МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

Учреждения образования «БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ

ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ»

Факультет Информационных технологий

Кафедра Программной инженерии

Специальность 1-40 05 01 Информационные системы и технологии

**ПОЯСНИТЕЛЬНАЯ ЗАПИСКА**

**КУРСОВОГО ПРОЕКТА:**

по дисциплине «Объектно-ориентированные технологии программирования и стандарты проектирования»

Тема Программное средство «Афиша»

Исполнитель

Студент(ка) 2 курса группы 6 Мазенкова И.В.

(Ф.И.О.)

Руководитель работы ассистент Радиванович Д.А.

(учен. степень, звание, должность, подпись, Ф.И.О.)

Курсовой проект защищен с оценкой

Председатель Радиванович Д.А

(подпись)

Минск 2022

МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

Учреждение образования   
«БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ»

Факультет информационных технологий   
Кафедра программной инженерии

Утверждаю

Заведующий кафедрой

\_\_\_\_\_\_\_\_\_\_\_\_ \_\_ Н.В. Пацей

подпись инициалы и фамилия

“\_\_\_”\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_2022г.

**ЗАДАНИЕ**

**к курсовому проектированию**

**по дисциплине** " Объектно-ориентированное технологии программирования и стандарты проектирования "

|  |  |
| --- | --- |
| Специальность: 1-40 01 01 Программное обеспечение информационных технологий  Студент: Мазенкова И.В. | Группа: 4\_\_ |
| **Тема:** Программное средство «Афиша» | |

**1. Срок сдачи студентом законченной работы**: "20 мая 2022 г."

**2. Исходные данные к проекту:**

**2.1**. Функционально ПС поддерживает:

* Функции администратора приложения:
  + Поддерживать работу c базой данных;
  + Управление пользователями и кинотеатрами.
* Функции пользователя:
  + Выполнять регистрацию и авторизацию;
  + Изменение личных данных;
  + Покупка билетов.
* Функции администратор – кинотеатра:
  + Предоставление информации о кино;
  + Создание расписания фильмов.

**2.2.** При выполнении курсового проекта необходимо использовать принципы проектирования ООП. Приложение разрабатывается под ОС Windows и представляет собой настольное приложение (desktop). Отображение, бизнес логика должны быть максимально независимы друг от друга для возможности расширения. Диаграммы вариантов использования, классов реализации задачи, взаимодействия разработать на основе UML. Язык разработки проекта – C#. Управление программой должно быть интуитивно понятным и удобным. При разработке использовать несколько наиболее подходящих шаблонов проектирования ПО.

**3. Содержание расчетно-пояснительной записки**

(перечень вопросов, подлежащих разработке)

* Введение
* Постановка задачи и обзор литературы (алгоритмы решения, обзор прототипов, актуальность задачи)
* Проектирование архитектуры проекта (структура модулей, классов).
* Разработка функциональной модели и модели данных ПС (выполняемые функции)
* Тестирование
* Заключение
* Список используемых источников
* Приложения

**4. Форма представления выполненного курсового проекта:**

* + Теоретическая часть курсового проекта должны быть представлены в фор­мате docx. Оформление записки должно быть согласно выданным правилам.
  + Листинги программы представляются частично в приложении.
  + Пояснительную записку, листинги, проект (инсталляцию проекта) необходимо загрузить диск, указанный преподавателем.

***Календарный план***

|  |  |  |  |
| --- | --- | --- | --- |
| №  п/п | Наименование этапов курсового проекта | Срок выполнения этапов проекта | Примечание |
| 1 | Введение | 19.02.2022 |  |
| 2 | Аналитический обзор литературы по теме проекта. Изучение требований, определение вариантов использования | 12.03.2022 |  |
| 3 | Анализ и проектирование архитектуры приложения (построение диаграмм, проектирование бизнес-слоя, представления и данных) | 26.03.2022 |  |
| 4 | Проектирование структуры базы данных. Разработка дизайна пользовательского интерфейса | 2.04.2022 |  |
| 5 | Кодирование программного средства | 23.04.2022 |  |
| 6 | Тестирования и отладка программного средства | 30.04.2022 |  |
| 7 | Оформление пояснительной записки | 7.05.2022 |  |
| 9 | Сдача проекта | 20.05.2022 |  |

**5. Дата выдачи задания \_\_\_\_**12.02.2022**\_\_\_\_**

Руководитель\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ *А.Н. Мущук*

(подпись)

Задание принял к исполнению \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(дата и подпись студента)

**СОДЕРЖАНИЕ**

[Введение 5](#_Toc72703160)

[1 Аналитический обзор прототипов и литературных источников 6](#_Toc72703161)

[2 Анализ требований к программному средству и разработка 9](#_Toc72703162)

[2.1 Постановка требований 9](#_Toc72703163)

[2.2 Диаграмма вариантов использования 9](#_Toc72703164)

[3 Проектирование программного средства 12](#_Toc72703165)

[3.1 Обобщение структуры 12](#_Toc72703166)

[3.2 Диаграммы UML 13](#_Toc72703167)

[3.4 Схема базы данных 14](#_Toc72703169)

[4 Разработка функциональной модели 17](#_Toc72703170)

[4.1 Реализация MVVM 17](#_Toc72703171)

[4.2 Command 17](#_Toc72703172)

[4.3 Реализация авторизации и регистрации пользователей 17](#_Toc72703173)[4.4 Реализация заказа билета пользователем 18](#_Toc72703174)

[4.5 Функции для работы с базой данных 18](#_Toc72703175)

[5 Тестирование 20](#_Toc72703176)

[6 Руководство по установке и использованию 22](#_Toc72703177)

[6.1 Регистрация и авторизация 22](#_Toc72703178)

[6.2 Использование приложения обычным пользователем 23](#_Toc72703179)

[6.3 Использование приложения администратором 25](#_Toc72703180)

[6.4 Использование приложения администратором кинотеатра 26](#_Toc72703181)

[Заключение 28](#_Toc72703182)

[Список использованной литературы 29](#_Toc72703183)

[Приложение А 30](#_Toc72703184)

[Приложение Б 31](#_Toc72703185)

[Приложение В 32](#_Toc72703186)

[Приложение Г 33](#_Toc72703187)

[Приложение Д 34](#_Toc72703188)

[Приложение Е 35](#_Toc72703189)

[Приложение Ж 36](#_Toc72703190)

[Приложение З 38](#_Toc72703191)

# ВВЕДЕНИЕ

Кино – это не просто вид искусства, это особый мир, который словно машина времени позволяет нам окунуться как в атмосферу настоящего, так и прошлого. Современному человеку сложно представить, что в самом начале своего пути кино было черно-белое и немое. Несмотря на это, ученые изобретают много технологий, которые улучшают качество кинотеатров. Развитие современных цифровых технологий сделало кино еще более совершенным и реалистичным, ведь спецэффекты, компьютерная графика и технология 3D способны создать любую, даже самую фантастическую реальность. Эти технологии позволяют зрителю полностью погрузиться в реальность происходящего.

В наше время кино имеет огромную силу, ведь оно формирует вкусы и мировоззрение, а также влияет на массовое сознание людей. Многие жанры кино не только помогают интересно провести время, но еще и просвещают нас. К жанрам подобного рода стоит отнести исторические и документальные фильмы. Эти фильмы в очень доступной форме передают нам знания о мире, которые накапливались веками и позволяют нам погрузиться в атмосферу любой эпохи.

Кинотеатр – хороший способ провести свободное время, но тем не менее заказать билеты бывает проблематично и не всегда удобно отправляться в кинотеатр за их покупкой, ведь к приезду билетов может и не быть. Программное средство “Афиша” позволяет быстро и удобно заказать билеты на интересующий вас фильм. Благодаря этой платформы, кинотеатры без проблем могут добавлять новые фильмы в прокат, а также сообщать о кинопремьерах.

1. **Аналитический обзор прототипов и литературных источников**

Немаловажным этапом в разработке программного продукта является аналитический обзор прототипов.

На сегодняшний день можно встретить достаточно большое количество программных решений, разработанных для просмотра фильмов в прокате и заказов билетов в кинотеатр.

«ByCard» – сервис продажи билетов в кинотеатры. На сайте представлены фильмы, которые идут в прокате, а также премьеры и дата их выхода в прокат. Вы можете найти интересующий вас фильм, выбрать дату и стоимость билета, также кинотеатр и купить билеты на свободные места.

Интерфейс «ByCard» представлен на рисунке 1.1.

