МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

Учреждение образования «БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ

ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ»

Факультет Информационных Технологий

Кафедра Программной инженерии

Специальность 1-98 01 03 Программное обеспечение информационной безопасности мобильных систем

Направление специальности 1-98 01 03 Программное обеспечение информационной безопасности мобильных систем

**ПОЯСНИТЕЛЬНАЯ ЗАПИСКА**

**КУРСОВОГО ПРОЕКТА:**

по дисциплине «Современные технологии программирования мобильных систем»

Тема Программное средство для бронирования билетов в кинотеатре

Исполнитель

студент (ка) 2 курса группы 8 Баланцевич Максим Сергеевич

(Ф.И.О.)

Руководитель работы ассистент Панченко О.Л.

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

Курсовой проект защищен с оценкой

Председатель Пацей Н.В.

(подпись)

Минск 2021

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

“\_\_\_”\_\_\_\_\_\_\_\_\_\_\_\_\_\_2021г.

**ЗАДАНИЕ**

**к курсовому проектированию**

**по дисциплине** "Современные технологии программирования мобильных систем"

|  |  |
| --- | --- |
| Специальность: 1-98 01 03 Программное обеспечение информационной безопасности мобильных систем  Студент: Баланцевич М.С. | Группа: 8 |
| **Тема: Программное средство для бронирования билетов в кинотеатр** | |

**1. Срок сдачи студентом законченной работы**: "20 мая 2020 г."

**2. Исходные данные к проекту:**

**2.1**. Функционально ПС поддерживает:

* Функции администратора производства:
  + Поддерживать работу с базой данных;
  + Оповещать пользователя о успешном заказе
  + Выполнять поисковые запросы;
* Функции клиента:
  + Выполнять регистрацию и авторизацию;
  + Возможность выбора места в зале;
  + Просматривать информацию о выбранном фильме;

**2.2.** При выполнении курсового проекта необходимо использовать принципы и приемы 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 должна быть представлена в формате docx. Оформление записки должно быть согласно выданным правилам.
  + Листинги программы представляются частично в приложении.
  + Пояснительную записку, листинги, проект (инсталляцию проекта) необходимо загрузить на диск, указанный преподавателем.

#### Календарный план

|  |  |  |  |
| --- | --- | --- | --- |
| №  п/п | Наименование этапов курсового проекта | Срок выполнения этапов проекта | Примечание |
| 1 | Введение | 19.02.2020 |  |
| 2 | Аналитический обзор литературы по теме проекта. Изучение требований, определение вариантов использования | 12.03.2020 |  |
| 3 | Анализ и проектирование архитектуры приложения (построение диаграмм, проектирование бизнес-слоя, представления и данных) | 26.03.2020 |  |
| 4 | Проектирование структуры базы данных. Разработка дизайна пользовательского интерфейса | 2.04.2020 |  |
| 5 | Кодирование программного средства | 23.04.2020 |  |
| 6 | Тестирования и отладка программного средства | 30.04.2020 |  |
| 7 | Оформление пояснительной записки | 7.05.2020 |  |
| 9 | Сдача проекта | 20.05.2020 |  |

**5. Дата выдачи задания** 12.02.2020 г.

Руководитель \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ *О. Л. Панченко*

(подпись)

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

(дата и подпись студента)

Содержание

[ВВЕДЕНИЕ 6](#_Toc41541454)

[1 Аналитический обзор прототипов и литературных источников 7](#_Toc41541455)

[2 Анализ требований к программному средству и разработка функциональных требований 10](#_Toc41541456)

[3 Проектирование программного средства 12](#_Toc41541457)

[3.1 Проектирование архитектуры приложения 12](#_Toc41541458)

[3.2 Проектирование базы данных 15](#_Toc41541459)

[3.3 Проектирование доступа к базе данных 18](#_Toc41541460)

[3.4 Проектирование логики сценариев использования 18](#_Toc41541461)

[3.5 Проектирование доступа к сервисам приложения 20](#_Toc41541462)

[4 Реализация программного средства 21](#_Toc41541463)

[4.1 Реализация сущностей 21](#_Toc41541464)

[4.2 Реализация архитектуры MVVM 22](#_Toc41541465)

[4.3 Реализация доступа к сервисам приложения 23](#_Toc41541466)

[4.4 Реализация представления 24](#_Toc41541467)

[5 Тестирование, проверка работоспособности и анализ полученных результатов 25](#_Toc41541468)

[6 Руководство по использованию 32](#_Toc41541469)

[ЗАКЛЮЧЕНИЕ 36](#_Toc41541470)

[СПИСОК ЛИТЕРАТУРЫ 37](#_Toc41541471)

[ПРИЛОЖЕНИЕ А 38](#_Toc41541472)

[ПРИЛОЖЕНИЕ Б 39](#_Toc41541473)

[ПРИЛОЖЕНИЕ В 43](#_Toc41541474)

[ПРИЛОЖЕНИЕ Г 44](#_Toc41541475)

[ПРИЛОЖЕНИЕ Д 47](#_Toc41541476)

ВВЕДЕНИЕ

Данный курсовой проект посвящён разработке программного средства для бронирования билетов в кинотеатр, основной целью которого является автоматизация работы кинотеатра. В настоящее время существует малое количество аналогов, реализованных в качестве десктопных приложений. Большинство таких сервисов представлено в виде Web-сайтов.

Приложение «Palace of Arts» - инструмент позволяющий вам ознакомиться со списком актуальных фильмов, без особых усилий заказать билет на удобную для вас дату, посмотреть ваши билеты.

Также, большое внимание я уделил дизайну. Так как в настоящее время, чтобы выделится среди конкурентов, необходимо давать пользователям что-то новое и интересное, ведь большинство приложений имеют схожий функционал.

В качестве интерфейса приклaдного программирования был выбран обширный API-интерфейс — Windows Presentation Foundation (WPF), предназначенный для создания настольных программ с графически насыщенным пользовательским интерфейсом.

Для работы с WPF использовался объектно-ориентированный язык программирования с С-подобным синтаксисом — С#, разработанный для создания приложений на платформе Microsoft .NET Framework.

Хранение данных осуществляется в Microsoft SQL Server.

1 Аналитический обзор прототипов и литературных источников

Важным этапом в разработке программного продукта является аналитический сбор информации прототипов и различных источников информации с темой разработки.

На сегодняшний день приложения для бронирования билетов в кинотеатр довольно распространены, но малое количество имеет интуитивно понятный интерфейс.

