Федеральное государственное образовательное бюджетное
учреждение высшего образования
**«Финансовый университет
при Правительстве Российской Федерации»
(Финансовый университет)**

Департамент анализа данных и машинного обучения

Пояснительная записка к курсовой работе по дисциплине
«Современные технологии программирования»
на тему:

«Разработка информационной системы для кинотеатра с использованием Java, Spring и технологий мобильной разработки»

Выполнил:
Студент группы ПИ19-4
Козловский Алексей Дмитриевич

Научный руководитель:
профессор, доктор эконом. наук
Демин Игорь Святославович

Москва, 2021

ОГЛАВЛЕНИЕ

I. ВВЕДЕНИЕ

В крупных городах России и мира давно уже не встретишь кинотеатра с кассой. Всё потому, что большинство сетей кинотеатров автоматизировали свои задачи благодаря информационно-справочным системам.

К примеру, сети кинотеатров «КАРО Фильм» и «Кино Окко» давно используют в своей работе информационно-справочные стенды для удобной покупки билетов (см. прил. 1), а также сейчас пользуются популярностью такие сервисы как «Яндекс Афиша», «Афиша.ру», «Кинопоиск Афиша» и другие, которые являются информационным сервисом-посредником между кинотеатром и пользователем и позволяют последним с легкостью покупать билеты на сеансы с любого устройства или браузера.

Ставится вопрос: как устроены такие сервисы изнутри? Какие решения используются для их реализации? Какова актуальность внедрения таких информационно-справочных систем в кинотеатры?

Цель данной курсовой работы – проанализировать предметную область продажи билетов в кинотеатре, опираясь на проведенный анализ составить информационную модель задачи автоматизации и продемонстрировать возможный вариант её решения на примере серверного приложения на языке Java и с использованием фреймворка Spring, а также клиентского мобильного приложения под платформу Android OS, написанного на языке Java и с использованием современных технологий программирования.

Выполнение мною курсовой работы поспособствует приобретению и усовершенствованию навыков анализа предметной области, построения информационной модели задачи, проектирования реляционных баз данных, а также программирования на языке Java и разработки клиент-серверных приложений с технологией RESTful API.

II. ПОСТАНОВКА ЗАДАЧИ

В рамках курсовой работы поставлена задача продемонстрировать возможное клиент-серверное решение по автоматизации работы кинотеатров с использованием библиотеки Spring Boot для серверной части и также Android SDK и библиотек RxJava2, Retrofit2, Picasso для клиентской.

Разработанное серверное приложение должно обрабатывать запросы пользователей (клиентов), хранить и обрабатывать данные с помощью СУБД (системы управления базами данных, в данном случае MySQL).

Разработанное клиентское приложение должно иметь интуитивный и эргономичный дизайн экранов для удобного взаимодействия с пользователем, четкую и продуманную логику взаимодействия с серверной частью, а также быть оптимальным по затратности ресурсов и времени исполнения кода.

Разработанная система должна удовлетворять всем установленным требованиям, указанным в техническом задании, а именно:

1. В курсовом проекте должна быть разработана информационная модель предметной области, представленная в виде пользовательских классов и таблиц БД.

Взаимодействие с БД должно быть реализовано при помощи ORM.

2. Должно быть разработано несколько форм пользовательского интерфейса для клиента.

3. Разработчик самостоятельно определяет интерфейс программы и ее функциональность, однако для получения максимальной оценки приложение в обязательном порядке независимо от предметной области, указанной в задании, должно выполнять следующие операции:

- ✓ Отображать в таблице данные предметной области.
- ✓ Для информационной модели, основанной на БД, таблицы должны быть предварительно заполнены записями.

- ✓ Реализовать добавление в БД нового объекта, удаление объекта, редактирование объекта.
- ✓ Реализовать фильтрацию записей БД, удовлетворяющих введенному пользователем сложному критерию.
- ✓ Реализовать сортировку записей.
- ✓ Обновлять изменения источника данных в базе данных.
- ✓ После сохранения данных при запуске программы загрузить данные из БД.
- ✓ Создать пункт меню «Об авторе».
- ✓ Разработать несколько полезных пользователю функций для отображения статистических данных, например, средних, максимальных или минимальных значений, данных для построения гистограммы или графика.

4. Программа не должна завершаться аварийно: сообщения о некорректном вводе данных, противоречивых или недопустимых значениях данных, при отсутствии данных по функциональному запросу пользователя и других нештатных ситуациях отображать в окнах сообщений.

5. Программа должна иметь содержательные комментарии, которые могут генерировать автоматически составляемую документацию при помощи инструмента Javadoc.

III. ОПИСАНИЕ ПРЕДМЕТНОЙ ОБЛАСТИ

Предметной областью является продажа билетов в кинотеатре. И хотя клиентские части у многих сервисов разные (чаще всего это веб-сайты), серверная часть зачастую похожа. Она хранит все необходимые данные, такие как: фильмы в прокате, сеансы на них, чеки клиентов, их данные для доступа в аккаунт в хешированном виде, их персональную информацию, избранные фильмы, может хранить меню кинобара и прочую информацию.

В свою очередь клиент делает HTTP-запросы к серверу в синхронном или асинхронном режиме. Полученные с сервера данные он отображает на веб-странице или экране пользователя.

В этой курсовой работе в качестве клиента выступает мобильное приложение, так как смартфоны наиболее удобные и популярные устройства (75.5% населения планеты имеют мобильные устройства, из них – 62.2% предпочитают их настольным компьютерам).

IV. АКТУАЛЬНОСТЬ АВТОМАТИЗАЦИИ

Актуальность автоматизации обслуживания клиентов в кинотеатрах обуславливается потребностью тех к быстрому и удобному получению предоставляемых услуг таких как: билет на определенный сеанс или заказ из кинобара, избегая очереди на кассе. Также должна существовать возможность оплатить заказ посредством удобного способа оплаты: банковской картой или платежным сервисом (к примеру Google Play, Apple Pay и прочие).

Также автоматизация обслуживания поспособствует управляющему персоналу кинотеатра быстро и легко обмениваться информацией с пользователем, тем самым сэкономив его время и свои деньги на оплату труда билетных кассиров; и помимо всего прочего иметь наглядную и точную отчетность за конкретный период, статистику посещаемости и проч.

V. АЛГОРИТМИЧЕСКИЕ РЕШЕНИЯ

Серверное приложение курсового проекта реализовано с помощью сборщика проектов Gradle и библиотеки Spring Boot. Оно обрабатывает запросы клиента параллельно по следующему алгоритму.

- Контроллер подписан на события HandlerMapping и ждёт, пока ему делегируют запрос, после чего он обращается к нужному сервису за получением данных, в результате – возвращает ответ клиенту.
- Сервис получает запрос от контроллера и обращаются за данными к нужному репозиторию, после чего возвращает контроллеру ответ.
- Репозиторий получает запрос от сервиса и обращается за данными уже к сущностям, то есть напрямую к базе данных. После получения данных он возвращает ответ сервису.