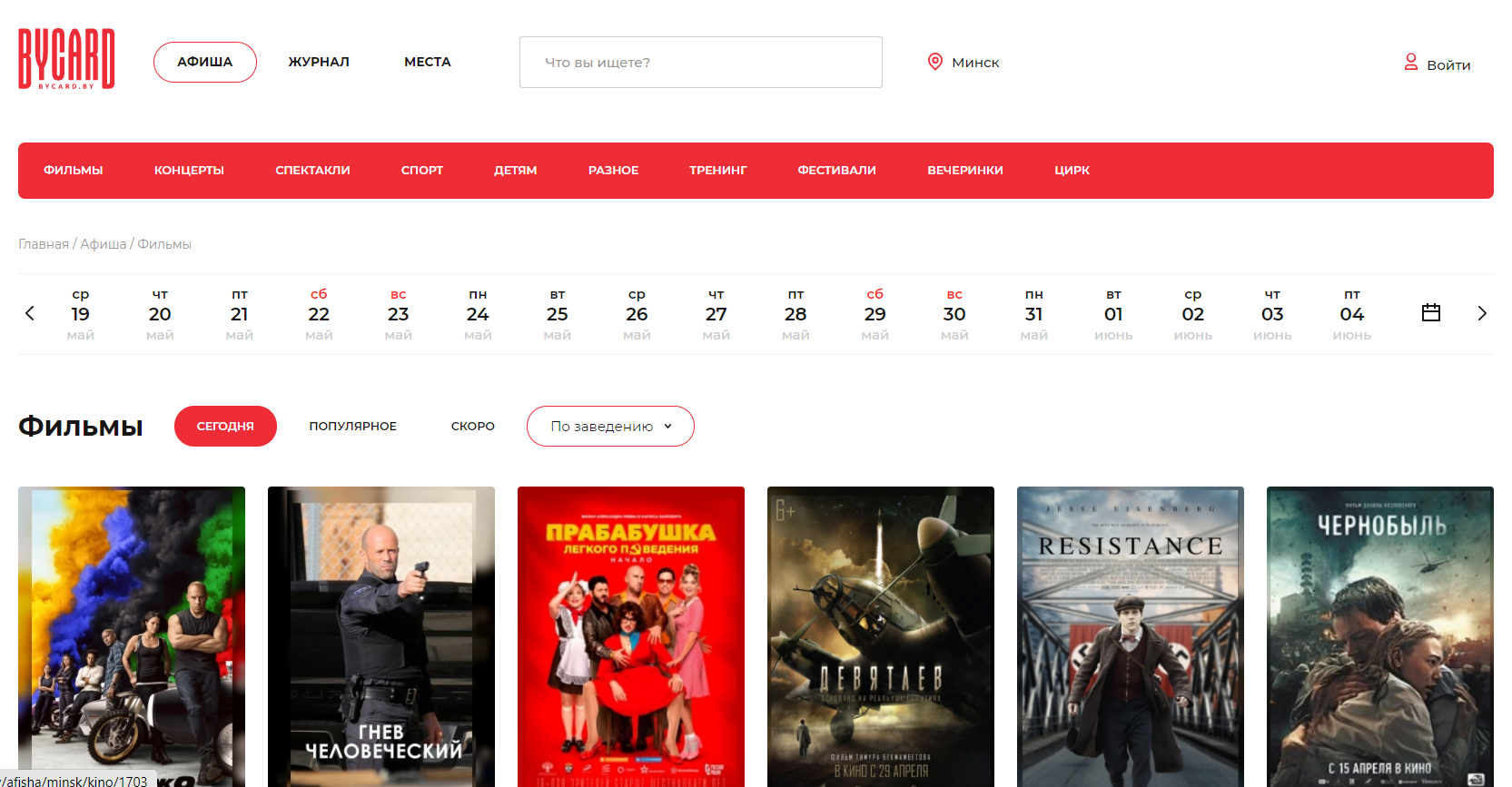


Рисунок 1.1 − Интерфейс «ByCard»

Данный сервис грамотно спроектировано и выполняет большое количество функций.

Основные минусы:

* Отсутствует работа техподдержки;
* Высокая комиссия;
* Необходимость печатать билет.

Основные плюсы:

* Простота в использовании;
* Хороший UI.

«Silver Screen Cinemas» – сеть кинотеатров в Минске. На сайте представлена информация о фильмах, которые идут в прокате, а также премьеры и дата их выхода в прокат только для сети кинотеатров Silver Screen.

Интерфейс «Silver Screen Cinemas» представлен на рисунке 1.2.

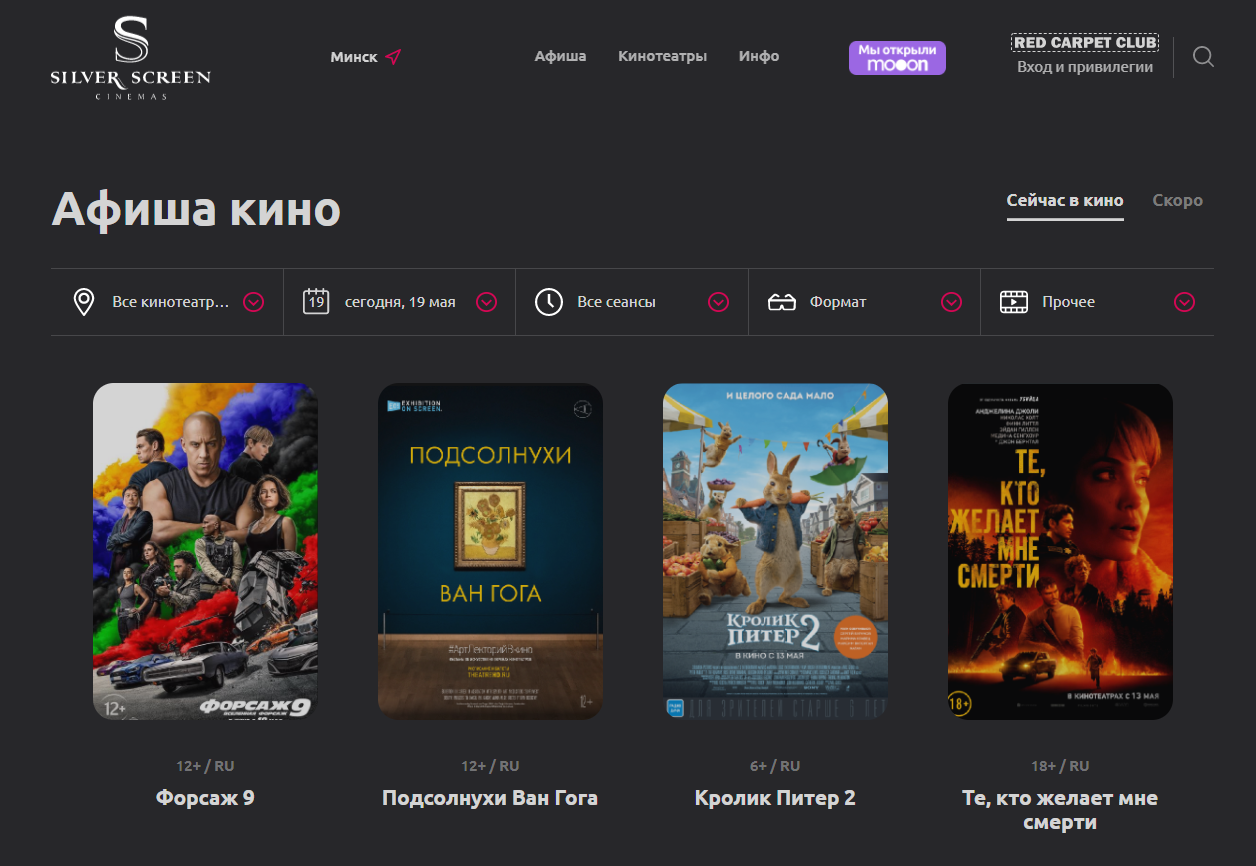


Рисунок 1.2 − Интерфейс «Silver Screen Cinemas»

Основные минусы:

* Высокая комиссия.

Основные плюсы:

* Простота в использовании;
* Хороший UI.

«Яндекс Афиша» – русскоязычный сервис по продаже билетов на развлекательные мероприятия: театры, концерты, кино, мюзиклы, шоу и другие события. На сайте представлена информация о фильмах, которые идут в прокате.

Интерфейс «Яндекс Афиша» представлен на рисунке 1.3.

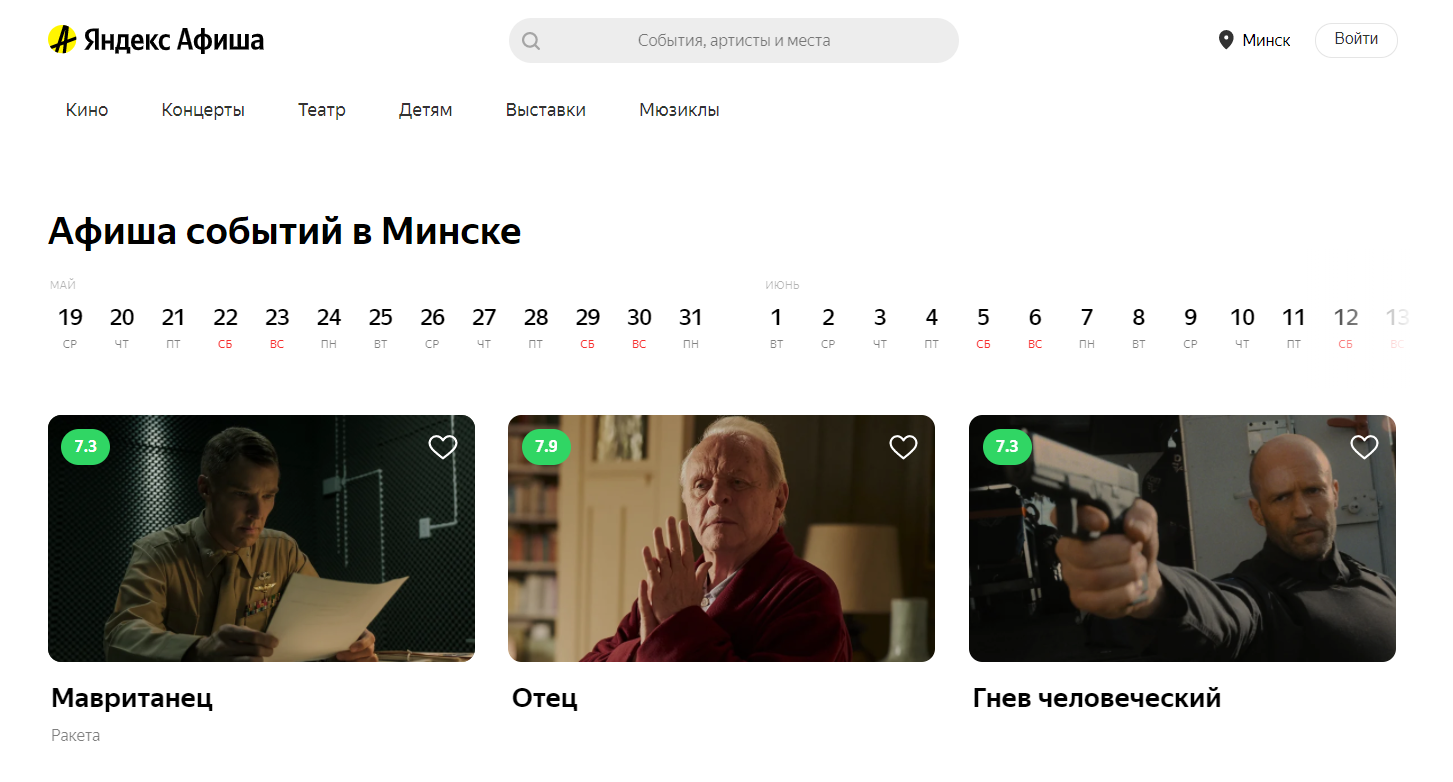


Рисунок 1.3 − Интерфейс «Яндекс Афиша»

Основные минусы:

* Отсутствует работа техподдержки.