Рассмотрим наиболее популярные программы для бронирования автомобилей:

«Киноход» - это ПО для бронирования билетов в разных кинотеатрах. Интерфейс «Киноход» представлен на рисунке 1.1, 1.2

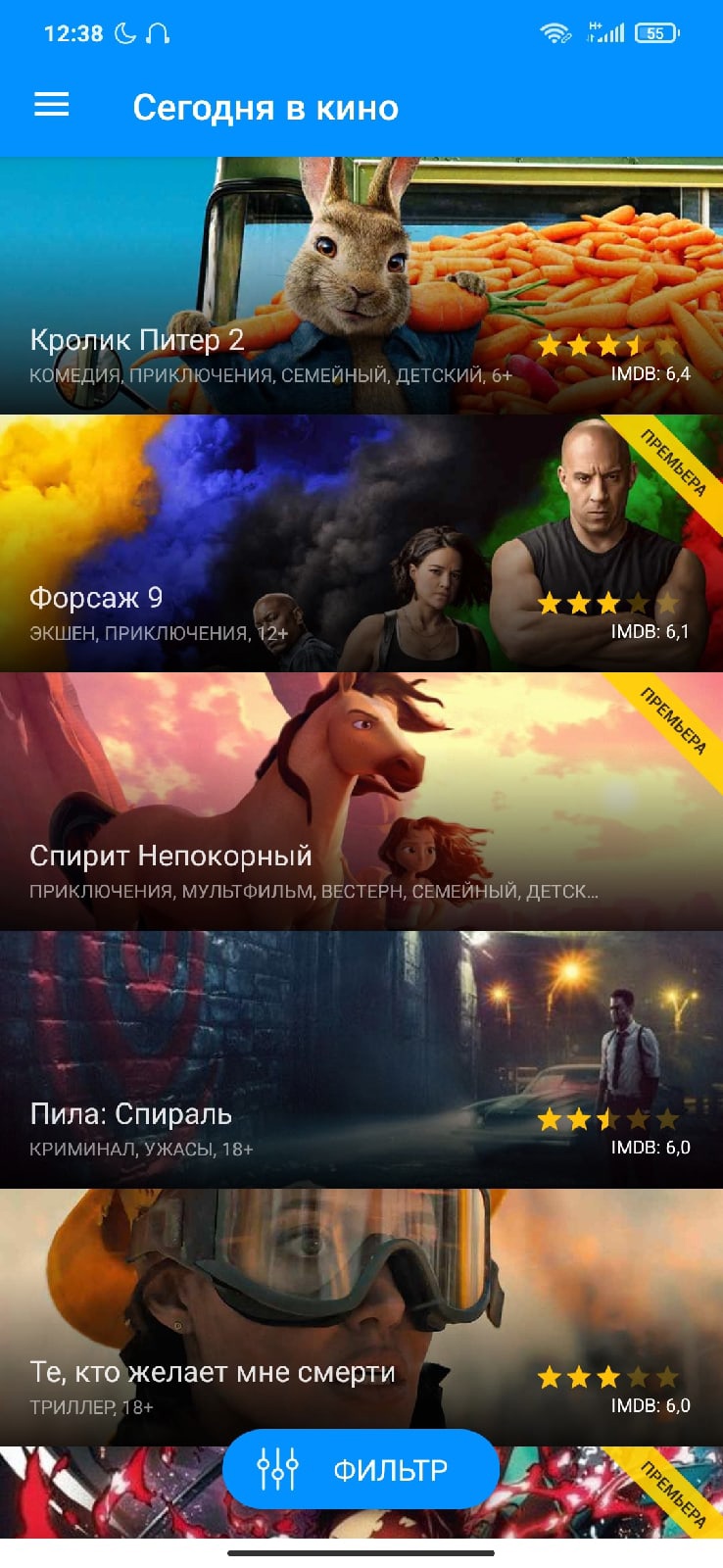


Рис. 1.1 – «Киноход» сегодня в кино

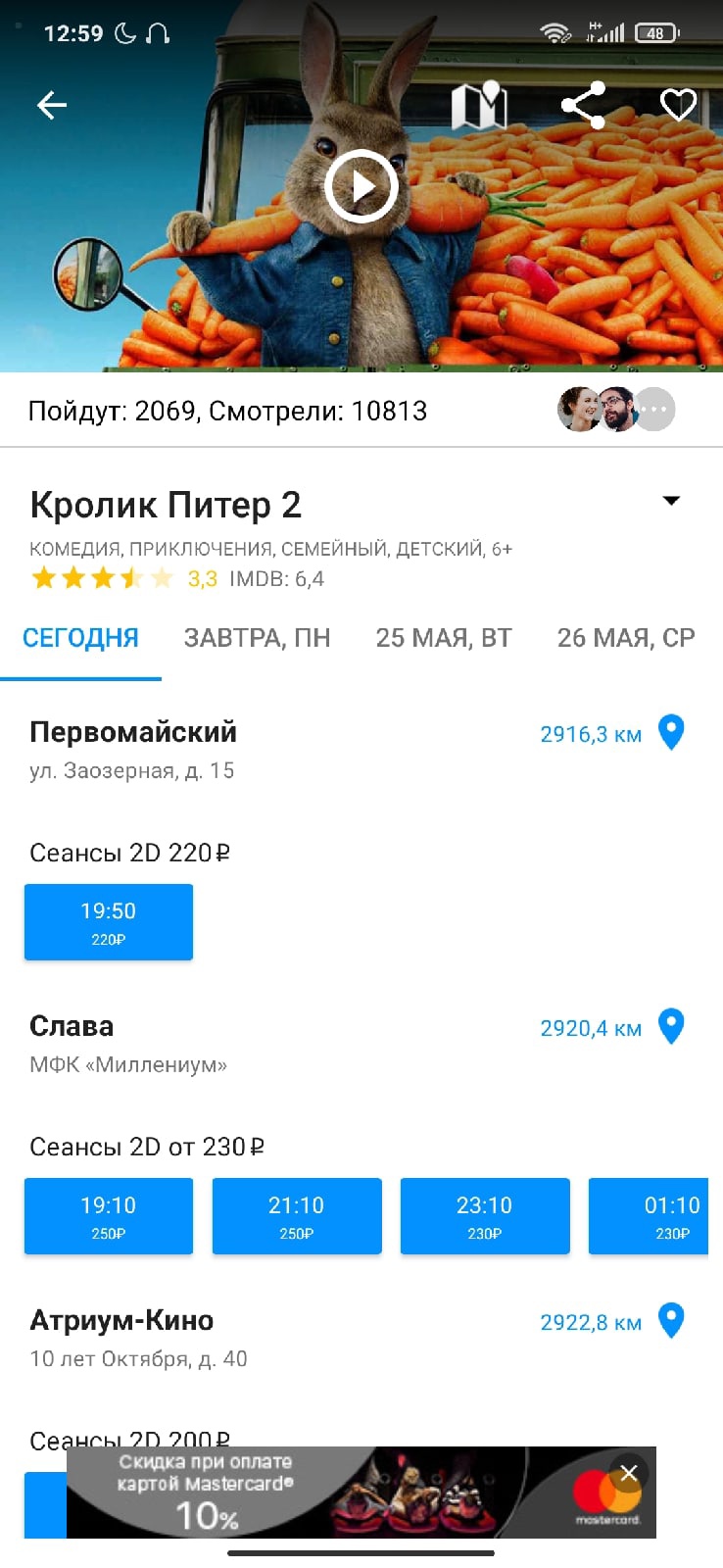


Рис. 1.2 – «Киноход» информация о фильме

Проанализировав «Киноход», можно выделить основные минусы данного программного сред.

Основные минусы:

* Загруженный интерфейс;
* Отсутствие подробной информации о фильме;
* Отсутствует информация о кинотеатре.