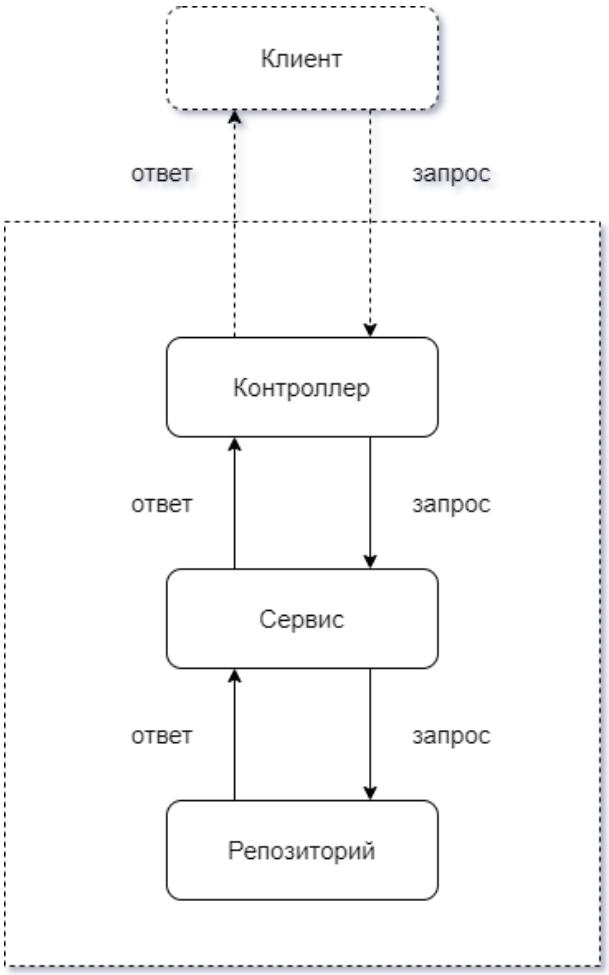


Рисунок 1. Иерархическая архитектура серверного приложения

Также на серверной стороне реализован алгоритм хеширования паролей при помощи алгоритма MD5 и метода «соления» паролей: к чистому паролю пользователя добавляется случайная комбинация из 20 байтов (соль) для предотвращения любой возможности декодирования пароля.

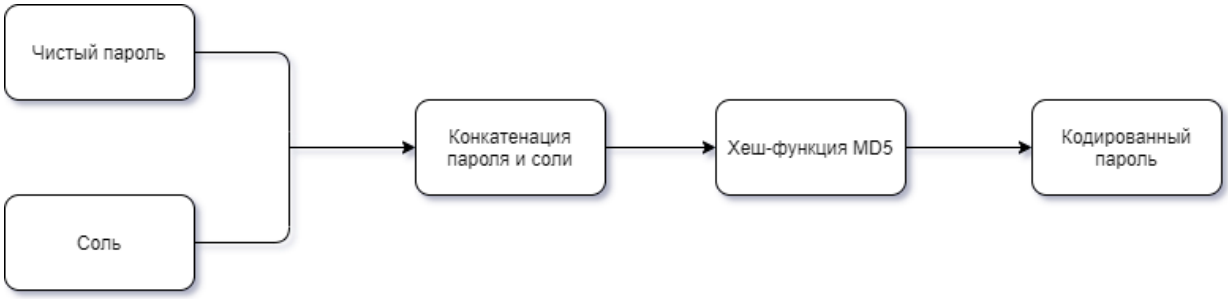


Рисунок 2. Алгоритм кодирования паролей пользователя

Клиентское приложение курсового проекта реализовано с помощью парадигмы асинхронного программирования и библиотеки RxJava2, что способствует более удобной и правильной обработке HTTP-запросов. В свою очередь, обработка HTTP-запросов реализована с помощью библиотеки Retrofit2, которая максимально удобна для разработчика тем, что позволяет хранить всю логику выполнения запросов в интерфейсе.

Приложение написано с использованием паттерна проектирования MVP (Model–View–Presenter). Суть заключается в том, чтобы создать абстракцию представления (того, что видит пользователь). Для этого необходимо выделить интерфейс представления с определенным набором свойств и методов. Presenter, в свою очередь, получает ссылку на реализацию интерфейса, подписывается на события представления и по запросу изменяет модель. Вся работа с моделью выполняется в фоновых потоках, чтобы не нагружать главный поток пользовательского интерфейса (Main Thread).

Для работы с пользовательскими изображениями на стороне клиента используется библиотека Picasso, позволяющая загружать изображения с любого интернет-ресурса быстро и асинхронно.

VI. ОПИСАНИЕ ИНТЕРФЕЙСА ПРОГРАММЫ

В серверном приложении интерфейс отсутствует, так как представление данных и обработка событий пользователя не входит в его основные задачи.

Пользовательский интерфейс клиентского приложения эргономично спроектирован и представлен в нескольких основных типах, а именно:

1. Отдельные экраны (активности)
   - заглушечный экран
   - стартовый экран
   - главный экран с меню
   - экран регистрации
   - экран авторизации
   - экран подробной информации о фильме
   - экран настроек
   - экран выбора мест на сеанс
   - экран выбор дополнительных товаров к заказу
2. Фрагменты
   - фрагмент фильмов, идущих в кинотеатре
   - фрагмент истории билетов (чеков)
   - фрагмент пользовательского профиля
3. Диалоговые окна с сообщениями об ошибках

Ниже подробно описаны экраны пользовательского интерфейса: их назначение, порядок перехода и возвращения.

Заглушечный экран

Заглушечный экран отображается всегда при запуске приложения. За это отвечает специальный фильтр у активности в манифесте:

```
<intent-filter>
    <action android: name="android.intent.action.MAIN" />
    <category android: name="android.intent.category.LAUNCHER" />
</intent-filter>
```

На заглушечном экране отображается логотип приложения без каких-либо дополнительных элементов интерфейса. В то время, как пользователь видит заглушечный экран, происходит загрузка приложения (его ресурсов и классов) системой Android на протяжении примерно одной секунды (около 2 секунд при холодном запуске).

Использование заглушечного экрана – довольно распространенная практика разработки мобильных приложений, чтобы представить пользователю на время системной загрузки приятную глазу картинку вместо белого экрана, который система рисует по умолчанию.

После системной загрузки ресурсов и классов в контексте заглушечного экрана проверяется факт авторизации пользователя (значение Boolean переменной «isAuthorized», которая хранится в локальном хранилище). Если пользователь не авторизован, запускается стартовый экран.

Стартовый экран

Стартовый экран отображается после заглушечного, если пользователь не авторизован. На нём у пользователя запрашивается зарегистрироваться или авторизоваться, без чего главный экран приложения не может быть открыт не при каких условиях. Также на стартовом экране снизу расположены кликабельные ссылки на Telegram и GitHub автора.

