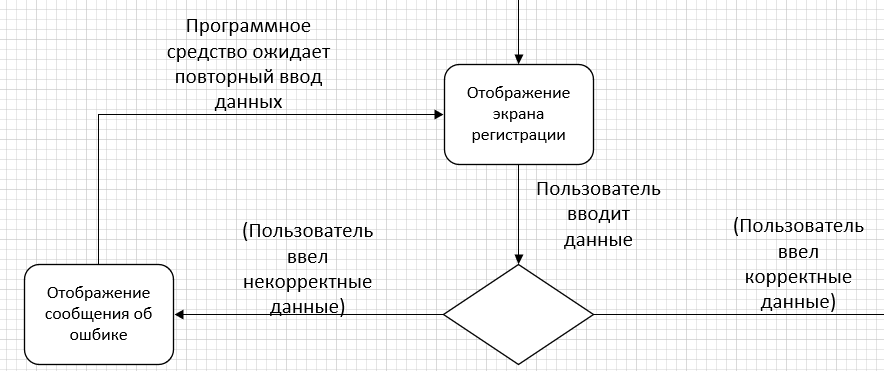
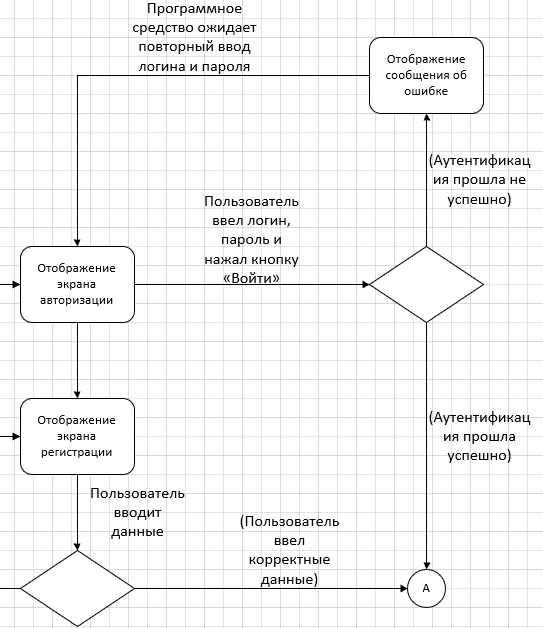


Рисунок 4.2

При выборе опции регистрации пользователем, будет запущен экран регистрации. Для прохождения регистрации необходимо ввести логин и пароль, который необходимо ввести дважды. Если пользователь будет успешно зарегистрирован, приложение снова отобразит экран авторизации. В противном случае, будет отображено уведомление об ошибке регистрации.

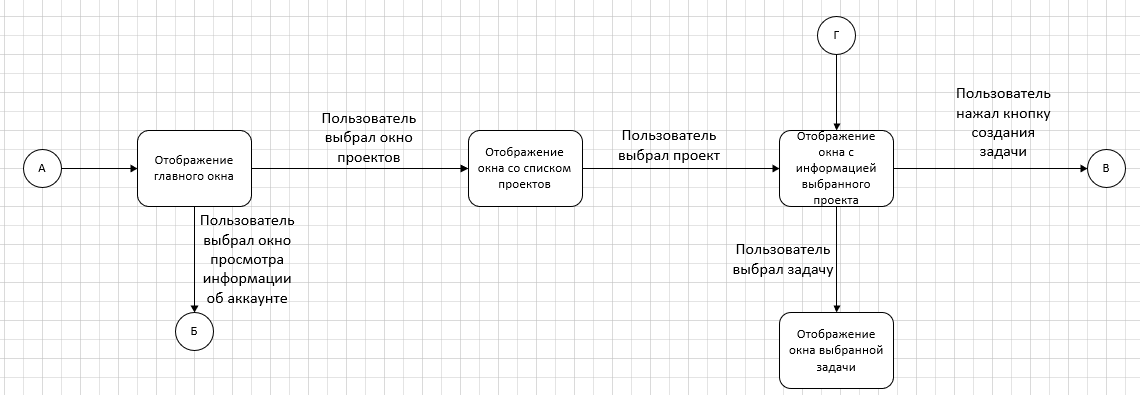
Процесс прохождения авторизации пользователем также требует ввода пары логина и пароля. Если пользователь был успешно авторизован, приложение будет отображать главный экран. В противном случае будет выведено сообщение об ошибке после которого пользователю будет предложено зарегистрироваться. Фрагмент диаграммы, на котором отображены данные состояния представлен на рисунке 4.3.

Рисунок 4.3



На главном экране пользователю предоставляется доступ к просмотру списка проектов. Фрагмент диаграммы, на котором отображены данные состояния представлен на рисунке 4.4.

Рисунок 4.4



Таким образом было произведено описание состояний программного средства при помощи нотации UML.

**4.2.2. Проектирование и разработка графического интерфейса**

Пользовательский интерфейс программного средства под операционную систему Android, созданного с целью упростить командную разработку проектов/продуктов состоит из следующих экранов:

* экран регистрации и авторизации пользователя;
* экран личного кабинета;
* экран смены пароля;
* экран со списком проектов;
* экран со списком задач;
* экран с информацией о задаче;
* экран с созданием задачи.

Многие компоненты часто используются на нескольких экранах, поэтому для них разработаны custom view.

Для экрана создания задачи была разработана разметка в файле xml, код которой представлен в листинге:

<?xml version="1.0" encoding="utf-8"?>  
<androidx.constraintlayout.widget.ConstraintLayout xmlns:android="http://schemas.android.com/apk/res/android"  
 xmlns:app="http://schemas.android.com/apk/res-auto"  
 xmlns:tools="http://schemas.android.com/tools"  
 android:layout\_width="match\_parent"  
 android:layout\_height="match\_parent"  
 tools:context=".AddTaskActivity">  
  
 <EditText  
 android:id="@+id/edtText"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 app:layout\_constraintTop\_toTopOf="parent"  
 app:layout\_constraintBottom\_toTopOf="@+id/btnAddTask"/>  
 <Button  
 android:id="@+id/btnAddTask"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:text="Добавить"  
 app:layout\_constraintTop\_toBottomOf="@id/edtText"/>  
  
</androidx.constraintlayout.widget.ConstraintLayout>

На рисунке 4.5 представлен экран создания задачи.

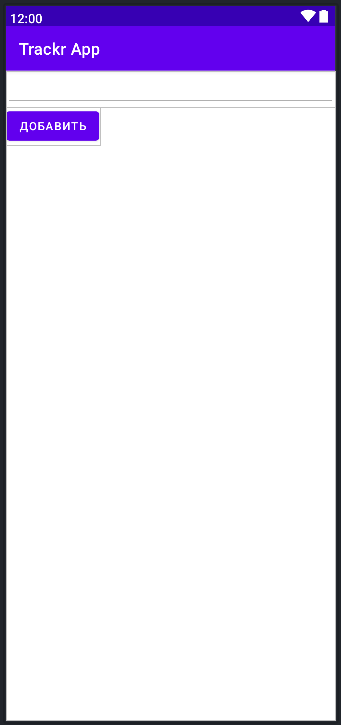


Рисунок 4.5

Таким образом, в рамках данного раздела была произведена разработка графического интерфейса программного средства для совместной разработки и учета задач. Было разработано мобильное приложение, которое создает задачи в определенные проекты и позволяет отслеживать их состояние. Для взаимодействия с другими программными комплексами реализован программный интерфейс по протоколу HTTP.