Основные плюсы:

* Неплохой UI;
* Большой выбор кинотеатров;

Продукт «Кинотеатр» - cервис бронирования билетов в кинотеатр

Интерфейс «Кинотеатр» представлен на рисунке 1.3, 1.4



Рис. 1.3 – «Кинотеатр» главная страница

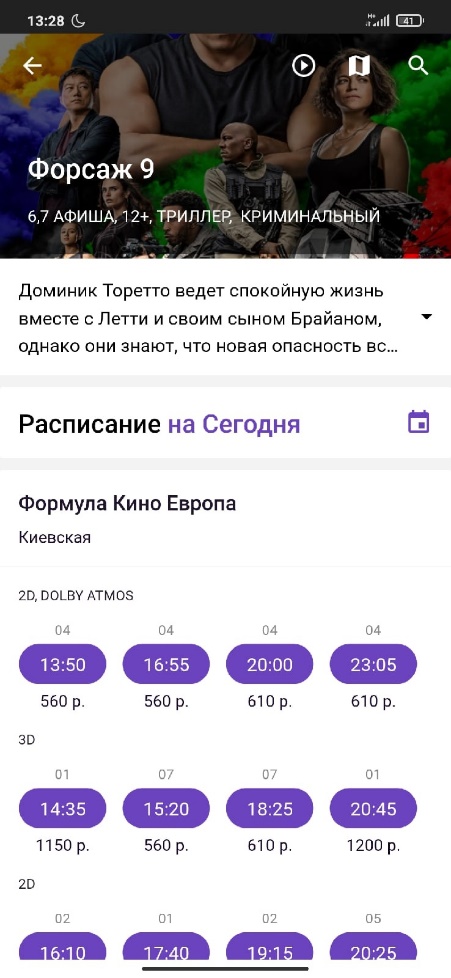


Рис. 1.3 – «Кинотеатр» главная страница

Проанализировав ПО «Кинотеатр», можно выделить её основные минусы и плюсы.

Основные минусы:

* Малое количество информации о фильмах и кинотеатрах;
* Отсутствие поиска;
* Отсутствие фильтрации;

Основные плюсы:

* Простота в использовании;
* Простые и приятные цвета;

2 Анализ требований к программному средству и разработка функциональных требований

Анализ требований — это процесс сбора требований к программному обеспечению, их систематизации, документирования, анализа, выявления противоречий, неполноты, разрешения конфликтов в процессе разработки программного обеспечения.

Цель анализа требований в проектах — получить максимум информации о заказчике и специфике его задач, уточнить рамки проекта, оценить возможные риски. На этом этапе происходит идентификация принципиальных требований методологического и технологического характера, формулируются цели и задачи проекта, а также определяются критические факторы успеха, которые впоследствии будут использоваться для оценки результатов внедрения. Определение и описание требований — шаги, которые во многом определяют успех всего проекта, поскольку именно они влияют на все остальные этапы.

Различают три уровня требований к проекту:

* бизнес-требования;
* пользовательские требования;
* функциональные требования.

Бизнес-требования содержат высокоуровневые цели организации или заказчиков системы. Как правило, их высказывают те, кто финансируют проект, покупатели системы, менеджер реальных пользователей, отдел маркетинга. Курсовой проект не подразумевает наличие заказчика, который мог бы выдвинуть бизнес-требования, поэтому в качестве таких высокоуровневых требований можно рассматривать общие требования к разрабатываемому средству. К их числу относятся:

* простота и лёгкость интерфейса;
* использование принципов объектно-ориентированного программирования;
* использование архитектурных шаблонов проектирования;
* использование системы управления базами данных (СУБД);

Весь дальнейший процесс проектирования и разработки программного средства должен находиться в очерченных бизнес-требованиями границах.

Следующими требованиями являются требования пользователей. Данные требования описывают цели и задачи, которые пользователям позволит решить система. Таким образом, в пользовательских требованиях указано, что клиенты смогут делать с помощью системы. Пользователь данного программного решения должен иметь возможность:

* Регистрироваться в системе;
* Ознакомиться с актуальными фильмами;
* Получить полную информацию о фильме;
* Производить поиск необходимого фильма;
* Просматривать предложенные ему билеты;
* Выбор билета с подходящим для него временем и местом;
* Забронировать билет;
* Просматривать забронированные билеты;
* Отменить бронь.

Администратор имеет возможность:

* вносить новую информацию о Фильмах, Залах, Сеансах;
* удалять информацию о Фильмах, билетах и сеансах;

После проведения анализа были выявлены следующие функциональные требования:

* архитектура приложения должна соответствовать шаблону проектирования, такому как MVVM;
* вся информация должна храниться в базе данных;
* приложение должно производить валидацию вводимых пользователем данных;
* приложение должно корректным образом обрабатывать возникающие исключительные ситуации: отображать понятное для пользователя сообщение о возникшей ошибке;
* приложение должно предоставлять пользователям возможность создания нового аккаунта в виде регистрационной формы;
* приложение должно предоставлять возможность пользователям проходить аутентификацию и входить в систему под соответствующим введенным данным;
* приложение должно предоставлять возможность поиска билетов по следующим критериям: название фильма и дата сеанса.
* Приложение должно уведомлять пользователя об успешном заказе билетов путем отправки сообщения на почту.

Таким образом, был проведен тщательный анализ требований к программному средству, который позволил разработать список функциональных требований. Разработка данной программной системы должна проводиться в соответствии с сформированными списком.

3 Проектирование программного средства

Проектирование программного средства — процесс создания проекта программного обеспечения. Целью проектировaния является определение внутренних свойств системы и детализации её внешних свойств на основе исходных условий задачи. Исходные условия задачи уже были сформулированы во втором разделе данной пояснительной записки. Этап проектирования подразумевает их анализ.

## **3.1 Проектирование архитектуры приложения**

Архитектура программного обеспечения — совокупность вaжнейших решений об организации программной системы. Архитектура включает:

* выбор структурных элементов и их интерфейсов, с помощью которых составлена система, а также их поведения в рамках сотрудничества структурных элементов;
* соединение выбрaнных элементов структуры и поведения во всё более крупные системы;
* архитектурный стиль, который направляет всю организацию — все элементы, их интерфейсы, их сотрудничество и их соединение.

Для удовлетворения проектируемой системы различным атрибутам качества применяются различные архитектурные шаблоны (паттерны). В разрабатываемом приложении используется архитектурный шаблон Model-View-ViewModel (MVVM).

Шаблон MVVM имеет три основных слоя: модель, которая представляет бизнес-логику приложения, представление пользовательского интерфейса, и представление-модель, в котором содержится вся логика построения графического интерфейса и ссылка на модель, поэтому он выступает в качестве модели для представления.

На рисунке 3.1 представлена диаграмма, которая показывает общую структуру приложения в рамках шаблона MVVM.



Рисунок 3.1 – Структура шаблона MVVM

Модель описывает используемые в приложении данные. Модели могут содержать логику, непосредственно связанную этими данными, например, логику валидации свойств модели. В то же время модель не должна содержать никакой логики, связанной с отображением данных и взаимодействием с визуальными элементами управления.