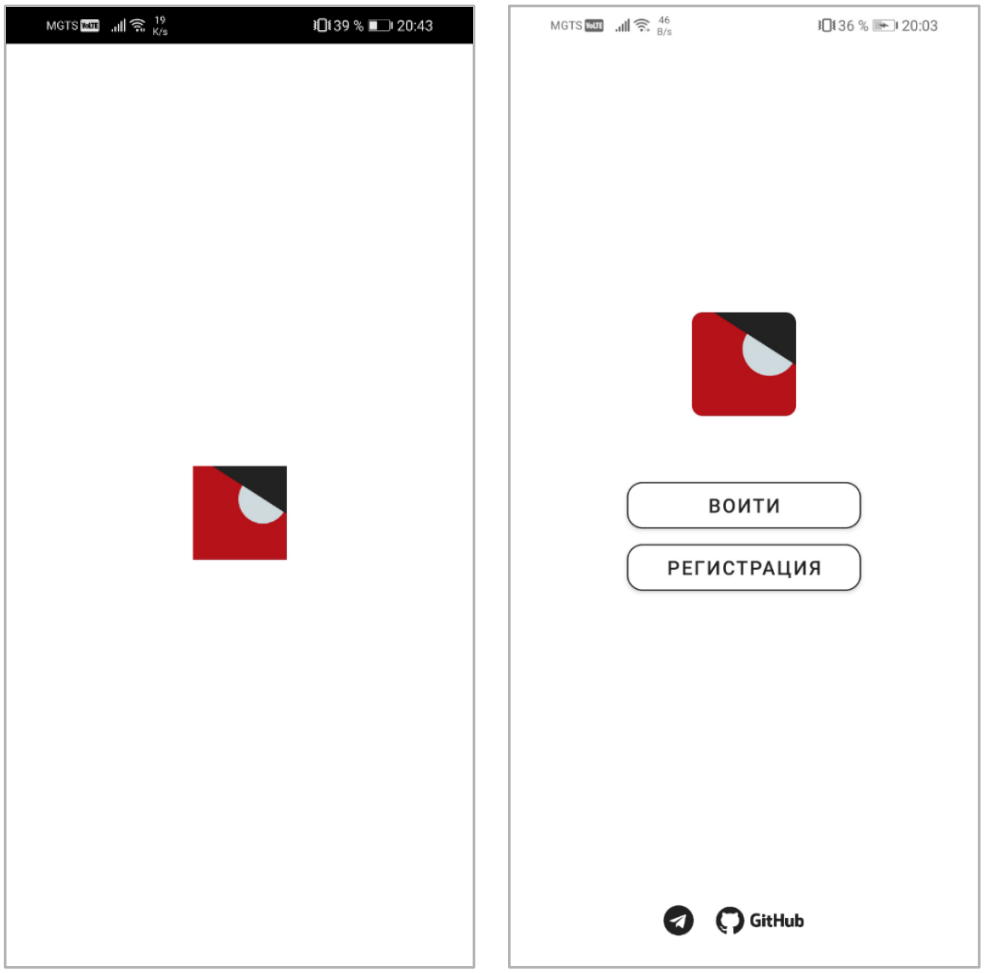


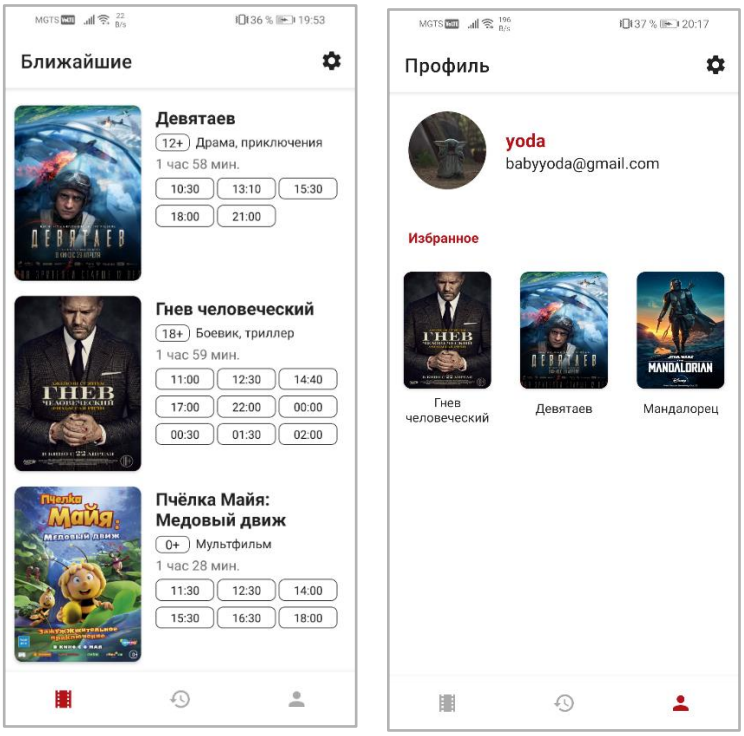
Рисунок 3–4. Заглушечный экран (слева) и стартовый (справа)

Главный экран с меню

Главный экран запускается после заглушечного, если пользователь авторизован в приложении или после экранов регистрации и авторизации, если он успешно зарегистрировался или авторизовался.

Главный экран является неким контейнером для фрагментов с фильмами, историей заказов и профилем пользователя, между которыми можно переключаться посредством нижнего меню. Также из главного экрана можно попасть в экран настроек, а слева сверху отображается название выбранного раздела или же фрагмента.

Фрагменты на главном экране сохраняют свое состояние, то есть не перерисовываются заново при выборе в меню, однако пользователь может обновить данные с сервера при помощи рефрешера.



Рисунки 5–7. Главный экран с фрагментом «Фильмы в прокате» (слева) и главный экран с фрагментом «Профиль» (справа)

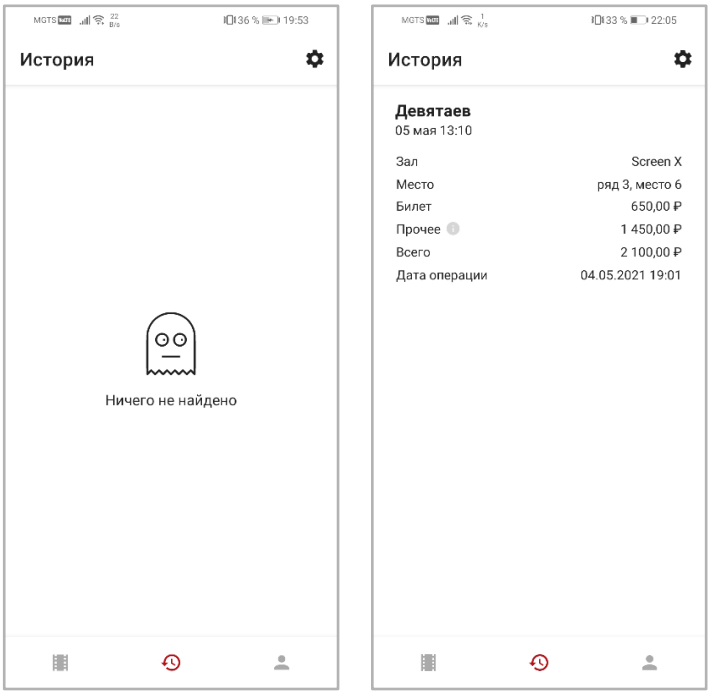


Рисунок 8. Главный экран с фрагментом «История»

Экраны регистрации и авторизации

На экране регистрации у пользователя запрашивается электронная почта, псевдоним и пароль для создания нового аккаунта. Кнопка «Вперед» изначально выключена и недоступна для нажатия. Как только во всех полях ввода появляется хотя бы один символ, не считая пробела, она становится включенной и доступной для нажатия.