Основные плюсы:

* Простота в использовании;
* Хороший UI.

1. **Анализ требований к программному средству и разработка**

**функциональных требований**

* 1. **Постановка требований**

Задачей данной курсовой работы является разработка приложения, которое будет представлять собой сервис по заказам билетов в кинотеатре. Приложение является расширяемым, так как поддерживает функции администратора и администраторов кинотеатров, которые могут изменять базу данных и давать пользователям новые возможности.

Пользователь данного программного решения имеет возможности:

* регистрировать себя в системе;
* входить в приложение, после ввода данных, необходимых для аутентификации;
* изменять личные данные;
* заказывать билеты;
* просматривать информацию о всех заказанных билетах;
* просматривать фильмы в прокате;
* поиск по фильмам.

Администратор имеет возможности:

* просматривать информацию о текущих пользователях и администраторах кинотеатров;
* удалять пользователей;
* редактировать пароль пользователей и администраторов кинотеатров;
* регистрировать новых пользователей;
* добавлять новые кинотеатры с их администраторами.

Администратор кинотеатра имеет возможности:

* просматривать информацию о всех фильмах, доступных к прокату;
* просматривать информацию о текущих билетах для своего кинотеатра;
* просматривать билеты, заказанные пользователями;
* добавлять новые фильмы в прокат;
* добавлять новые сеансы для своего кинотеатра;
* добавлять новые залы;
* удалять фильмы из проката.
  1. **Диаграмма вариантов использования**

В диаграмме использования проектируемая система представляется в виде множества сущностей или актеров, взаимодействующих с системой с помощью, так называемых прецедентов. При этом актером или действующим лицом называется любая сущность, взаимодействующая с системой извне. Другими словами, каждый вариант использования определяет некоторый набор действий, совершаемый системой при диалоге с актером. При этом ничего не говорится о том, каким образом будет реализовано взаимодействие актеров с системой.

Диаграмма вариантов использования для обычного пользователя приведена на рисунке 2.1.

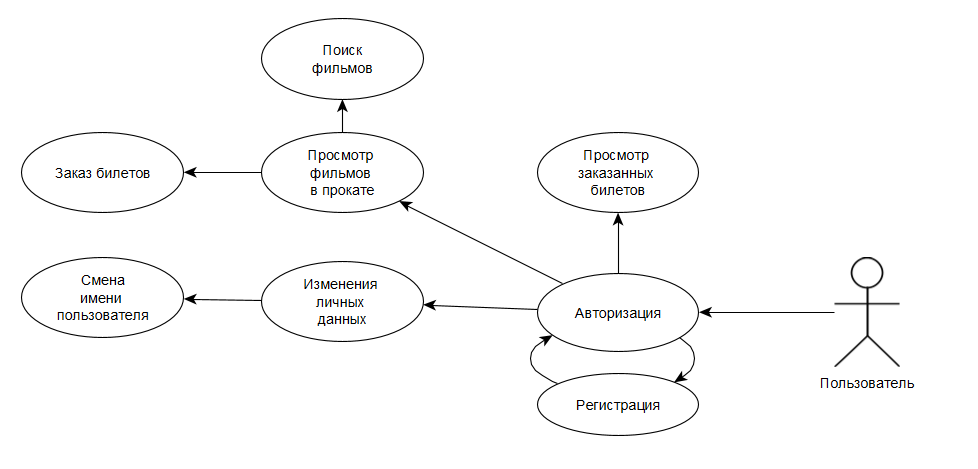


Рисунок 2.1 – Диаграмма вариантов использования для обычного пользователя

На данной диаграмме можно увидеть основные действия, которые доступ­ны пользователю: регистрация и авторизация; просмотр афиши фильмов; поиск по фильмам; заказ билетов по интересующему фильму; просмотр заказанных билетов; изменение личных данных – смена имени пользователя.

Диаграмма вариантов использования для администратора приложения приведена на рисунке 2.2.

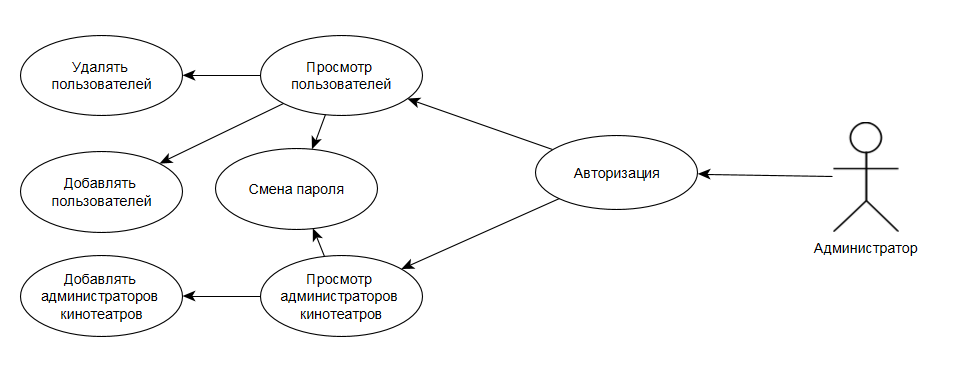


Рисунок 2.2 – Диаграмма вариантов использования для администратора

На данной диаграмме можно увидеть основные возможности, которые доступны администратору: просмотр всего перечня пользователей и администраторов кинотеатров; добавление пользователей и администраторов кинотеатров, где при добавлении администратора кинотеатра будет создан соответствующий ему кинотеатр и зал со вместимостью по умолчанию 100; удаление пользователей, где будут удалены заказанные билеты пользователя; смена пароля пользователя или администратора кинотеатра.

Диаграмма вариантов использования для администратора кинотеатра приведена на рисунке 2.3.

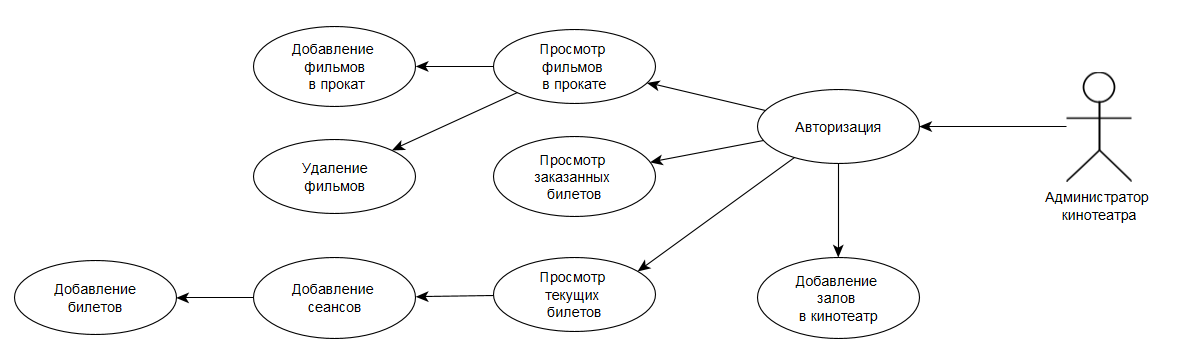


Рисунок 2.3 – Диаграмма вариантов использования для администратора кинотеатра

На данной диаграмме можно увидеть основные возможности, которые доступны администратору кинотеатру: добавление залов в кинотеатр с выбором вместимости; просмотр фильмов в прокате, заказанных билетов по всем кинотеатрам и просмотр текущих билетов для текущего кинотеатра; добавления фильма в прокат, а также его удаление, где удаляются все сеансы этого фильма; добавление сеансов, после создания которых автоматические генерируются билеты.

1. **Проектирование программного средства**
   1. **Обобщение структуры**

При создании проекта использовались две среды разработки: одна — для работы с базами данных, другая — для создания самого приложения.

Microsoft SQL Server — система управления реляционными БД. Основной используемый язык запросов – Transact-SQL — реализован на структурированном языке запросов (SQL) с расширениями. SQL Server Management Studio — утилита из Microsoft SQL Server 2008 и более поздних версий для конфигурирования, управления и администрирования всех компонентов Microsoft SQL Server.

Microsoft Visual Studio 2022 — это набор инструментов для создания программного обеспечения: от планирования до разработки пользовательского интерфейса, написания кода, тестирования, отладки, анализа качества кода и производительности, развертывания в средах клиентов и сбора данных по использованию.

Решение представлено одним проектом «MoviesAfisha», имеющим структуру, представленную на рисунке 3.1.

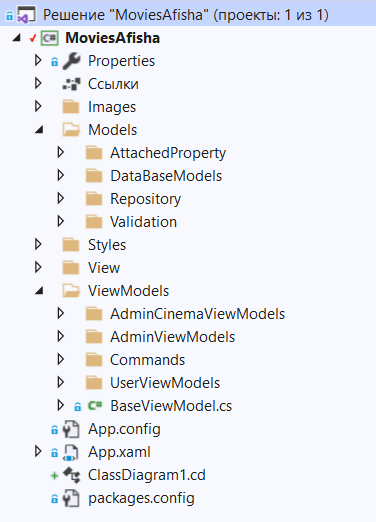


Рисунок 3.1 – Обобщенная структура проекта

Данный курсовой проект основывается на реали­зации паттерна MVVM (Model-View-ViewModel), который позволяет отделить логику приложения от визуальной части, разберем архитектуру данного паттерна:

* Models — папка с моделями описывающая используемые в приложении данные. Класс AttachedProperty необходим для привязки данных вводимых в элементе управления PasswordBox;
* Images – папка содержащая картинки, используемые для визуализации приложения, а также постеры к кино;
* Styles – папка, содержащая стили для элементов управления, используемых в приложении;
* View – папка, содержащая представления определяющие визуальный интерфейс приложения;
* ViewModels – папка, содержащая модель представления связывающая модель и представления через механизм привязки данных. В классе BaseViewModel реализован интерфейс INotifyPropertyChanged;
* Commands – подпапка ViewModels, содержащая класс RelayCommand, необходимый для привязки команд к элементам управления.
  1. **Диаграммы UML**

UML — унифицированный язык моделирования (Unified Modeling Language) — это система обозначений, которую можно применять для объектно-ориентированного анализа и проектирования.

Его можно использовать для визуализации, спецификации, конструирова­ния и документирования программных систем.