Нередко модель реализует интерфейсы INotifyPropertyChanged, который позволяет уведомлять систему об изменениях свойств модели. Благодаря этому облегчается привязка к представлению, хотя взаимодействие между моделью и представлением отсутствует.

View определяет визуальный интерфейс, через который пользователь взаимодействует с приложением. Интерфейс и качество его реализации занимает важную роль в конечном результате, разработка эффективного и понятного интерфейса, является важной задачей. Поэтому для создания качественного интерфейса необходимо понять, как пользователь будет взаимодействовать с приложением. Для этого была составлена схема (рис. 3.2), на которой представлен принцип работы приложения с точки зрения пользователя и администратора.

ViewModel или модель представления связывает модель и представление через механизм привязки данных. Если в модели изменяются значения свойств, при реализации моделью интерфейса INotifyPropertyChanged автоматически идет изменение отображаемых данных в представлении, хотя напрямую модель и представление не связаны.

ViewModel также содержит логику по получению данных из модели, которые потом передаются в представление. И также VewModel определяет логику по обновлению данных в модели. Поскольку элементы представления, то есть визуальные компоненты типа кнопок, не используют события, то представление взаимодействует с ViewModel посредством команд.

Для разработки приложения в качестве сервиса для хранения данных была выбрана СУБД Microsoft SQL Server. На диаграмме размещение на рисунке 3.2 показано графическое представление инфраструктуры приложения.

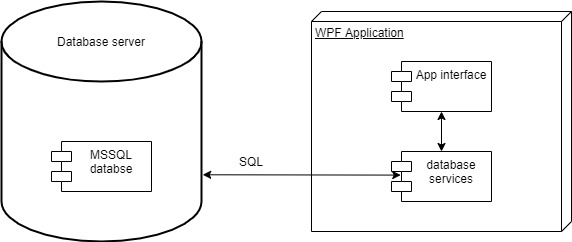


Рисунок 3.2 – Диаграмма размещения

3.2 Проектирование базы данных

Проектирование баз данных — процесс создания схемы базы данных и определения необходимых ограничений целостности.

Основные задачи проектирования базы данных:

* обеспечение хранения в БД всей необходимой информации;
* обеспечение возможности получения данных по всем необходимым запросам;
* сокращение избыточности и дублирования данных;
* обеспечение целостности базы данных.

Проектирование базы данных проводится в два этапа: концептуальное (инфологическое) и логическое (даталогическое) проектирование.

Проектирование базы данных проводится в два этапа: концептуальное (инфологическое) и логическое (даталогическое) проектирование.

Концептуальное (инфологическое) проектирование — построение семантической модели предметной области, то есть информационной модели наиболее высокого уровня aбстракции. В результате этого этапа создаётся ER-модель. Такая модель создаётся без ориентации на какую-либо конкретную СУБД и модель данных.

Основными понятиями ER-модели являются: сущность, связь и атрибут

Сущность – это реальный или представляемый объект, информация о котором должна сохраняться и быть доступна.

Связь – это графически изображаемая ассоциация, устанавливаемая между двумя сущностями. Эта ассоциация обычно является бинарной и может существовать между двумя разными сущностями или между сущностью и ей же самой (рекурсивная связь).

Атрибут сущности это любая детaль, которая служит для уточнения, идентификации, классификации, числовой характеристики или выражения состояния сущности.

В рамках этого этапа была создана ER-модель, которая включает 6 сущностей:

* фильм;
* зал;
* сеанс;
* билет;
* заказанные билеты;
* пользователь.

Также в ER-модели были определены необходимые связи. Например, между сущностями пользователь и заказанный билет была установлена связь один-ко-многим. Для каждой сущности были выделены атрибуты. Например, для Заказанные билеты в качестве атрибутов были выделены такие характеристики, как идентификатор Заказанного билета, идентификатор билета, идентификатор пользователя.

Логическое (даталогическое) проектирование — создание схемы базы данных на основе конкретной модели данных, например, реляционной модели данных. Для реляционной модели дaнных логическая модель — набор схем отношений, обычно с указанием первичных ключей, а также «связей» между отношениями, представляющих собой внешние ключи. На этапе логического проектирования учитывается специфика конкретной модели данных, но может не учитываться специфика конкретной СУБД.

Логическая модель базы данных представлена на рисунке 3.3

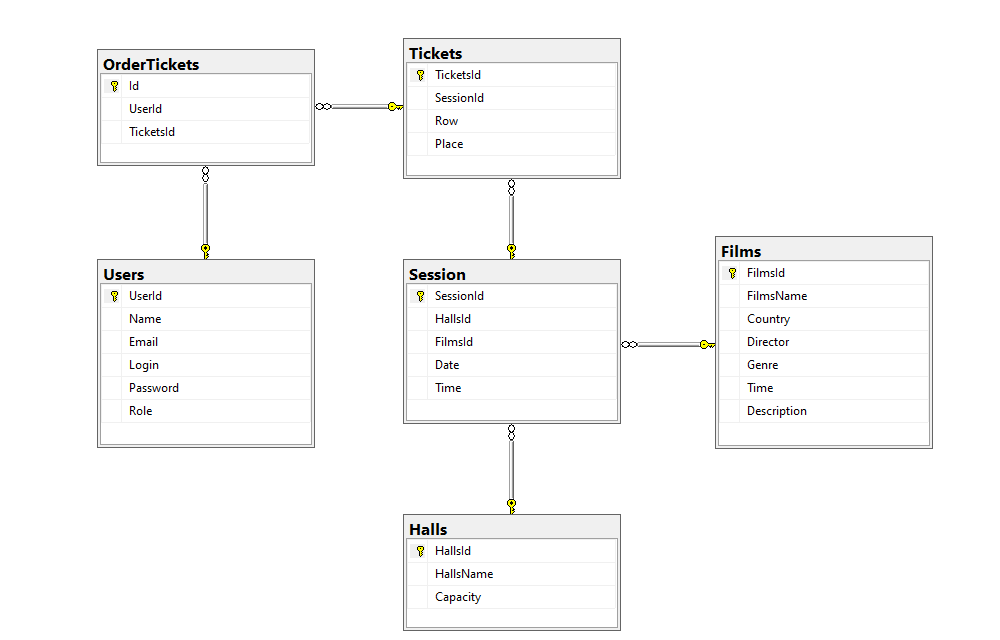


Рисунок 3.3 - Логическая модель базы данных

Всего в базе данных содержится 6 таблиц. В таблице Users хранятся все пользователи, зарегистрированные в приложении. Таблица OrderTickets содержит информация о билетах, которые заказал пользователь. В Таблице Tickets содержится информация о Билетах. Таблица Session содержит информация о возможных сеансах для фильмов. Таблица Films содержит информация о всех фильмов. Таблица Halls содержит информацию о залах кинотеатра.

Таблица Users состоит из 6 столбцов:

* UserId;
* Name;
* Email;
* Login;
* Password;
* Role.

В столбце Name хранится имя пользователя, Email - электронная почта зарегистрированного пользователя, Password – пароль пользователя, Role – является ли пользователь администратором.

Таблица OrderTickets состоит из 3 столбцов:

* Id;
* UserId;
* TicketsId.