При нажатии на кнопку «Вперед» проверяется корректность ввода.

Требования к регистрационным данным:

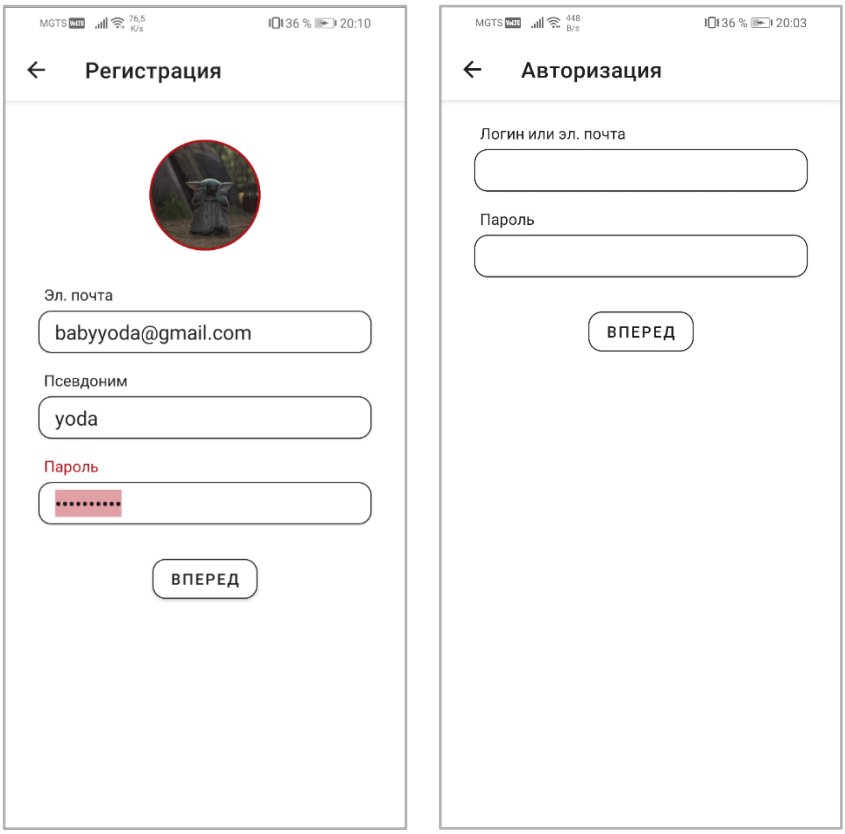
- Эл. почта должна соответствовать определенном паттерну.
- Псевдоним не должен начинаться с цифры и может содержать только буквы латинского алфавита в любом регистре.
- Пароль должен содержать минимум 8 символов, включая хотя бы один символ верхнего регистра и один спец. символ (#, ?, %, &).

В случае некорректного ввода поле становится красным, что говорит об ошибке, и регистрация не продолжается. Также пользователь может выбрать изображение профиля, для чего требуется дать приложению разрешение на доступ к хранилищу и камере. В случае если он не выберет изображение, оно будет сгенерировано по шаблону фон + первый символ псевдонима.

Если же ввод корректен, генерируется «соль» и появляется индикатор прогресса на время отправки клиентом данных на сервер, где пароль и соль хешируются по алгоритму, если ответ от сервера положительный – запускается главный экран, иначе – пользователя может увидеть ошибку.

Экран авторизации служит для авторизации пользователя в системе. У пользователя запрашивается ввод логина или электронной почты и пароля. Пароль отправляется на сервер для сравнения с уже имеющимся хешированным паролем, сохраненным в базе данных. В случае совпадения паролей возвращается сущность пользователя с указанным логином или эл.

почтой, в другом случае – пользователь может увидеть у себя на экране ошибку о некорректности введенных данных.



Рисунки 9–10. Экран регистрации (слева) и авторизации (справа)

Экран подробной информации о фильме

Экран отображает подробную информацию о фильме, которая может быть интересна пользователю: возрастной рейтинг, категории, длительность, описание, режиссеров, актерский состав и ближайшие сеансы.

При нажатии на «звездочку» фильм будет добавлен в «Избранное».

При нажатии на стрелочку «назад» пользователь вернется на главный экран, и будет установлен последний отрисованный фрагмент.

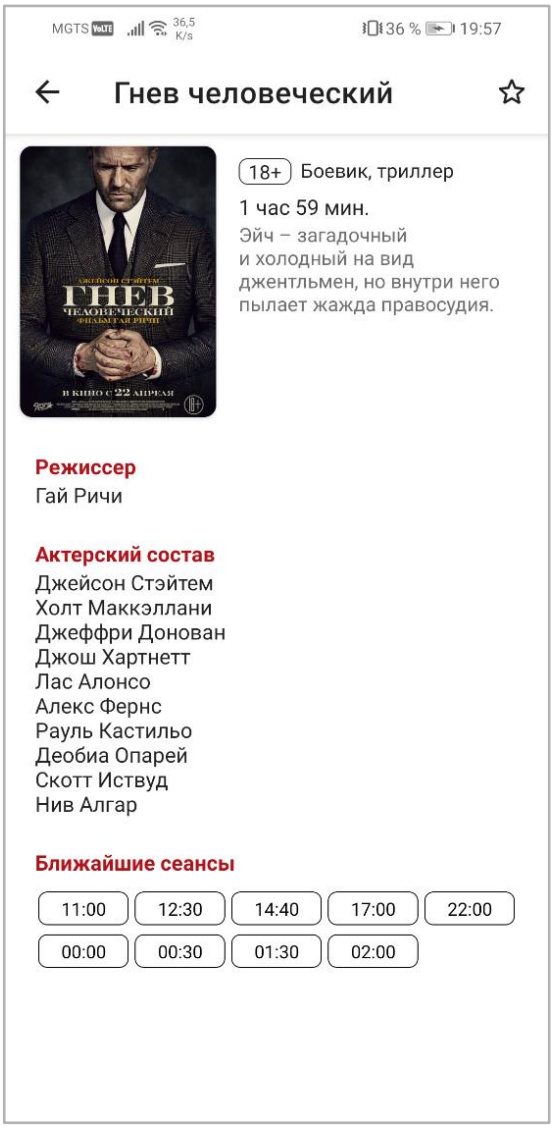


Рисунок 11. Экран подробной информации о фильме

Экран настроек

Экран настроек позволяет пользователю изменить данные, указанные пре регистрации и указать дополнительные данные. При нажатии на элемент вызывается диалоговое окно (рисунок 13), в котором при нажатии на «Ок» выбранный параметр будет изменен. При нажатии на «Отмена» диалоговое окно будет закрыто без изменений параметра.