Диаграмма — это графическое представление набора элементов, чаще всего изображенного в виде связного графа вершин (сущностей) и путей (связей).

Диаграммы последовательностей используются для уточнения диаграмм прецедентов, более детального описания логики сценариев использования. Это отличное средство документирования проекта с точки зрения сценариев использования.

Диаграмма последовательности заказа билета приведена в приложении А.

Диаграмма деятельности, как и диаграмма состояний, отражает динамические аспекты поведения системы. По существу, эта диаграмма представляет собой блок-схему, которая наглядно показывает, как поток управления переходит от одной деятельности к другой.

Диаграммы деятельности используются при моделировании бизнес-процессов, технологических процессов, последовательных и параллельных вычислений.

Диаграмма деятельности пользователя приведена в приложении Б.

Диаграмма компонентов — элемент языка моделирования UML, статическая структурная диаграмма, которая показывает разбиение программной системы на структурные компоненты и связи (зависимости) между компонентами. В качестве физических компонентов могут выступать файлы, библиотеки, модули, исполняемые файлы, пакеты и т. п.

Диаграмма развертывания показывает топологию системы и распределение компонентов системы по ее узлам, а также соединения — маршруты передачи информации между аппаратными узлами. Это единственная диаграмма, на которой применяются “трехмерные” обозначения: узлы системы обозначаются кубиками. Все остальные обозначения в UML — плоские фигуры.

Диаграммы компонентов приведена в приложении В. Так как приложение построено на архитектуре паттерна MVVM, то присутствует 3 основных блока, View, ViewModel и Model. В Model входит паттерн репозиторий, паттерн UnitOfWork, классы, созданные Entity Framework способом подключения CodeFirst. View представляет собой внешний вид приложения. Данный паттерн позволяет легко извлекать информацию из базы данных и передать ее сначала во ViewModel, а затем уже отобразить во View.

Диаграмма классов служит для представления статической структуры мо­дели системы в терминологии классов объектно-ориентированного программирования. Диаграмма классов может отражать, в частности, различные взаимосвязи между отдельными сущностями предметной области, такими как объекты и подсистемы, а также описывает их внутреннюю структуру (поля, методы) и типы отношений (наследование, реализация интерфейсов). На данной диаграмме не указывается информация о временных аспектах функционирования системы. С этой точки зрения диаграмма классов является дальнейшим развитием концептуальной модели проектируемой системы. Диаграмма всех классов представлена в приложении Г.

* 1. **Схема базы данных**

В реляционной модели, в отличие от иерархической или сетевой, не суще­ствует физических отношений. Вся информация хранится в виде таблиц (отношений), состоящих из рядов и столбцов. А данные двух таблиц связаны общими столбцами, а не физическими ссылками или указателями. Для манипуляций с рядами данных существуют специальные операторы.

В отличие от двух других типов СУБД, в реляционных моделях данных нет необходимости просматривать все указатели, что облегчает выполнение запросов на выборку информации по сравнению с сетевыми и иерархическими СУБД. Это одна из основных причин, почему реляционная модель оказалась более удобна. В реляционной модели, как объекты, так и их отношения представлены только таблицами, и ничем более.

Для реализации приложения была разработана база данных MoviesAfisha. Для создания базы данных был выбран Microsoft SQL Server 2018.

Модель базы данных — тип модели данных, которая определяет логическую структуру базы данных и принципиально определяет, каким образом данные могут быть сохранены, организованы и обработаны.

Текущая база данных состоит из девяти таблиц, ее структура представлена на рисунке 3.4.

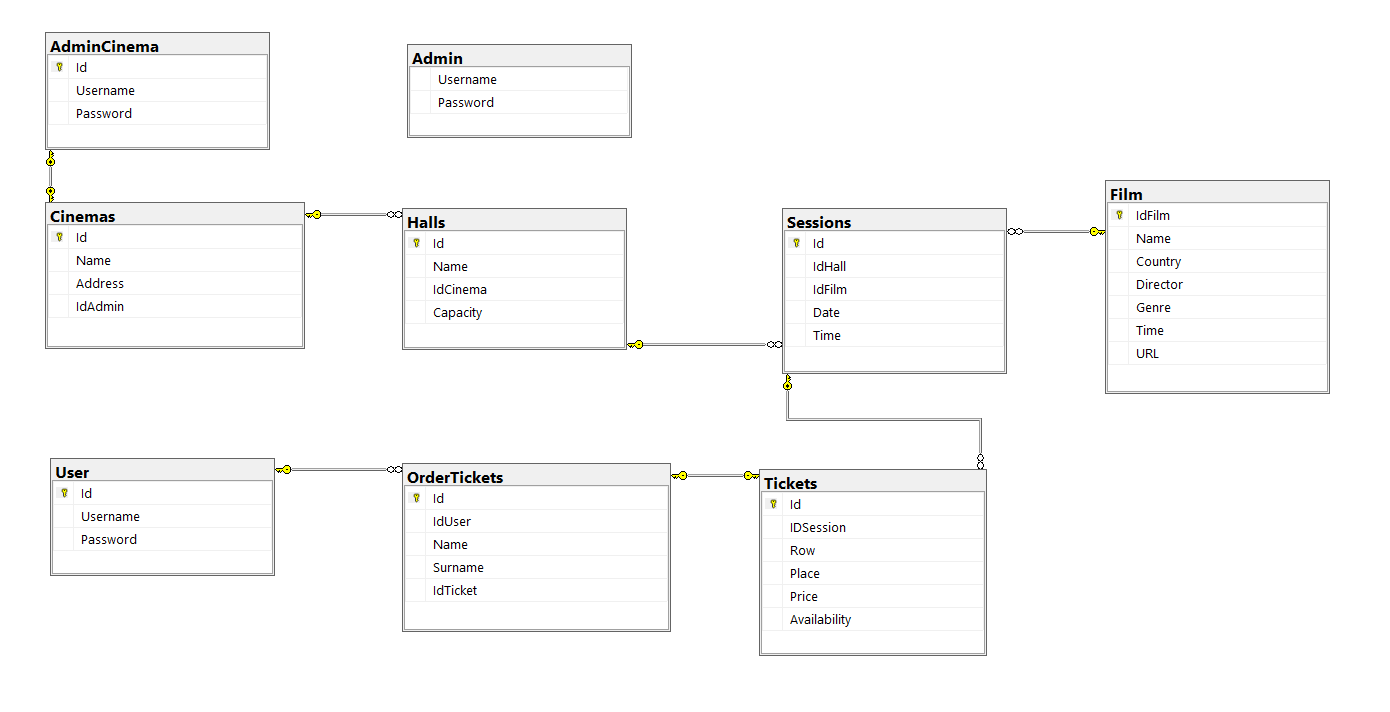


Рисунок 3.4 – Схема базы данных

Таблица User хранит каждого пользователя, так же всю информацию о нем. Хранятся личный данные пользователя: id, имя пользователя, пароль. Id является уникальным для каждого пользователя. Имя пользователя и пароль используются для входа: имя пользователя, которое должно быть уникальным, и пароль, который хранится в базе в виде хэш-значения, для безопасности данных пользователя.

Таблица OrderTickets хранит информацию о всех заказанных билетах пользователями: Id заказанного билета, Id пользователя, имя и фамилия заказчика, который будет забирать билеты, Id выбранного билета. Id заказанного билета является уникальным.

Таблицы User и OrderTickets связаны одни-ко-многим.

Таблица Tickets хранит все билеты кинотеатров, доступные для заказа: Id билета, Id сеанса, ряд, место и доступность к заказу. Id билета является уникальным. Доступность к заказу является булевым значением, по умолчанию имеет значение true. При заказе билета, доступность меняется на false.

Таблицы Tickets и OrderTickets связаны один-к-одному.

Таблица Sessions хранит информацию о всех сеансах: Id сеанса, Id зала, Id фильма, дату и время. Id сеанса является уникальным.

Таблицы Sessions и Tickets связаны один-ко-многим.

Таблица Film хранит информацию о фильмах: Id фильма, название фильма, страну, режиссёра, жанр, продолжительность фильма и путь к обложке. Id фильма является уникальным значением. Столбцы страна, режиссёр, продолжительность и жанр являются необязательными для заполнения.

Таблицы Sessions и Film связаны один-ко-многим.

Таблица Halls хранит информацию о залах для кинотеатров: Id зала, название, Id кинотеатра и вместимость зала. Id зала является уникальным значением.

Таблицы Halls и Sessions связаны один-ко многим.

Таблица Cinemas хранит информацию о кинотеатрах: Id кинотеатра, название, адрес и Id администратора приложения. Id кинотеатра и Id администратора приложения являются уникальными.

Таблицы Cinemas и Halls связаны один-ко-многим.

Таблица AdminCinema хранит информацию об администраторах кинотеатров: id, имя администратора кинотеатра, пароль. Id является уникальным. Имя администратора кинотеатра и пароль используются для входа: имя, которое должно быть уникальным, и пароль, который хранится в базе в виде хэш-значения, для безопасности данных.

Таблица AdminCinema и Cinemas связаны один-к-одному.

Таблица Admin хранит информацию об одном администраторе приложения. Имя пользователя “admin”, пароль “admin”. В базе пароль хранится в виде хэш-значения.

1. **Разработка функциональной модели**
   1. **Реализация MVVM**

Применением паттерна MVVM является функциональное разделение приложения на три компонента, которые проще разрабатывать и тестировать, а также в дальнейшем модифицировать и поддерживать.

Первым делом для разработки проекта было определить базовые классы-модели, выражающие сущности, с которыми будет происходить взаимодействие.

Для реализации классов моделей представления был создан класс-родитель BaseViewModel, который реализует интерфейс INotifyPropertyChanged. Программный код этого класса представлен на рисунке 4.1.

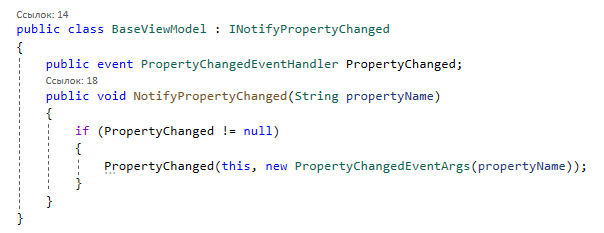


Рисунок 4.1 – Класс BaseViewModel

Реализация интерфейса INotifyPropertyChanged позволяет отлавливать изменения и правильно на их реагировать и сохранять.