В столбце UserId хранится идентификатор пользователя, TicketsId – идентификатор билета.

Таблица Tickets состоит из 4 столбцов:

* TicketsId;
* SessionId;
* Row;
* Place.

В столбце SessionId хранится идентификатор сеанса, в столбце Row – выбранный ряд, Place – место в зале.

Таблица Session состоит из 5 столбцов:

* SessionId;
* HallsId;
* FilmsId;
* Date;
* Time.

В столбце HallsId хранится идентификатор зала, в столбце FilmsId – идентификатор фильма, Date – дата сеанса, Time – Время сеанса.

Таблица Films состоит из 7 столбцов:

* FilmsId;
* FilmsName;
* Country;
* Director;
* Genre;
* Time;
* Description.

В столбце FilmsName хранится название фильма, в столбце Country – страна , Director – дата сеанса, Genre – Время сеанса.

Таблица Halls состоит из 3 столбцов:

* HallsId;
* HallsName;
* Capacity;

В столбце HallsName хранится название зала, в столбце Capacity – вместимость зала.

3.3 Проектирование доступа к базе данных

Для доступа к базе данных используется Entity Framework Core 6. Этот подход предоставляет ряд существенных преимуществ: нам не нужно беспокоиться о коде доступа к данным и не нужно знать деталей работы СУБД SQL Server и синтаксиса языка запросов T-SQL, вместо этого мы работаем с таблицами базы данных как с классами C#, с полями этих таблиц - как со свойствами классов, а синтаксис SQL-запросов заменен на  [LINQ](https://professorweb.ru/my/LINQ/base/level1/info_linq.php). Entity Framework Core берет на себя обязанности по преобразованию кода C# в SQL-инструкции.

Существуют 3 подхода по проектированию базы данных:

* Database-First;
* Model-First;
* Code-First.

В своей работе я использовал подход Code-First, при данном подходе модель EDMX не используется, и мы вручную настраиваете классы C# объектной модели (данный подход поддерживает как генерацию сущностных классов из существующей базы данных, так и создание базы данных из созданной вручную модели объектов C#).

3.4 Проектирование вариантов использования

На рисунке 3.4 представлена диаграмма использования приложения для пользователя и администратора. Данная диаграмма представлена в приложении Д.

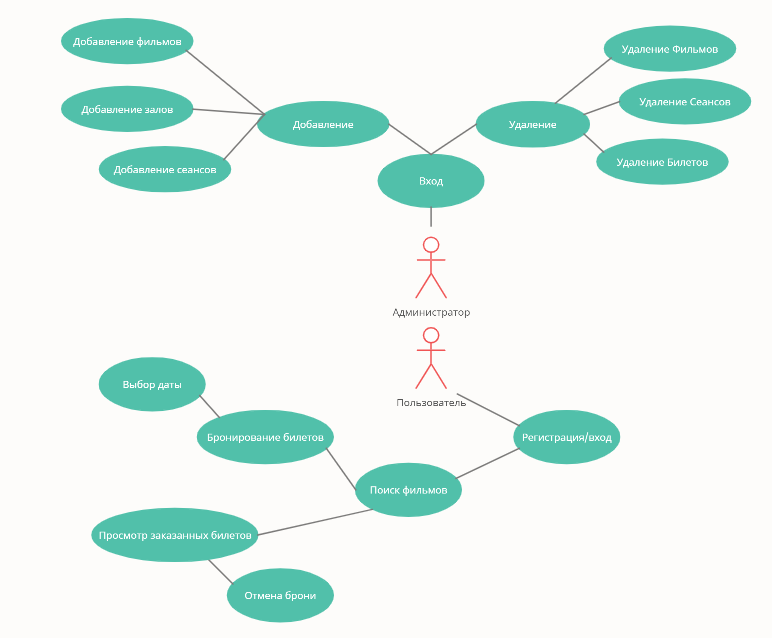


Рисунок 3.4 - Диаграмма вариантов использования для пользователя и администратора

После входа в приложение пользователю откроется окно «Каталога», где пользователь может ознакомиться со всеми актуальными фильмами, узнать подробную информацию, а также осуществить поиск и сортировку по алфавиту. Далее следует перейти в окно «Билеты», где, указав нужный фильм и дату выбрать билет.

При авторизации пользователя с ролью «Администратор» ему будет доступна панель администратора со всеми соответствующим функционалом.

Для администратора была создана отдельная ветвь окон, при помощи которых возможно добавление и удаление данных о фильмах, сеансах и залах.

4 Реализация программного средства

Следующим этапом разработки приложения является непосредственная реализация программного решения в соответствии с уже сформированными требованиями и шаблонами

4.1 Реализация сущностей

В соответствии с требованиями в качестве хранилища данных программного средства должна быть база данных, поэтому первым шагом в реализации программы является выбор технологии, позволяющей это осуществить. Выбор остановился на ORM технологии Entity Framework. Она предоставляет три подхода по проектированию базы данных. В данном программном решении был использован подход Code-First. При данном подходе модель EDMX не используется. Создание базы данных происходит из созданной вручную модели объектов C#. Созданные модели объектов совпадают с сущностями, которые были сформированы раннее в разделе 3.2.

Диаграмма классов UML для сущностных классов представлена на рисунке

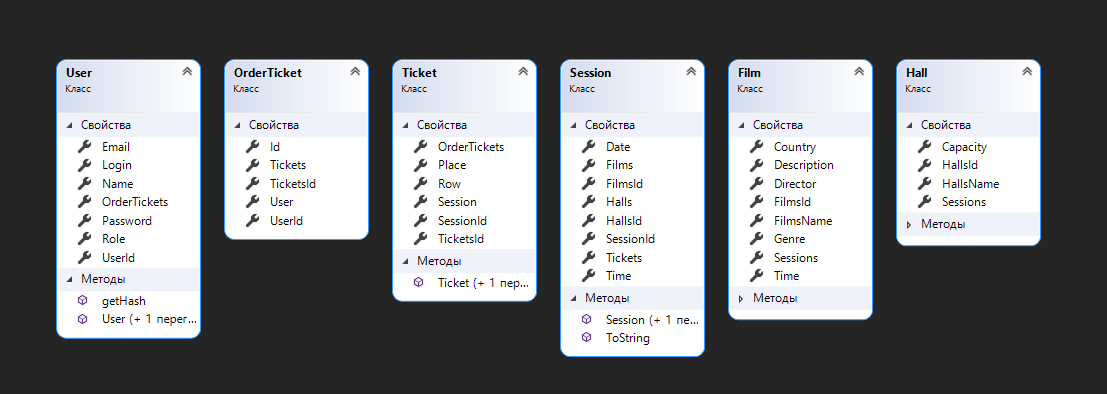


Рисунок 4.1 − Диаграмма классов

4.2 Реализация архитектуры MVVM

Проект построен на архитектуре MVVM, поэтому файлы проекта разделены на 3 логических модуля: Model, View и ViewModel.

Разделение проекта на логические модули представлено на рисунке 4.2.