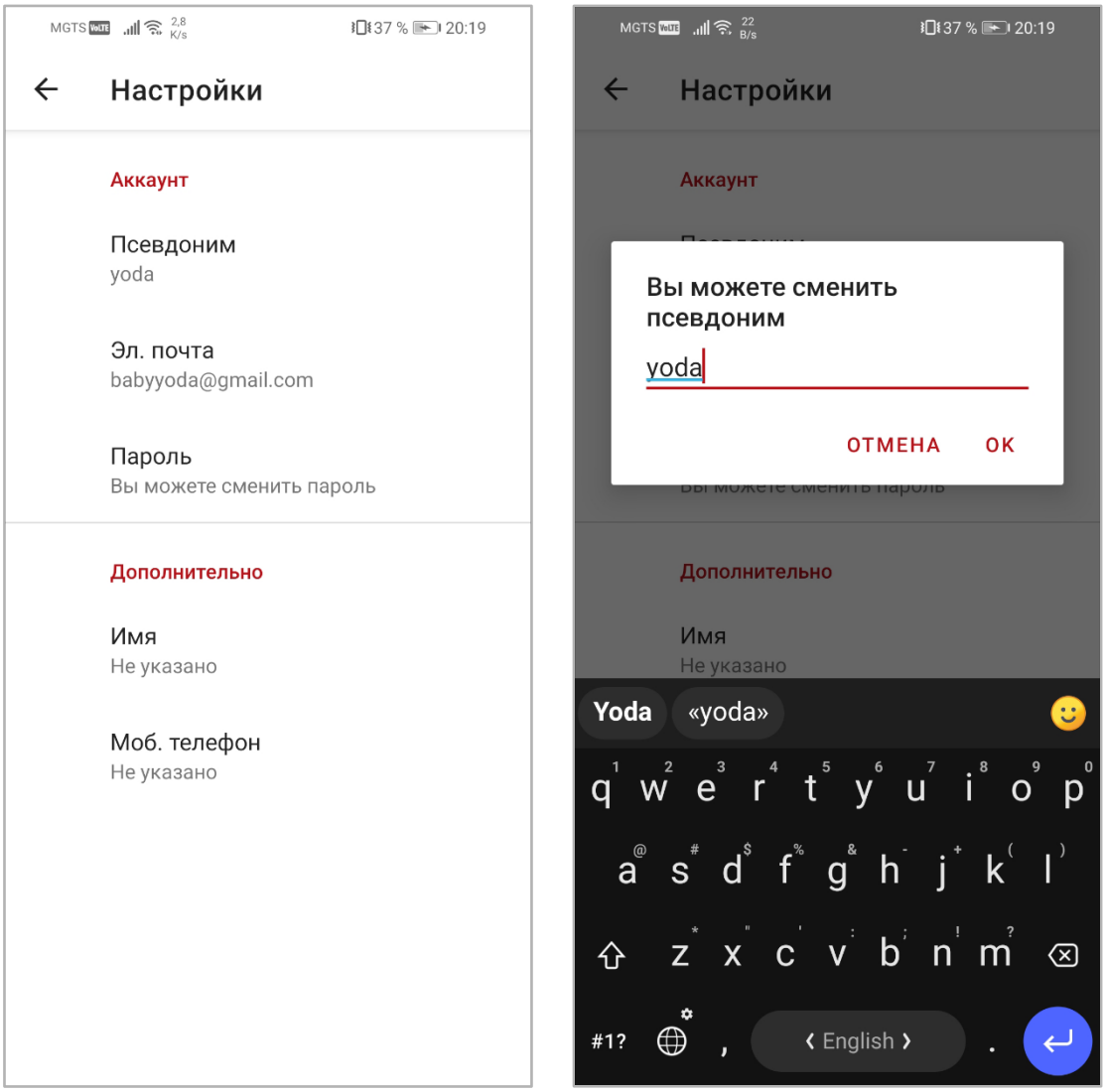


Рисунок 12–13. Экран настроек (слева) и экран настроек после выбора элемента меню «Псевдоним» (справа)

Экран выбора мест на сеанс

Экран выбора мест на сеанс открывается при выборе пользователем сеанса во фрагменте «В прокате» или на экране «Подробной информации о фильме». Данный экран отображает информацию о сеансе, такую как: время сеанса, зал, возрастное ограничение и язык озвучки.

Ниже при помощи виджета «GridView» показано расположение мест относительно экрана. Занятые места отмечены серым цветом, свободные – синим, выбранные пользователем – зеленым. Также ниже отображена

стоимость свободных мест. Чтобы выбрать места на сеанс необходимо нажать на свободное место, а затем подтвердить выбор, нажав на кнопку «Купить». Максимальное количество мест в одном заказе – 5.