* 1. **Command**

Для взаимодействия пользователя и приложения в MVVM используются команды. Для использования команд в проекте следовало добавить в него новый класс, который был назван RelayCommand.

Программный код этого класса и команды, реализованные с его помощью предоставлены в приложении Д.

* 1. **Реализация авторизации и регистрации пользователей**

Для того чтобы пользоваться приложением каждый пользователь должен войти в систему, предварительно зарегистрировавшись.

Код, реализующий регистрацию, приведен в приложении Е.

В данном фрагменте кода присутствует проверка на заполнение полей “имя пользователя” и “пароль”. При заполнении только пробелами или оставление полей пустыми, пользователь информируется об необходимости заполнить то или иное поле. Для безопас­ности хранения информации используется хэширование, в данном случае применяется библиотека BCrypt.Net с алгоритмом SHA384. Если все данные были введены верно и имя пользователя еще не занят, то осуществляется регистрация и переход на страницу входа.

Код, реализующий авторизацию, приведен в приложении Ж.

При авторизации также присутствует проверка на заполнение всех полей, где в противном случае пользователь информируется об необходимости заполнить поля. После проверяется имя пользователя, в зависимости от которого будет открыто одно из окон: окно администратора приложения, окно пользователя или окно администратора кинотеатра, с соответствующими для него данными.

* 1. **Реализация заказа билета пользователем**

Для заказа билета пользователю необходимо выбрать интересующий его фильм и нажать на кнопку “Заказать билет”. В новом окне пользователь выбирает нужный ему билет. Реализован поиск билетов по параметрам: кинотеатр, дата и время. После выбора пользователь нажимает на кнопку заказать, где вводит в новом окне имя и фамилию. Информация сохраняется в таблицу заказанных билетов, где булевое значение статуса билета меняется с “true” на “false”. Заказанный билет недоступен для повторного заказа. При успешном заказе открывается окно с информацией “Билет заказан”. Пользователь может заказать еще билеты, не открывая повторно окно со всеми билетами для текущего фильма. Все заказанные билеты пользователь может посмотреть в личном аккаунте. Код, реализующий добавление заказанного билета, приведен в приложении З.

* 1. **Функции для работы с базой данных**

Работа с базой данных осуществляется согласно паттернам Repository и Unit Of Work. Repository создает абстракцию при работе с базой данных, и дает возможность без каких-либо проблем изменить используемую базу данных. Unit Of Work позволяет упростить работу с различными репозиториями и дает уверенность, что все репозитории будут использовать один и тот же контекст данных.

Для реализации паттерна Repository необходимо определить интерфейс и создать отдельные его реализации для каждого класса, соответствующие таблицам из базы данных MoviesAfisha.

На рисунке 4.2 приведен интерфейс для паттерна Repository.

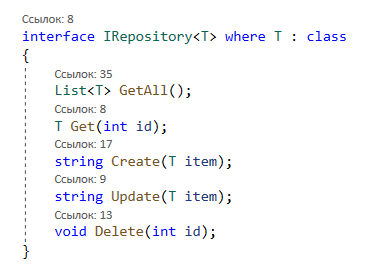


Рисунок 4.2 – Интерфейс для паттерна Repository

Интерфейс содержит CRUD функции.

Функции, которые определяет интерфейс: добавление объекта в базу данных, добавление коллекции объектов, удаление коллекции объектов, получение всех объектов таблицы, удаление одного объекта, изменение объекта.

Для использования паттерна Unit of Work необходимо создать новый класс, предоставляющий доступ к репозиториям через отдельные свойства и определяющий общий контекст для всех репозиториев.

На рисунке 4.3 приведена реализация паттерна Unit Of Work.

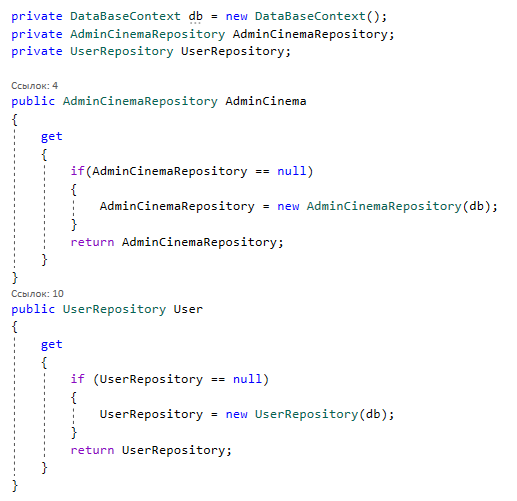


Рисунок 4.3 – Реализация паттерна Unit Of Work

В итоге мы получаем абстракцию при работе с базой данных, и получаем возможность без каких-либо проблем изменить используемую базу данных.

1. **Тестирование**

В курсовом проекте задействуется обработка исключений, таким образом, что пользователь будет уведомлен о неудачном выполнении запроса в базе дан­ных. Присутствуют различные всплывающие окна, выводящие сообщение об ошибке.

Ниже на рисунках 5.1, 5.2, 5.3 представлены результаты обработки исключений при неверно введенных данных при авторизации.



Рисунок 5.1 – Не введено имя пользователя

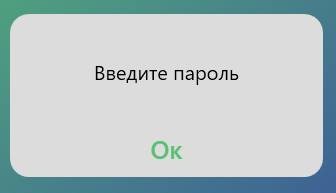


Рисунок 5.2 – Не введен пароль

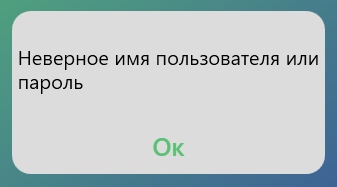


Рисунок 5.3 – Неверно введен пароль или имя пользователя

На рисунках 5.1, 5.2, 5.4 представлены результаты обработки исключений при регистрации пользователя. Ошибка на рисунке 5.4 происходит по причине того, что для поля имя пользователя стоить ограничение Unique.

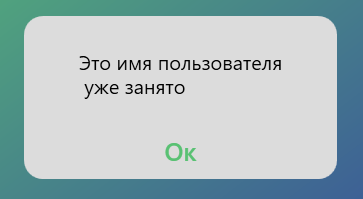


Рисунок 5.4 – Имя пользователя уже занято

Во всех остальных окнах, где присутствуют входные данные имеется такая же система обработки ошибок. Помимо этого, приложение постоянно уведомляет пользователя о событиях, произошедших в системе: удаления объекта, сохранение проекта, сохранение данных. Так же в случае случайного нажатия на кнопку уда­ления, появляется окно, которое проверяет действительно ли пользователь хочет удалить данные. Пример уведомления пользователя об успешной регистрации на рисунке 5.5.

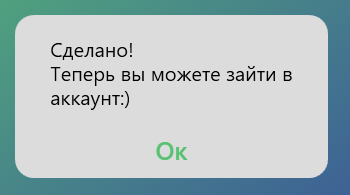


Рисунок 5.5 – Успешная регистрация

После успешной регистрации и авторизации пользователю откроется главное окно, где он сможет выбрать фильм для заказа билетов в кинотеатр, а также изменить своё имя пользователя.

1. **Руководство по установке и использованию**

Запустить приложение.

* 1. **Регистрация и авторизация**

После запуска приложения пользователь видит главное окно приложения, на котором он может авторизоваться и получить возможность заказать билет. Окно представлено на рисунке 6.1.

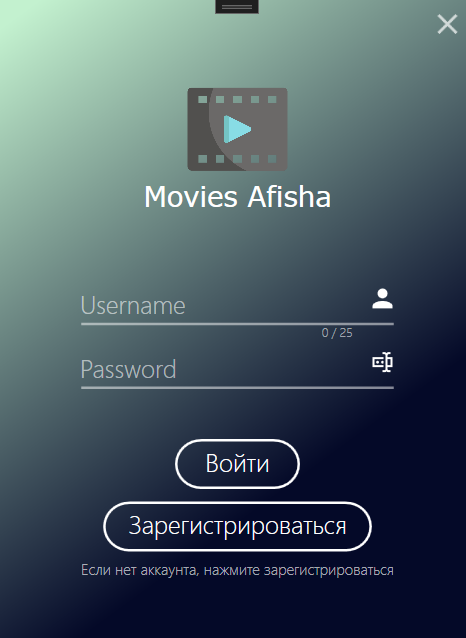


Рисунок 6.1 – Окно входа в аккаунт

Если пользователь ещё не имеет своего аккаунта, то по нажатию на кнопку “Зарегистрироваться” перейдет к окну регистрации. Необходимо ввести уникальное имя пользователя и пароль. При успешной регистрации пользователь будет уведомлен и перенаправлен на окно авторизации.

Окно регистрации показано на рисунке 6.2.

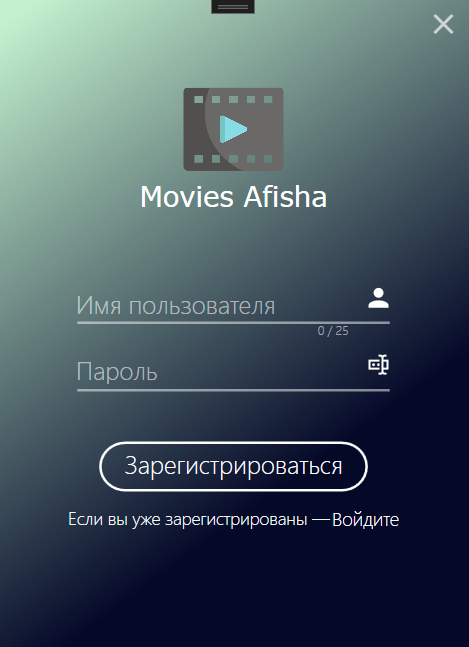


Рисунок 6.2 – Окно регистрации пользователя

После регистрации пользователь должен авторизоваться и войти.

* 1. **Использование приложения обычным пользователем**

В случае успешного входа пользователь переходит на главную страницу приложения, в котором отображаются постера к фильмам.