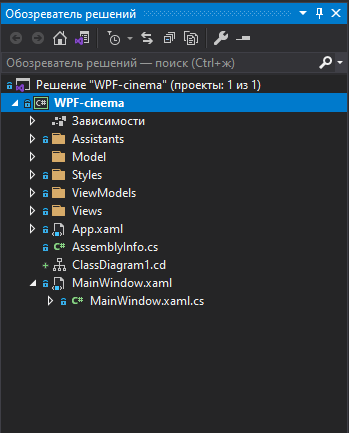


Рисунок 4.2 – Логические модули проекта

В папке Styles находятся картинки, использующиеся в приложении, стили для представлений.

В папке Model содержатся сущностные классы, которые используются для создания БД. Описание сущностных классов представлено в разделе 3.2.

В приложении используется паттерн Command который позволяет инкапсулировать запрос на выполнение определенного действия в виде отдельного объекта. В WPF команды представлены интерфейсом ICommand. . Данный класс представлен в листинге 4.1

using System;

using System.Windows.Input;

namespace WPF\_cinema.Assistants.Commands

{

abstract class Command : ICommand

{

public event EventHandler CanExecuteChanged

{

add => CommandManager.RequerySuggested += value;

remove => CommandManager.RequerySuggested -= value;

}

public abstract bool CanExecute(object parameter);

public abstract void Execute(object parameter);

}

}

Листинг 4.1 – Класс Command

Класс реализует два метода:

* CanExecute: определяет, может ли команда выполняться;
* Execute: собственно, выполняет логику команды.

В папке ViewModels находятся все классы ViewModel для каждого представления. Диаграмма классов UML для классов модуля ViewModel представлена на рисунке 4.1. Также данная диаграмма представлена в приложении Г.

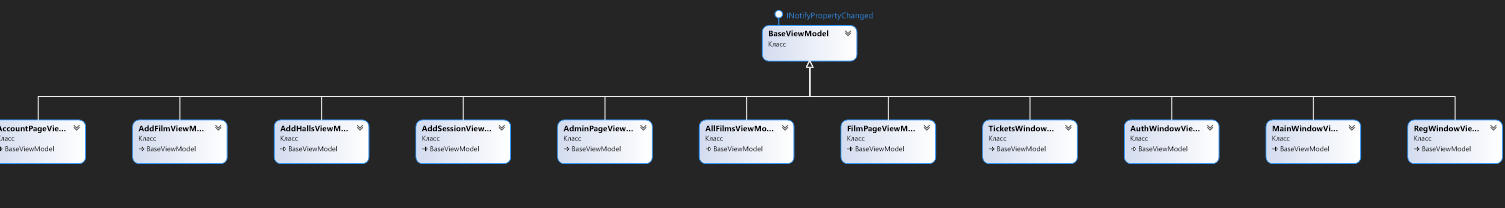


Рисунок 4.1 – Диаграмма классов модуля ViewModel

На диаграмме 4.1 видно, что все классы ViewModel наследуются от базового класса ViewModel – BaseViewModel.

В папке Views хранятся все используемые представления.

**4.3 Реализация авторизации и регистрации**

Для того, чтобы пользоваться приложением, необходимо для начала зарегистрироваться в системе. В приложении А представлен код реализующий регистрацию.

Если пользователь уже зарегистрирован, то следует войти в систему. В приложении Б представлен код реализующий вход в систему.

В базе данных хранятся захешированные пароли пользователей. Хеширование было произведено при помощи пространства using System.Security.Cryptography, который позволяет зашифровывать данные алгоритмом MD5

5 Тестирование, проверка работоспособности и анализ полученных результатов

В курсовом проекте задействуется обработка исключений, таким образом, что пользователь будет уведомлен о неудачном выполнении операции. Присутствуют различные всплывающие окна, выводящие сообщение об ошибке. На рисунках 5.1-5.3 приведены примеры обработки ошибок при неверно введенных данных в окне регистрации в приложение.

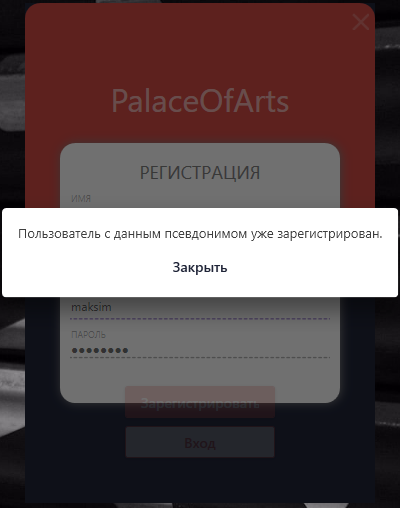


Рисунок 5.1 – Пользователь с таким логином уже зарегистрирован

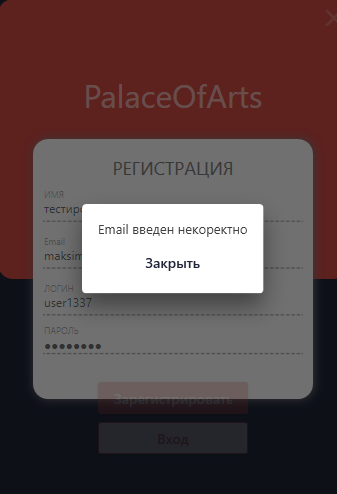


Рисунок 5.2 – Неверный формат Email

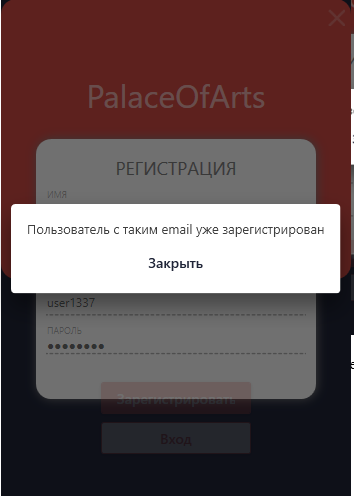


Рисунок 5.3 – Пользователь с таким email уже зарегистрирован

Также присутствует уведомление о других ошибках: например, неправильный формат данных или просьба выполнить какое-то действие, которое пользователь забыл. Примеры представлены на рисунке 5.4

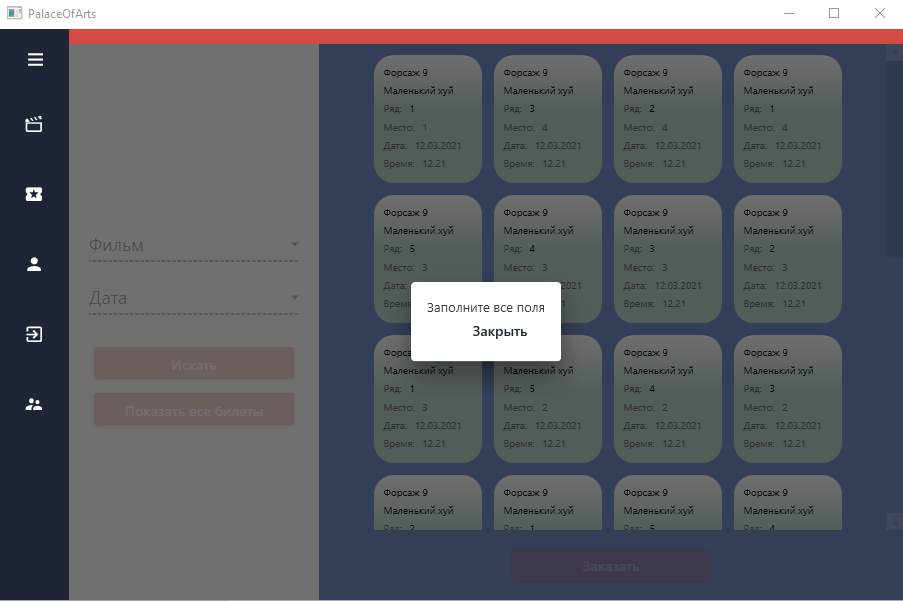


Рисунок 5.4 – Просьба заполнить все поля для поиска билета

6 Руководство по использованию

Для того, чтобы пользоваться приложением, необходимо для начала зарегистрироваться в системе. Окно регистрации показано на рисунке 6.1