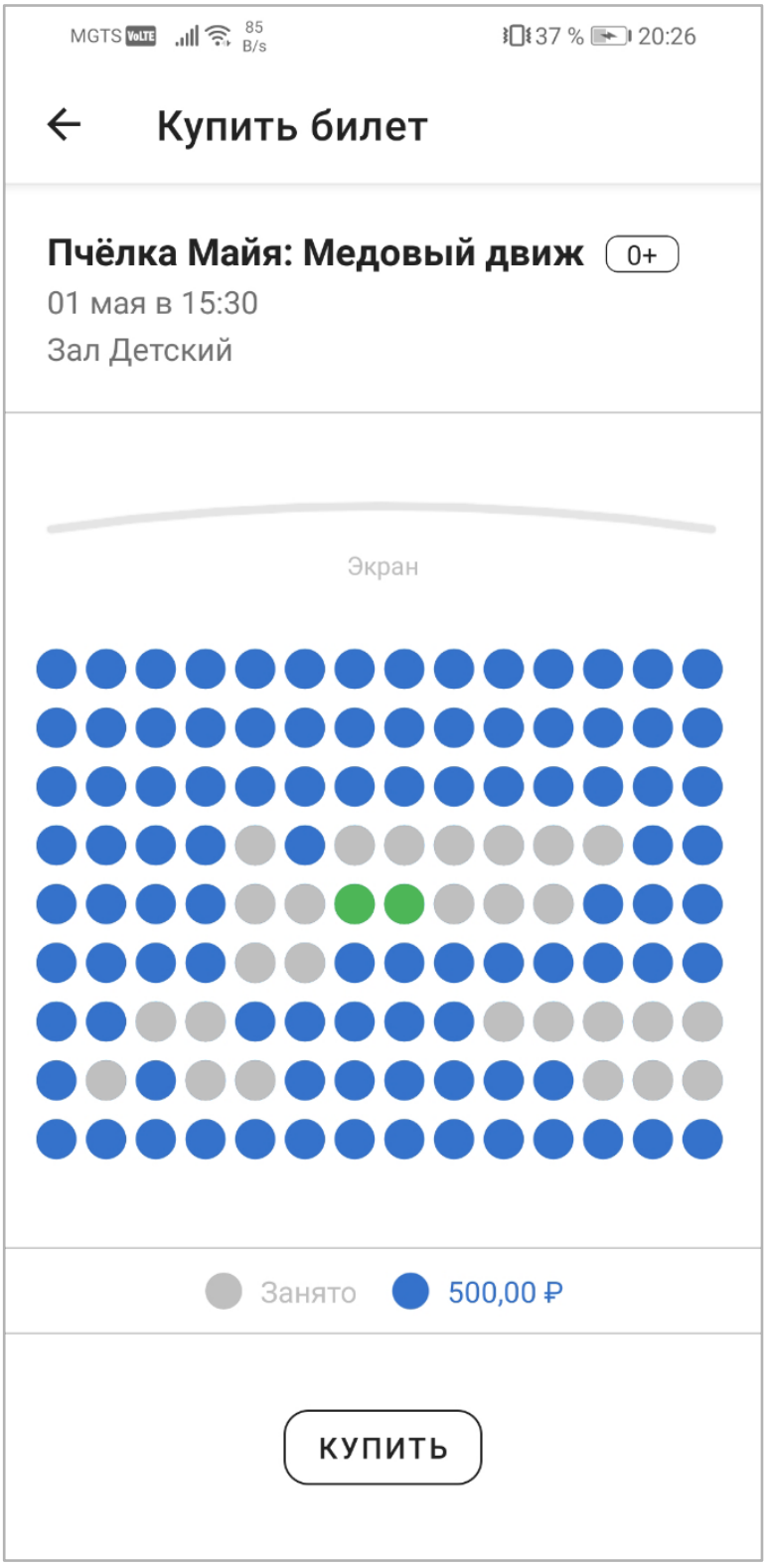


Рисунок 14. Экран выбора мест на сеанс

Экран кинобара

Экран кинобара открывается после нажатия пользователем кнопки «Продолжить» на экране выбора мест на сеанс. Сверху с помощью виджета «TabLayout» отображаются доступные категории закусок и напитков из кинобара. После выборы категории происходит фильтрация загруженного списка всех товаров из кинобара по заданном критерию.

Пользователь может выбрать до 4 позиций каждого товара, которые есть в наличии. Пользователь может не выбрать ни одной позиции и сразу нажать на кнопку «Перейти к оплате».

VII. СОСТАВ ПРИЛОЖЕНИЯ

Приложение состоит из серверной части, которая занимается хранением и обработкой информации, базы данных MySQL и клиентской части.

Подробный состав отдельных частей приложения:

1. Серверная часть

В состав сервера входят следующие компоненты:

- Spring Boot – библиотека для построения REST решений.
- MySQL Driver – драйвер для преобразования сущностей в таблицы базы данных MySQL и работы с ними посредством библиотеки JpaRepository.
- Lombok – библиотека для замены конструкторов, геттеров и сеттеров на удобные аннотации.
- Gradle – сборщик проектов.

Настройки сервера отображены в файле application.yml, который находится в корневой директории проекта.

2. База данных

Хранение данных на сервере реализовано при помощи СУБД MySQL. Каждая сущность имеет обязательный искусственный **уникальный идентификатор**, с помощью которого можно однозначно идентифицировать объект сущности. Также сущности, в зависимости от своего типа, имеют архитектуру, необходимую для хранения определенной информации.

Схема разработанных сущностей представлена на рисунке 14.

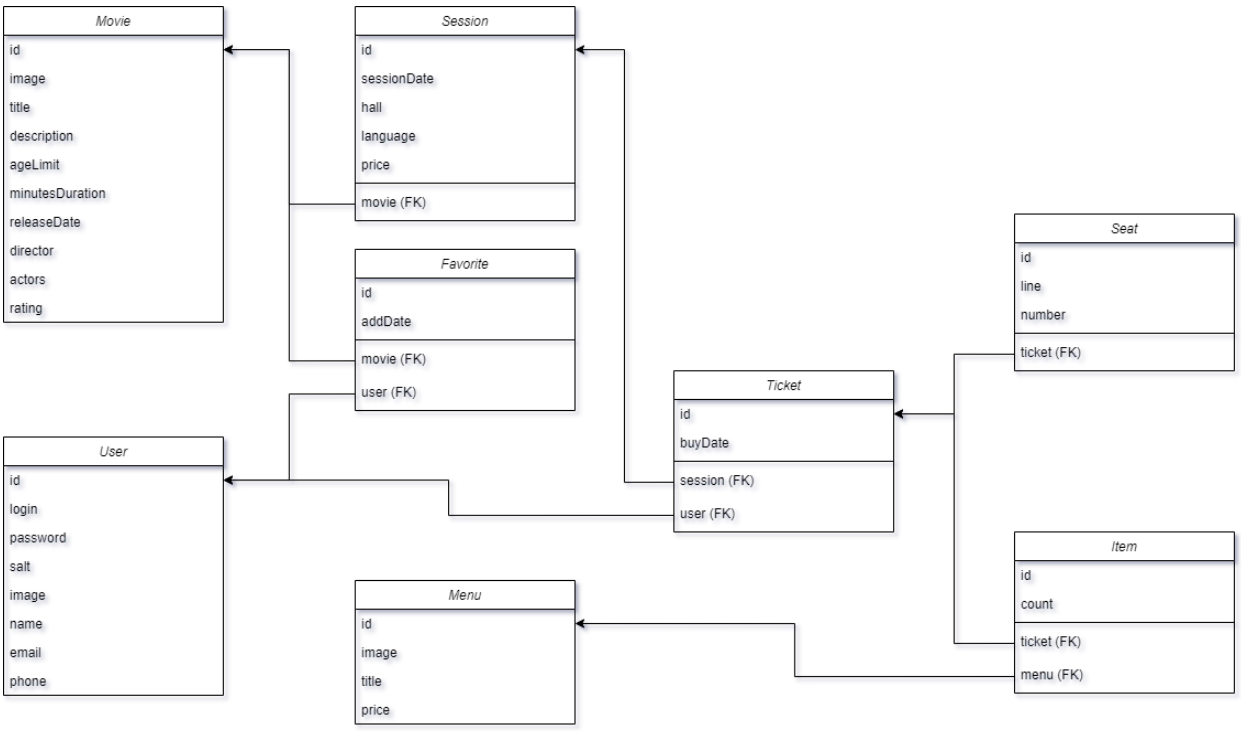


Рисунок 15. Схема разработанных сущностей

К примеру, сущность «Movie» содержит информацию о фильме, который идет или когда-либо шел в прокате:

- название фильма – столбец title
- описание фильма – столбец description
- ссылка на постер фильма – столбец image
- возрастное ограничение – столбец age_limit
- продолжительность (в мин.) – столбец minutes_duration
- дата релиза – столбец release_date
- режиссер(ы) – столбец director
- актеры – столбец actors
- рейтинг фильма – столбец rating

Сущность «Session» имеет связь «многие к одному» с сущностью «Movie», а также хранит информацию о сеансе на фильм, такую как:

- дата сеанса – столбец session_date
- название зала сеанса – столбец hall

– язык сеанса – столбец language
– цена билета на сеанс – столбец price
– внешний ключ фильма – столбец movie_id

Сущность «User» хранит данные пользователя, такие как:

– псевдоним – столбец login
– пароль в захэшированном виде – столбец password
– соль (комбинация из 12 случайных символов для невозможности декодирования пароля пользователя) – столбец salt
– изображение профиля пользователя – столбец image
– полное имя пользователя – столбец name
– эл. почта пользователя – столбец email
– номер телефона пользователя – столбец phone

Сущность «Favorite» является решением связи «многие ко многим» для сущностей «Movie» и «User». Она хранит информацию об избранных фильмах конкретного пользователя, такую как:

– дата добавления в избранное – столбец add_date
– внешний ключ фильма – столбец movie_id
– внешний ключ пользователя – столбец user_id

Сущность «Menu» хранит информацию о товаре в кинобаре:

– изображения товара – столбец image
– название товара – столбец title
– стоимость товара – столбец price

Сущность «Ticket» хранит данные об одном чеке пользователя:

– дата приобретения билета – столбец buy_date
– выбранный ряд – столбец line
– выбранное место – столбец seat

– внешний ключ сеанса – столбец session_id
– внешний ключ пользователя – столбец user_id

Сущность «Item» является решением связи «многие ко многим» для сущностей «Ticket» и «Menu» и хранит информацию о том, какие товары из кинобара были добавлены к приобретенному пользователем билету:

– количество товаров – столбец count
– внешний ключ билета – столбец ticket_id
– внешний ключ товара – столбец menu_id

Сущность «Seat» хранит данные о том, какие места заняты в кинозале:

– ряд – столбец line
– место – столбец number

3. Клиентская часть

В состав клиентской части входят следующие компоненты:

– Пакеты Android AppCompat и Material для разметки UI.
– Пакеты AndroidX Lifecycle LiveData и ViewModel для использования удобного типа данных LiveData.
– Picasso – библиотека для работы с изображениями.
– Retrofit2 – библиотека для работы с HTTP-запросами.
– Retrofit2 Gson Converter – библиотека, позволяющая Retrofit2 автоматически преобразовывать полученный ответ от сервера в сущность.
– ReactiveX RxJava2 – библиотека для работы с асинхронностью и параллелизмом в фоновых потоках.
– ReactiveX RxAndroid – дополнение для RxJava2, предоставляющее возможность перенаправления данных в главный поток.

VIII. НАЗНАЧЕНИЕ И СОСТАВ КЛАССОВ ПРОГРАММЫ

Серверная часть

На сервере реализовано 32 класса для его корректной и стабильной работы. Их можно разделить на некоторые типы:

- Классы-сущности (8 шт.) – классы, экземпляры которых принимаются сервером в запросе и отдаются в ответе.
  - Favorite – класс для сущности «Избранное».
  - Item – класс для сущности «Элемент».
  - Menu – класс для сущности «Меню».
  - Movie – класс для сущности «Фильм».
  - Session – класс для сущности «Сеанс».
  - Ticket – класс для сущности «Билет».
  - UploadResponse – класс, который является сущностью ответа пользователю о загруженном изображении.
  - User – класс для сущности «Пользователь».

- Контроллеры (7 шт.) – классы, получающие запросы от клиента по указанному маппингу и возвращающие клиенту ответ.
  - AuthController – контроллер, который принимает запросы на регистрацию, авторизацию и получение информации о пользователе.
  - BarController – контроллер, который принимает запросы по получению всех доступных товаров в кинобаре.
  - FavoritesController – контроллер, который принимает запросы по получению избранных фильмов пользователя, добавлению нового избранного фильма и удалению существующего.
  - MoviesController – контроллер, который принимает запросы по получению всех доступных фильмов в прокате (т. е. те, где текущая дата находится между датой начала проката и датой конца).

- SessionsController – контроллер, принимающий запросы по получению всех активных сеансов на конкретный фильм и информации о сеансе по его ID для проверки актуальности сеанса на экране выбора мест.
- TicketsController – контроллер, принимающий запрос по получению всех билетов пользователя для отображения его истории и всех билетов на сеанс для расчёта свободных мест на сеанс.
- UserImagesController – контроллер, принимающий запросы по загрузке на сервер изображения и его скачиванию.

– Репозитории (6 шт.) – классы, взаимодействующие с сущностями (или таблицами в базе данных).
  - AuthRepo – интерфейс, работающий с сущностью «Пользователь», с методом поиска по логину или эл. почте.
  - BarRepo – интерфейс, работающий с сущностью «Меню».
  - FavoriteRepo – интерфейс, работающий с сущностью «Избранное», с методом поиска избранного по ID пользователя.
  - MovieRepo – интерфейс, работающий с сущностью «Фильмы».
  - SessionRepo – интерфейс, работающий с сущностью «Сеанс», с методами поиска по ID фильма и проверки валидности сеанса.
  - TicketRepo – интерфейс, работающий с сущностью «Билет», с методами поиска билетов по ID пользователя и ID сеанса.
– Сервисы (7 шт.) – классы, взаимодействующие с репозиториями и выполняющие основную логику проверок.
  - AuthService – сервис, работающий с сущностью «Пользователь», с доп. методом хеширования пароля с «солением».
  - BarService – сервис, работающий с сущностью «Меню». FavoriteService – сервис, работающий с сущностью «Избранное».
  - MoviesService – сервис, работающий с сущностью «Фильм».
  - SessionService – сервис, работающий с сущностью «Сеанс».
  - TicketService – сервис, работающий с сущностью «Билет».

- UserImagesService – сервис, отвечающий за сохранение изображений на сервере и их скачивание.

– Классы исключений (2 шт.)
  - NotFoundException – исключение, которое возвращается в случае если подходящий ответ на запрос не был найден.
  - StorageException – исключение, которое возвращается в случае если произошла ошибка при загрузке изображения.

– Классы конфигурации (1 шт.)
  - FileStorageProperties – класс, устанавливающий путь к директории, куда будут сохраняться изображения и устанавливающий прочие конфигурации хранилища.

– Классы-лаунчеры (1 шт.)
  - ProdApplication – класс, запускаемый при старте программы.

Клиентская часть

– Классы-модели (8 шт.) – аналоги сущностей на стороне сервера, которые клиент отправляет в теле запроса и получает в теле ответа.
  - Favorite – модель по сущности «Избранное».
  - Item – модель по сущности «Элемент».
  - Menu – модель по сущности «Меню».
  - Movie – модель по сущности «Фильм».
  - Session – модель по сущности «Сеанс».
  - Ticket – модель по сущности «Билет».
  - UploadResponse – класс, который является моделью по ответу пользователю о загруженном изображении.
  - User – модель по сущности «Пользователь».

– Классы-утилиты (5 шт.) – классы, в которых вынесена повторяющаяся статичная логика приложения.
    - AuthUtil – утилита, позволяющая удобнее работать с локальным хранилищем и получать данные о состоянии пользователя.
    - BitmapUtil – утилита, позволяющая преобразовать первую букву псевдонима и фон в системное изображение профиля.