Главное стартовое окно, которое запускается после авторизации приведено на рисунке 6.3

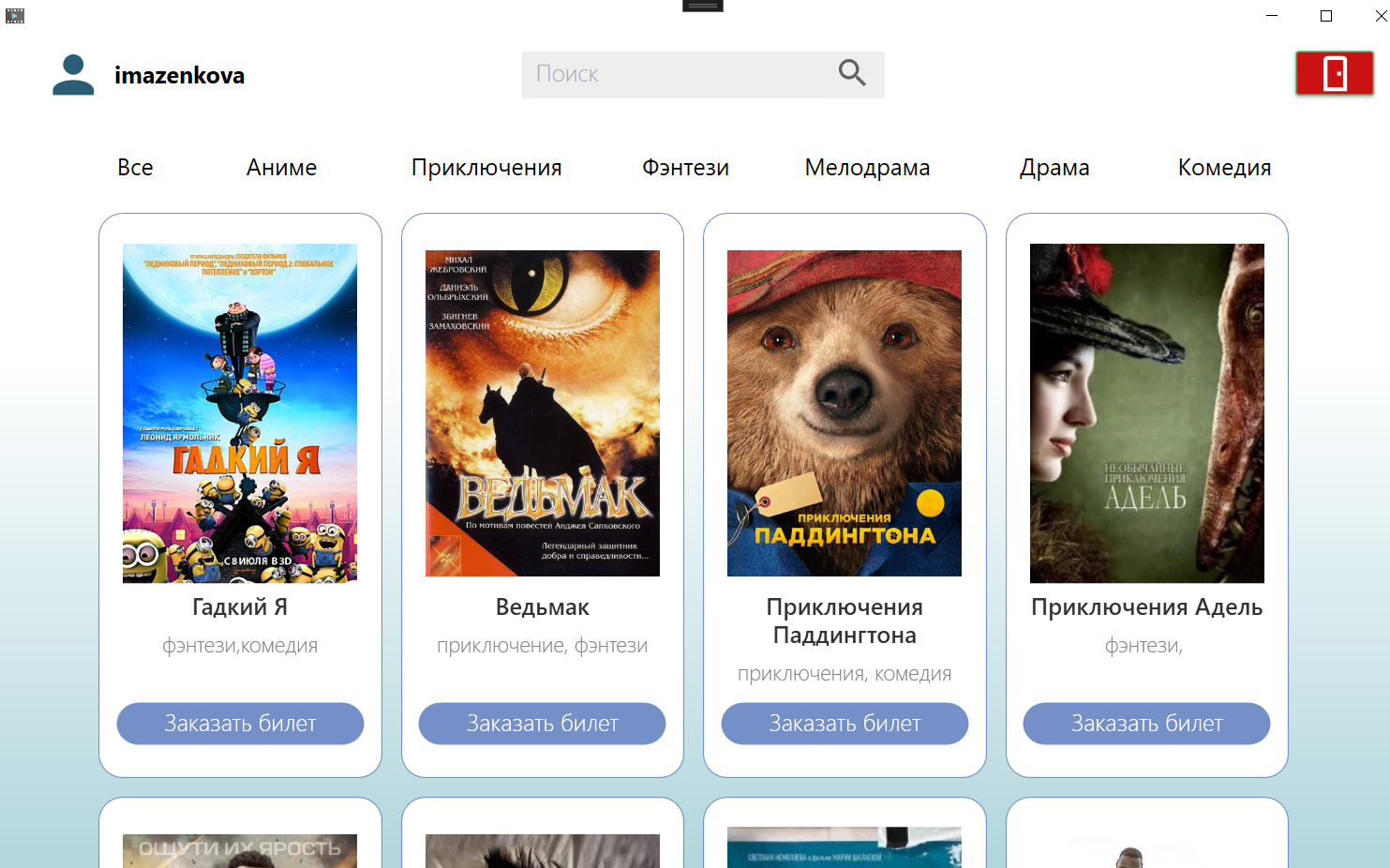


Рисунок 6.3 – Главное стартовое окно

В главном окне сверху слева кнопка перехода на профиль пользователя, сверху справа возвращение на окно авторизации.

Возможен поиск по названиям фильмов и сортировка по жанрам, также заказ билетов на интересующий фильм.

При нажатии на кнопку профиля пользователь переходит в окно, которое показано на рисунке 6.4.

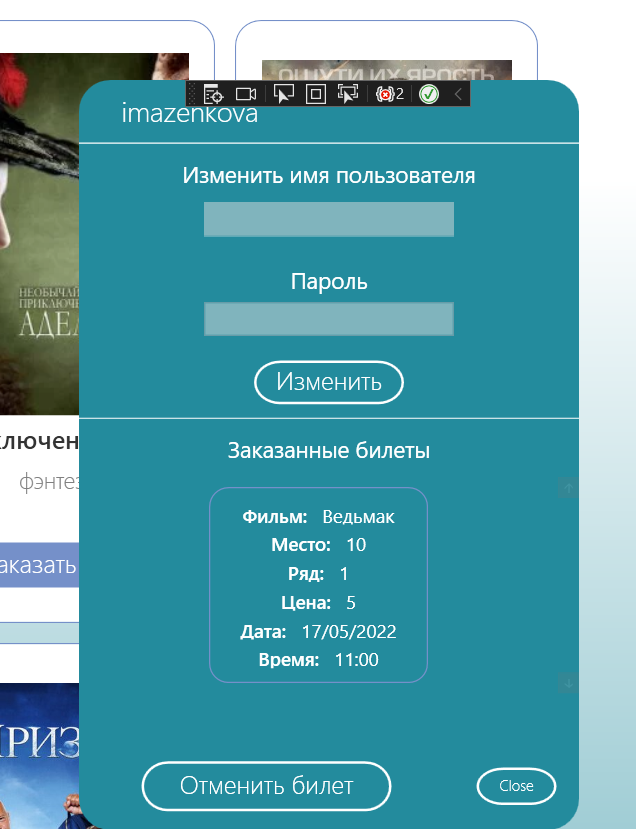


Рисунок 6.4 – Окно настроек

В данном окне можно сменить только имя пользователя при условии, что оно не занято другим пользователем. Возможность смены пароля пользователя доступно только администратору приложения. Также в этом окне пользователь может посмотреть заказанные билеты.

При выборе нужного фильма, пользователь нажимает на кнопку «заказать билет», где открывается окно заказа билетов, которое показано на рисунке 6.5.



Рисунок 6.5 – Окно заказа билетов

Пользователь может сортировать билеты, выбрав критерии: название кинотеатра, дата и время сеанса. После нажав на кнопку “Искать”.

При отсутствии билетов по заданным критериям пользователь будет уведомлен. При нажатии на кнопку “Показать все билеты” пользователю будут показаны все билеты, доступные к заказу.

После выбора нужного билета, пользователь нажимает “Заказать”, где открывается окно для ввода информации заказчика, которое показано на рисунке 6.6.

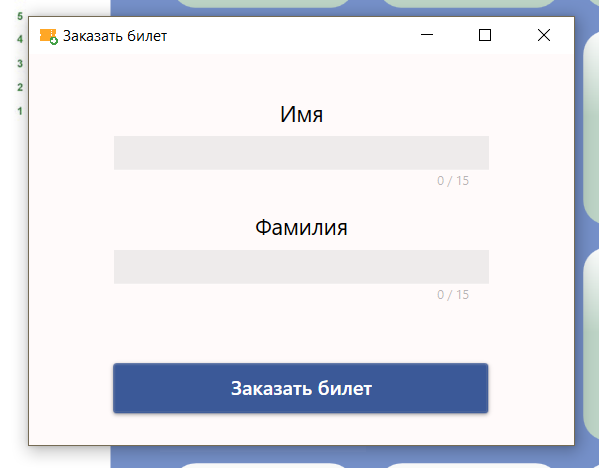


Рисунок 6.6 – Окно ввода информации для заказа

После корректного ввода информации пользователь будет уведомлен об успешном заказе, где вернется на окно билетов для выбранного ранее фильма. Это дает возможность заказать ещё билеты, не искав опять тот же фильм.

* 1. **Использование приложения администратором**

Администратор приложения может войти по имени пользователя “admin” и паролю “admin”. В системе возможен только администратор приложения. После входа откроется главное окно, пример которого показан на рисунке 6.7.

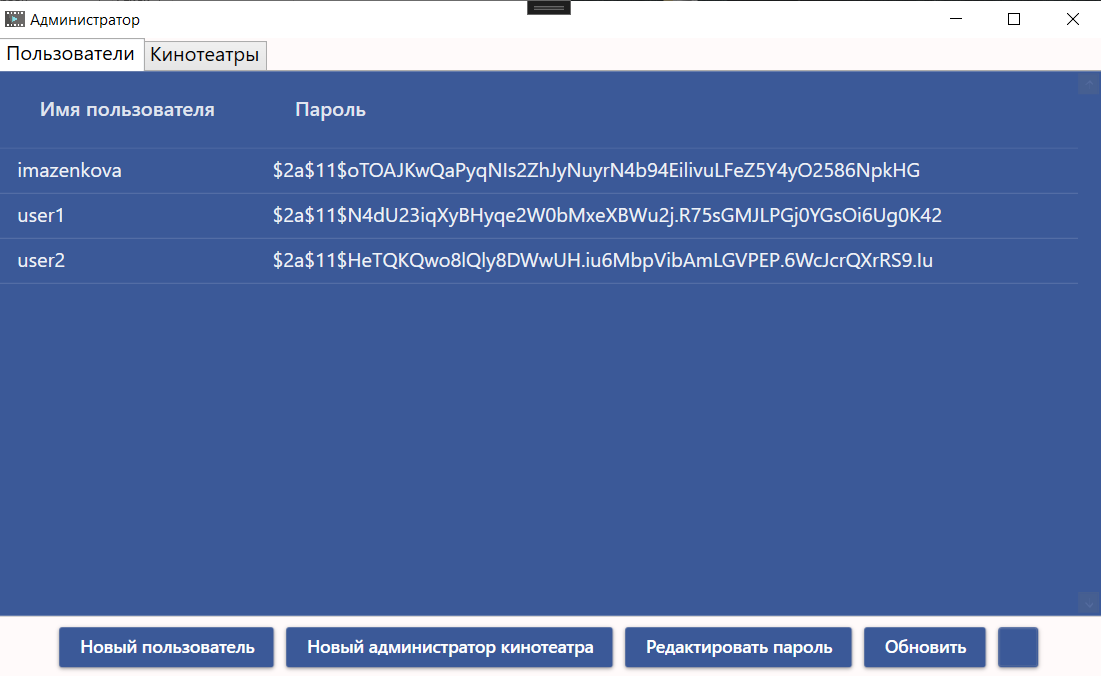


Рисунок 6.7 – Окно администратора приложения

Доступна информация по пользователям и администраторам кинотеатров.