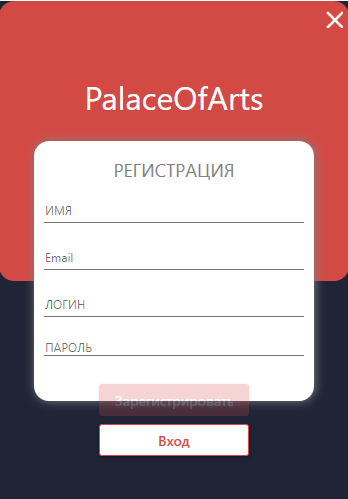


Рисунок 6.1 – Окно регистрации

Если у пользователя уже есть аккаунт, то нужно войти в систему. Окно входа показано на рисунке 6.2.

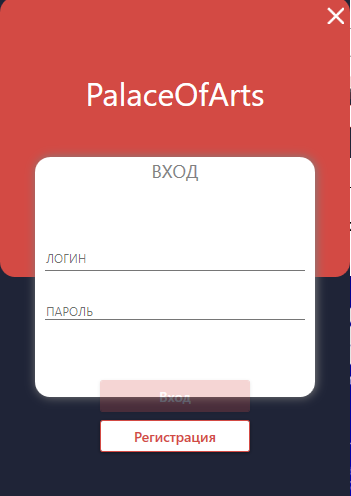


Рисунок 6.2 – Окно входа

6.1 Руководство по использованию администратором

После входа в систему администратору откроется главное окно, в котором ему станет доступна вкладка «Админ». Где находятся все необходимые элементы управления.

Рабочая область администратора и описанные элемент интерфейса представлены на рисунке 6.3 - 6.4

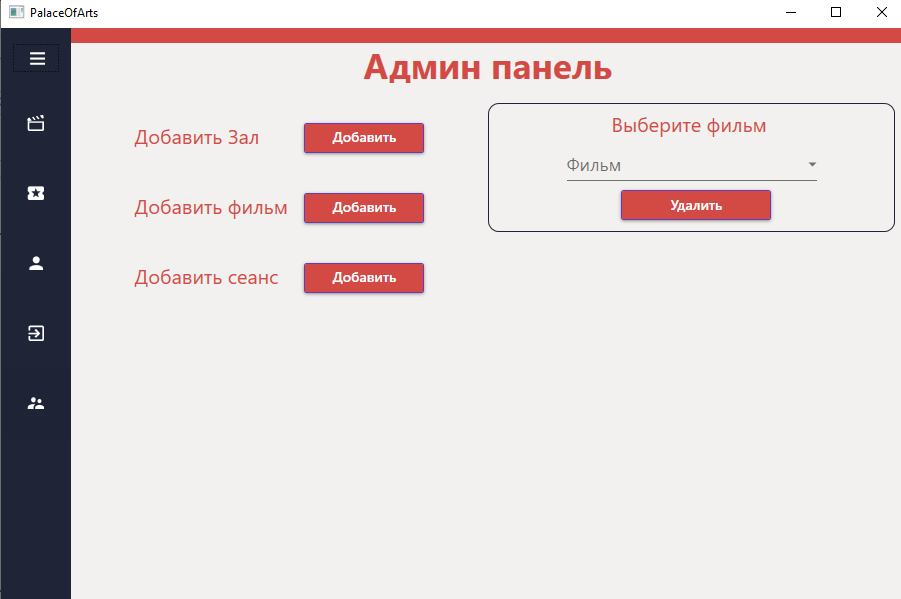


Рисунок 6.3 – Окно администратора

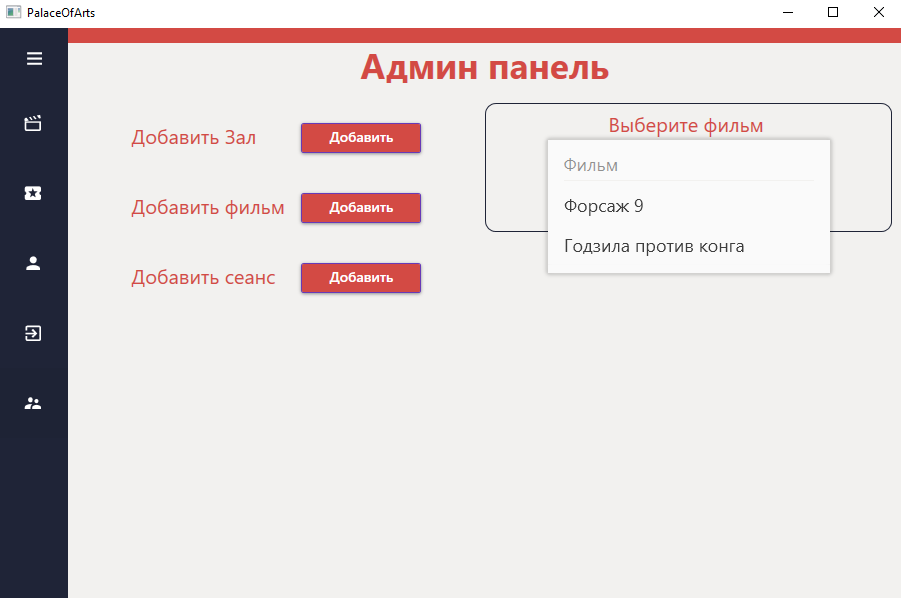


Рисунок 6.4 – Выбор фильма для удаления

При помощи кнопок «Добавить зал», «Добавить фильм» и «Добавить сеанс» администратор может добавить новый объект в базу данных. Представлено на рисунках 6.6 - 6.8