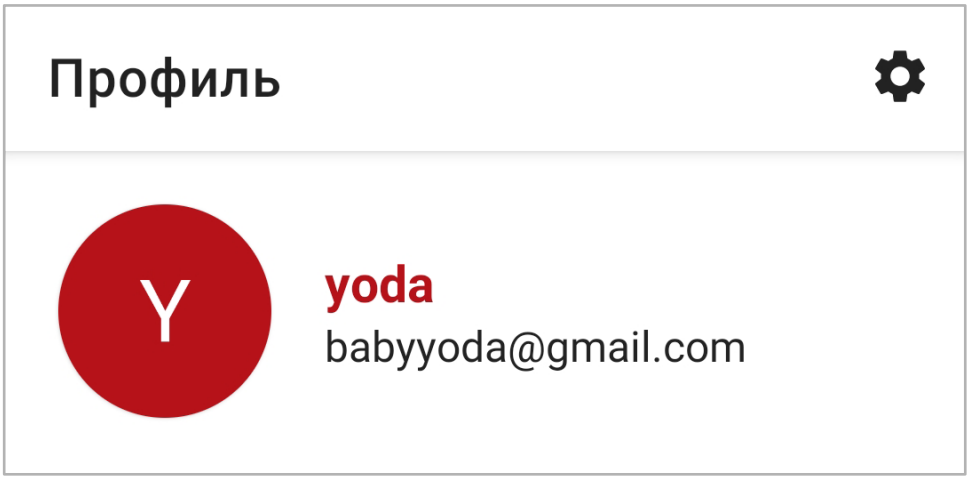


Рисунок 16. Системное изображение профиля

    - Converter – утилита, позволяющая преобразовать значение типа Double в строку денежного формата. Используется для красивого отображения сумм и стоимостей.
    - DurationUtil – утилита, позволяющая преобразовывать значение длительности в минутах в формат N часа (-ов) M мин.
    - PathUtil – утилита, позволяющая узнать тип файла и его путь.

– Репозитории (1 шт.)
    - ResponseRepository – класс, хранящий в себе все методы обращения к серверу за данными.

– Классы исключений (1 шт.)

- NotFoundException – модель ошибки, объявленной на серверной части, для получения типа ошибки оператором instanceof.

– Классы API (2 шт.)
  - MovietonApi – интерфейс, который содержит в себе все методы запросов к серверу для библиотеки Retrofit2.
  - MovietonService – сервис, конфигурирующий изначальное состояние Retrofit2 и позволяющий получить образ интерфейса MovietonApi для обращения к нему.

– Активности (9 шт.) – классы, контролирующие состояние макетов конкретного экрана и работающие с их презентерами.
  - AuthActivity – класс, работающий с экраном авторизации.
  - BarActivity – класс, работающий с экраном кинобара.
  - MainActivity – класс, работающий с главным экраном.
  - MovieActivity – класс, работающий с экраном подробной информации о фильме.
  - RegisterActivity – класс, работающий с экраном регистрации.
  - SessionActivity – класс, работающий с экраном сеанса.
  - SettingsActivity – класс, работающий с экраном настроек.
  - SplashActivity – класс, работающий с заглушечным экраном.
  - StartActivity – класс, работающий со стартовым экраном.

– Презентеры (7 шт.) – классы, взаимодействующие с моделью и обновляющие интерфейс, взаимодействуя с его представлением.
  - AuthPresenter – презентер активности AuthActivity.
  - BarPresenter – презентер активности BarActivity.
  - MainPresenter – презентер активности MainActivity.
  - MoviePresenter – презентер активности MovieActivity.
  - RegisterPresenter – презентер активности RegisterActivity.

- SessionPresenter – презентер активности SessionActivity.
- StartPresenter – презентер активности StartActivity.

– Представления (10 шт.) – интерфейсы, являющиеся представлениями экрана, с которыми работают презентеры.
  - AuthView – представление экрана авторизации.
  - BarView – представление экрана кинобара.
  - HistoryView – представление фрагмента истории.
  - MoviesView – представление фрагмента фильмов в прокате.
  - ProfileView – представление фрагмента профиля.
  - MainView – представление главного экрана.
  - MovieView – представление экрана подробной информации о фильме.
  - RegisterView – представление экрана регистрации.
  - SessionView – представление экрана сеанса.
  - StartView – представление стартового экрана.

– Адаптеры (6 шт.) – классы для динамической работы с виджетами RecyclerView и GridView и их элементами.
  - MenuAdapter – адаптер для списка товаров на экране кинобара.
  - HistoryAdapter – адаптер для списка чеков на фрагменте истории.
  - MoviesAdapter – адаптер для списка фильмов во фрагменте фильмов в прокате.
  - SessionsAdapter – адаптер для сетки сеансов у каждого фильма.
  - FavoritesAdapter – адаптер для списка избранного во фрагменте профиля.
  - SeatsAdapter – адаптер для сетки свободных мест на экране сеанса.

– Фрагменты (3 шт.) – классы, контролирующие состояние фрагментов и взаимодействующие с их моделью представления (модель MVVM).

    - HistoryFragment – класс фрагмента истории.
    - ProfileFragment – класс фрагмента профиля.
    - MoviesFragment – класс фрагмента фильмов в прокате.
- Модели представлений (3 шт.) – классы, которые при изменении модели обновляют пользовательский интерфейс фрагмента.
    - HistoryViewModel – модель представления фрагмента истории.
    - ProfileViewModel – модель представления фрагмента профиля.
    - MoviesViewModel – модель представления фрагмента фильмов в прокате.

IX. ЗАКЛЮЧЕНИЕ

В ходе выполнения мною курсовой работы была проанализирована предметная область продажи билетов в кинотеатрах, поставлена задача автоматизировать данную предметную область и продемонстрировано одно из возможных решений поставленной задачи – разработана информационно-справочная система по архитектуре сервер-клиент-база данных на языке программирования Java и с использование Spring Boot и Android SDK.

Созданное мною решение полностью готово к использованию и соответствует всем установленным техническим заданием требованиям:

1. Разработана информационная модель предметной области, представленная в виде пользовательских классов и таблиц БД MySQL. Взаимодействие с БД реализовано при помощи ORM.

2. Разработано 9 экранов и 3 фрагмента пользовательского интерфейса.

3. Данные представлены в виде записей в таблицах базы данных MySQL, которая предварительно заполнена данными.

4. Реализовано добавление в БД новых объектов: билетов, избранного; удаление: билетов, избранного и редактирование пользовательской информации.

5. Реализована сортировка и фильтрация записей.

6. Реализовано обновление источника данных.

7. Создан пункт меню об авторе в виде ссылок на страницы автора и его портфолио на стартовом экране.

8. Программа не завершается аварийно, все исключения сервера и ошибки ввода обрабатываются, пользователь уведомляется.

9. Программа содержит содержательные комментарии Javadoc.

X. СПИСОК ЛИТЕРАТУРЫ

1. Уоллс К. Spring в действии. – М.: ДМК Пресс, 2013. – 752 с.

2. Лонг Джош, Бастани Кеннет Java в облаке. Spring Boot, Spring Cloud, Cloud Foundry. – СПБ.: Питер, 2019. – 624 с.

3. Филлипс Билл, Стюарт Крис, Марсикано Кристин, Гарднер Брайан Android. Программирование для профессионалов. 4-е издание. — СПб.: Питер, 2021. — 704 с.

4. Кузнецов М. В., Симдянов И. В. Самоучитель MySQL 5. — СПб.: БХВ-Петербург, 2006. — 560 с.

5. Козмина, Юлиана, Харроп, Роб, Шефер, Крис, Хо, Кларенс. Spring 5 для профессионалов.: Пер. с англ. - СПб.: ООО "Диалектика", 2019. - 1120 с.

6. Мартин Роберт К. Чистый код. Создание анализ и рефакторинг. - СПб: Питер, 2019. - 464 с.

7. Шилдт Г. Java. Полное руководство. - Киев: Диалектика, 2018. - 1488 с.

XI. ПРИЛОЖЕНИЕ

1. Информационные терминалы сети кинотеатров «КАРО Фильм»