Администратор приложения может удалять пользователей, добавлять новых или менять пароль текущим. Об успешных операциях изменений администратор будет уведомлен. После следует нажать на кнопку “обновить”, для получения последней информации из базы данных.

Пример окна создания администратора кинотеатра приведен на рисунке 6.8.

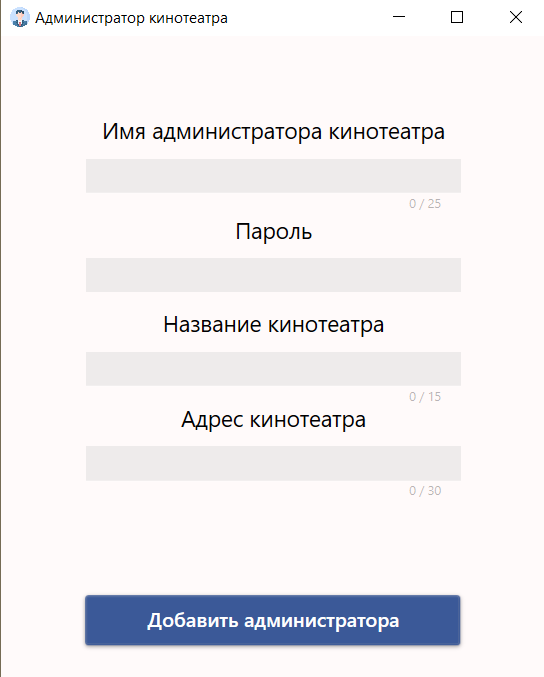


Рисунок 6.8 – Окно создания администратора кинотеатра

При создании нового администратора кинотеатра вводится информация по кинотеатру и автоматически создается зал с вместимостью 100. Администратор кинотеатра может позже добавить новые залы.

Кнопка возврата на окно авторизации находится справка от кнопки обновить.

* 1. **Использование приложения администратором кинотеатра**

Каждому администратору кинотеатра соответствует один кинотеатр.

Зарегистрироваться самостоятельно администратор кинотеатра не может. Добавить его может только администратор приложения. Вход в аккаунт осуществляется через окно авторизации.

После входа откроется главное окно, пример которого показан на рисунке 6.9.



Рисунок 6.9 – Окно администратора кинотеатра

Администратору кинотеатра отображается информация по всем фильмам, которые доступны для показов. Также администратор кинотеатра видит всю информацию по созданным билетам для своего кинотеатра и заказанным билетам пользователями. Администратор кинотеатра может добавить новый фильм для проката, создать сеанс, где в зависимости от вместимости выбранного зала будет формироваться количество билетов. Также можно добавить новый зал, где указывается его название и вместимость. Доступно удаление фильма, после которого удаляются все сеансы с этим фильмом.

После изменения данных надо нажать “обновить страницу”, чтобы увидеть последние обновления. Кнопка возврата на окно авторизации находится справка от кнопки обновить.

**ЗАКЛЮЧЕНИЕ**

В итоге выполнения данного курсового проекта было разработано про­граммное средство «Афиша», представляющее возможность заказа билетов в кинотеатр. В процессе выбора фильма пользователь может воспользоваться гибким поиском или сортировкой по жанрам, что облегчает выбор и экономит время. В личном кабинете пользователь может посмотреть все заказанные билеты или поменять своё имя пользователя. Так же приложение поддерживает администратора и администраторов кинотеатров, которые осуществляют возможность создания фильмов, сеансов и управление пользователями, например, удаление или смену пароля.

Благодаря этой платформы, кинотеатры без проблем могут добавлять новые фильмы в прокат, а также сообщать о кинопремьерах.

В процессе реализации программного средства были достигнуты выше указанные требования и задачи. Приложение было успешно протестировано, что означает пригодность этого приложения для своей цели.

Также в процессе выполнения данного курсового проекта были закреплены навыки в программировании на языке C#, создании приложений на WPF, использование Entity Framework 6, работа с современным паттерном MVVM, проектирование базы данных и реализация их в СУБД MS SQL Server 2018.

Для разработки дизайна использовалась библиотека MaterialDesign, которая имеет огромное количество красивых и удобных элементов управления.

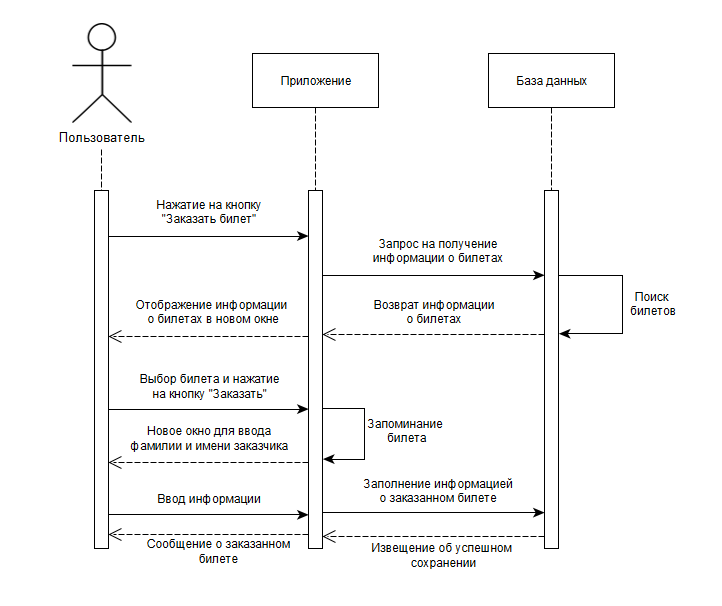
Для хранения исходного кода и удобства контроля версий проекта использовался крупнейший веб-сервис для совместной разработки – GitHub. Ссылка на проект: (https://github.com/imazenkova/MoviesAfisha).

**СПИСОК ИСПОЛЬЗОВАННОЙ ЛИТЕРАТУРЫ**

1. MSDN сеть разработчиков в Microsoft [Электронный ресурс] – <http://msdn.microsoft.com/library/rus/>. Дата доступа: 10.05.2018
2. WPF | Определение паттерна MVVM [Электронный ресурс] – <https://metanit.com/sharp/wpf/22.1.php>. Дата доступа: 18.10.2016
3. Пацей, Н.В. Курс лекций по языку программирования С# / Н.В. Пацей. – Минск: БГТУ, 2016. – 175 с.
4. Generic Repository | Entity Framework 6 [Электронный ресурс] – <https://metanit.com/sharp/entityframework/3.13.php>. Дата доступа: 13.10.2016
5. ProfessorWeb .NET & Web Programming [Электронный ресурс] / Режим доступа: <https://professorweb.ru> Дата доступа: 13.04.2018

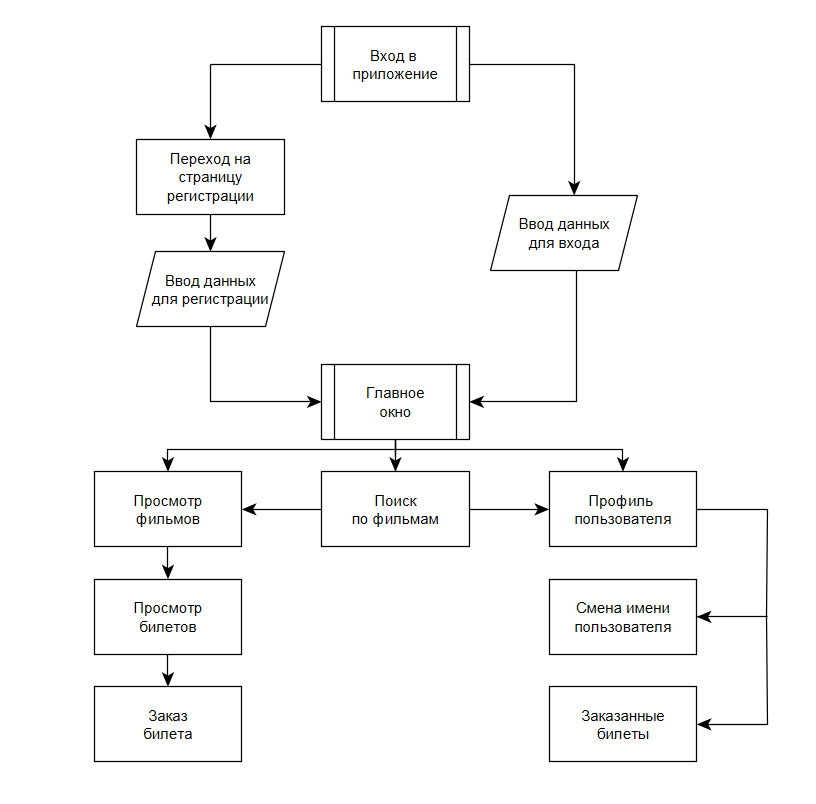
# ПРИЛОЖЕНИЕ А

**Диаграмма последовательности заказа билета**



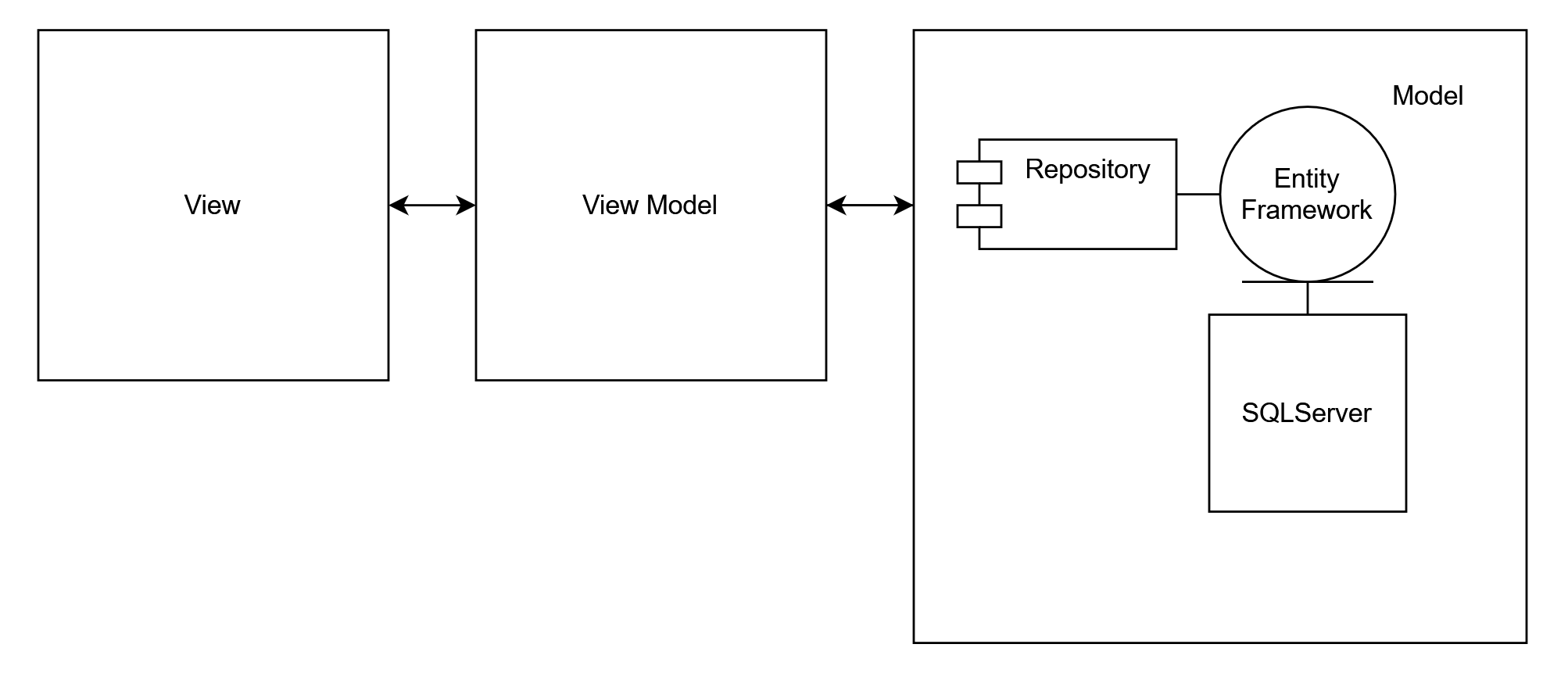
# ПРИЛОЖЕНИЕ Б

**Диаграмма деятельности пользователя**



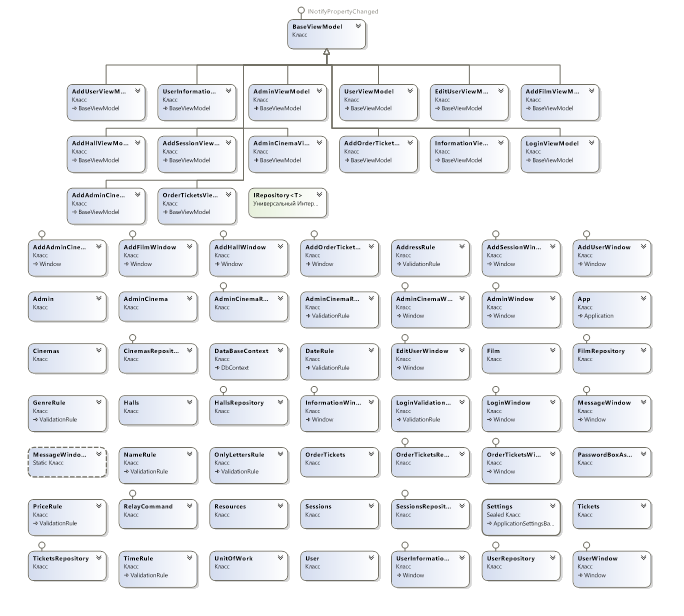
# ПРИЛОЖЕНИЕ В

**Диаграммы компонентов**



# ПРИЛОЖЕНИЕ Г

**Диаграмма всех классов**



# ПРИЛОЖЕНИЕ Д

**Листинг реализации и использования команд**

public class RelayCommand : ICommand

{

private Action<object> \_execute;

private Func<object, bool> \_canExecute;

public event EventHandler CanExecuteChanged

{

add { CommandManager.RequerySuggested += value; }

remove { CommandManager.RequerySuggested -= value; }

}

public RelayCommand(Action<object> execute, Func<object, bool> canExecute = null)

{

\_execute = execute;

\_canExecute = canExecute;

}

public bool CanExecute(object parameter)

{

return \_canExecute == null || \_canExecute(parameter);

}

public void Execute(object parameter)

{

\_execute(parameter);

}

}

private RelayCommand addOrderTicketWindow = null;

public RelayCommand AddOrderTicketWindow

{

get

{

return addOrderTicketWindow ?? new RelayCommand(obj =>

{

try

{

if (SelectedTicket == null)

{

MessageWindowCommand.ShowMessageToUser("Выберите билет");

}

else

{

OpenAddOrderTicketWindowCommand(SelectedTicket, Username);

}

}

catch (Exception e)

{

MessageWindowCommand.ShowMessageToUser(e.Message);

}

});

}

}

# ПРИЛОЖЕНИЕ Е

**Листинг регистрации пользователя**

try {

string Username = UsernameRegister.Text;

string Password = PasswordRegister.Password;

string resultStr = "";

if (Username == "" || Username.Replace(" ", "").Length == 0)

{

MessageWindowCommand.ShowMessageToUser("Введите имя пользователя");

}

else if (Password == "" || Password.Replace(" ", "").Length == 0)

{

MessageWindowCommand.ShowMessageToUser("Введите пароль");

}

else

{

UnitOfWork unitOfWork = new UnitOfWork();

User user = new User

{

Username = Username,

Password = BCrypt.Net.BCrypt.HashPassword(Password)

};

resultStr = unitOfWork.User.RegisterUser(user);

MessageWindowCommand.ShowMessageToUser(resultStr);

}

if (resultStr.Equals("Сделано!\nТеперь вы можете зайти в аккаунт:)"))

{

registerWindow.Visibility = Visibility.Collapsed;

loginWindow.Visibility = Visibility.Visible;

}

}

catch (Exception ex)

{

MessageWindowCommand.ShowMessageToUser(ex.Message);

}

# Приложение Ж

**Листинг авторизации пользователя**

Window window = GetWindow(this);

try

{

List<User> allUsers = new UnitOfWork().User.GetAll();

List<AdminCinema> allAdminCinemas = new UnitOfWork().AdminCinema.GetAll();

List<Admin> admin;

using (DataBaseContext db = new DataBaseContext())

{

admin = db.Admin.ToList();

}

bool isUser = false; ;

bool isAdminCinema = false;

string Username = UsernameLogin.Text;

string Password = PasswordLogin.Password;

if (Username == "" || Username.Replace(" ", "").Length == 0)

{

MessageWindowCommand.ShowMessageToUser("Введите имя пользователя");

}

else if (Password == "" || Password.Replace(" ", "").Length == 0)

{

MessageWindowCommand.ShowMessageToUser("Введите пароль");

}

else

{

#region if your are admin

if (Username == admin[0].Username)

{

if (BCrypt.Net.BCrypt.Verify(Password, admin[0].Password))

{

AdminWindow adminWindow = new AdminWindow

{

WindowStartupLocation = WindowStartupLocation.CenterScreen

};

window.Hide();

adminWindow.ShowDialog();

window.Close();

}

}

#endregion

#region if your are user

foreach (User user in allUsers)

{

if (Username == user.Username)

{

if (BCrypt.Net.BCrypt.Verify(Password, user.Password))

isUser = true;

}

}

foreach (AdminCinema adminCinema in allAdminCinemas)

{

if (Username == adminCinema.Username)

{

if (BCrypt.Net.BCrypt.Verify(Password, adminCinema.Password))

isAdminCinema = true;

}

}

if (isUser)

{

UserWindow userWindow = new UserWindow(Username)

{

WindowStartupLocation = WindowStartupLocation.CenterScreen

};

window.Hide();

userWindow.ShowDialog();

window.Close();

}

else if (isAdminCinema)

{

AdminCinemaWindow adminCinemaWindow = new AdminCinemaWindow(Username)

{

WindowStartupLocation = WindowStartupLocation.CenterScreen

};

window.Hide();

adminCinemaWindow.ShowDialog();

window.Close();

}

else if (!isAdminCinema || !isUser)

{

MessageWindowCommand.ShowMessageToUser("Неверное имя пользователя или пароль");

}

#endregion

}

}

catch (Exception ex)

{

MessageWindowCommand.ShowMessageToUser(ex.Message);

}

# ПРИЛОЖЕНИЕ З

**Листинг заказа билета**

try

{

Window window = obj as Window;

string resultStr = "";

if (Name == null || Name.Replace(" ", "").Length == 0)

{

SetRedBlockControll(window, "NameBox");

}

else if (Surname == null || Surname.Replace(" ", "").Length == 0)

{

SetRedBlockControll(window, "SurnameBox");

}

else

{

UnitOfWork unitOfWork = new UnitOfWork();

User user = unitOfWork.User.GetAll().FirstOrDefault(el => el.Username == Username);

OrderTickets newOrderTicket = new OrderTickets

{

Name = this.Name,

Surname = this.Surname,

IdTicket = Ticket.Id,

IdUser = user.Id

};

resultStr = unitOfWork.OrderTicket.Create(newOrderTicket);

Tickets orderTicket = unitOfWork.Ticket.GetAll().FirstOrDefault(el => el.Id == Ticket.Id);

if (orderTicket != null)

{

orderTicket.Id = Ticket.Id;

orderTicket.IDSession = Ticket.IDSession;

orderTicket.Place = Ticket.Place;

orderTicket.Price = Ticket.Price;

orderTicket.Row = Ticket.Row;

orderTicket.Availability = false;

unitOfWork.OrderTicket.Save();

resultStr = "Билет заказан";

}

MessageWindowCommand.ShowMessageToUser(resultStr);

window.Close();

}

}

catch(Exception e)

{

MessageWindowCommand.ShowMessageToUser(e.Message);

}