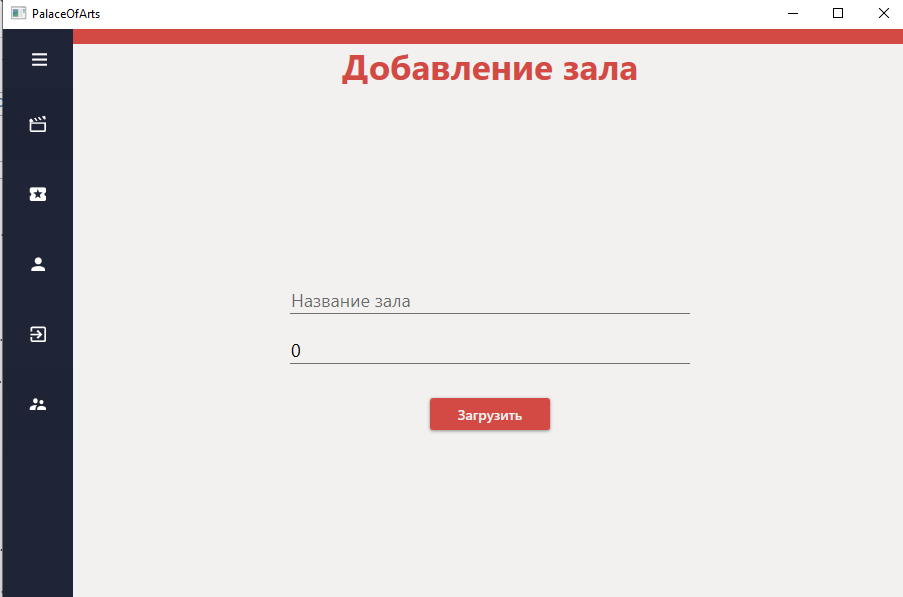


Рисунок 6.6 – Окно добавления зала

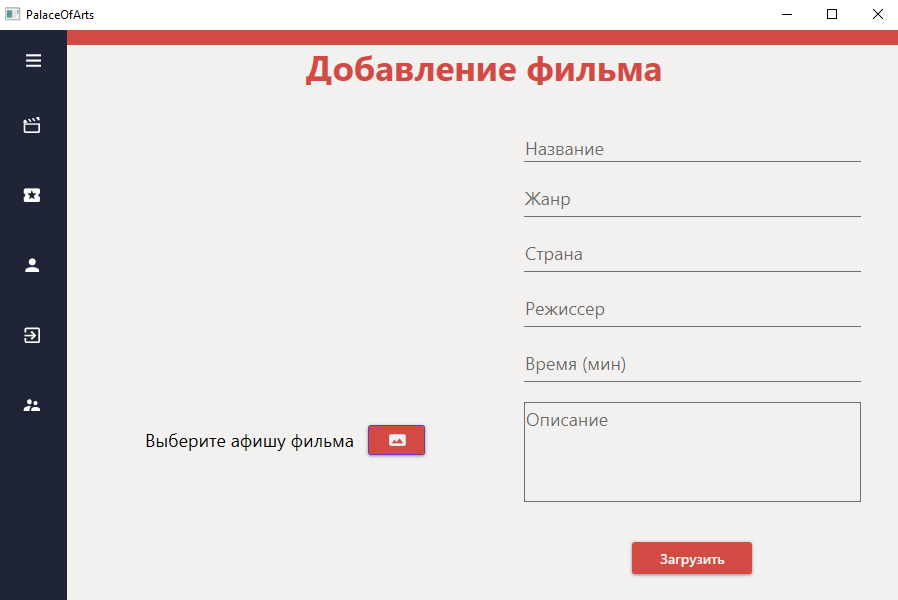


Рисунок 6.7 – Окно добавления фильма

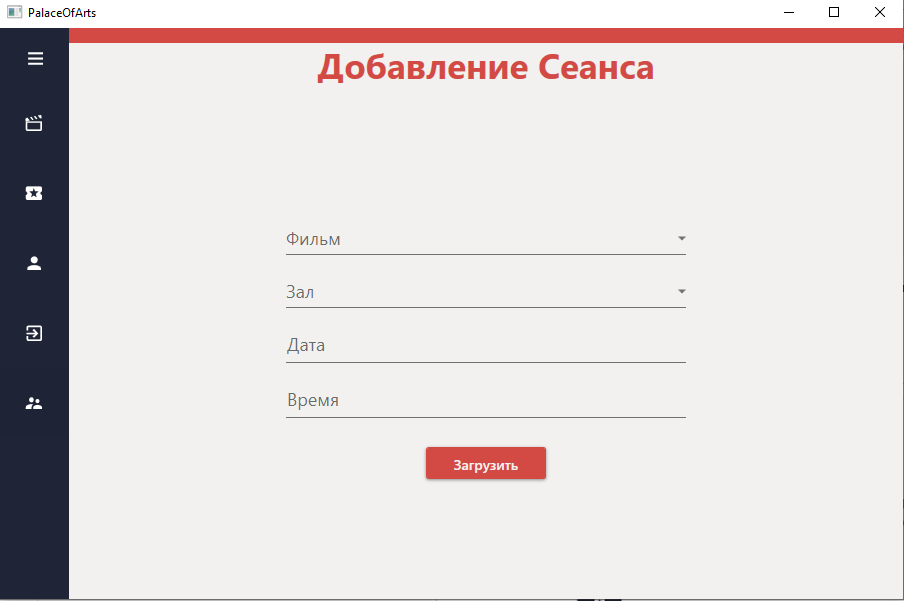


Рисунок 6.8 – Окно добавления сеанса

При нажатии на иконку выхода из аккаунта пользователь может вернуться к окну авторизации.

6.2 Руководство по использованию пользователем

После входа в систему пользователю откроется главное окно. Необходимо заполнить поля формы поиска, после чего выбрать интересующий авиарейс. Представлено на рисунке 6.9.

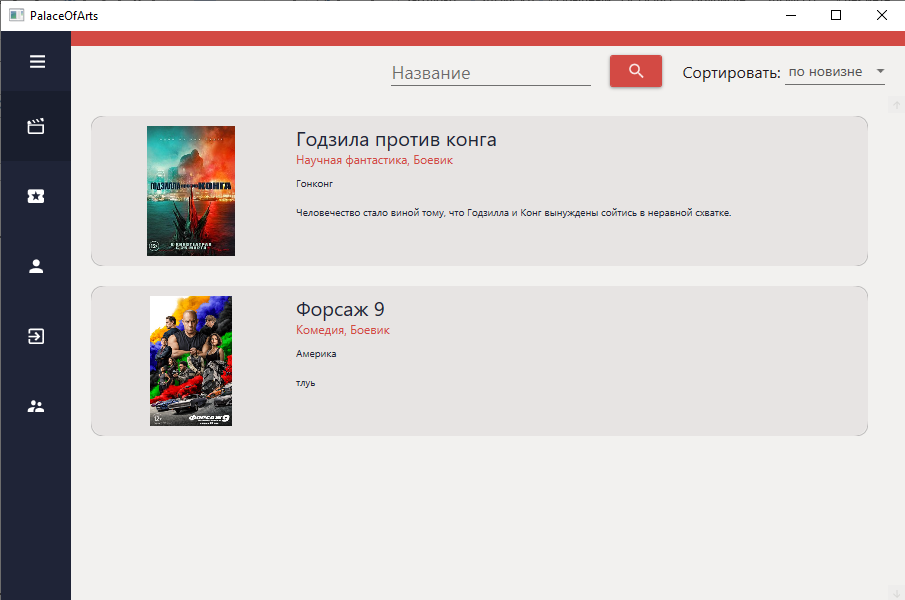


Рисунок 6.9 – Поиск фильма

Подробную информацию о фильме можно на отдельной странице каждого фильма. Представлено на рисунке 6.10.

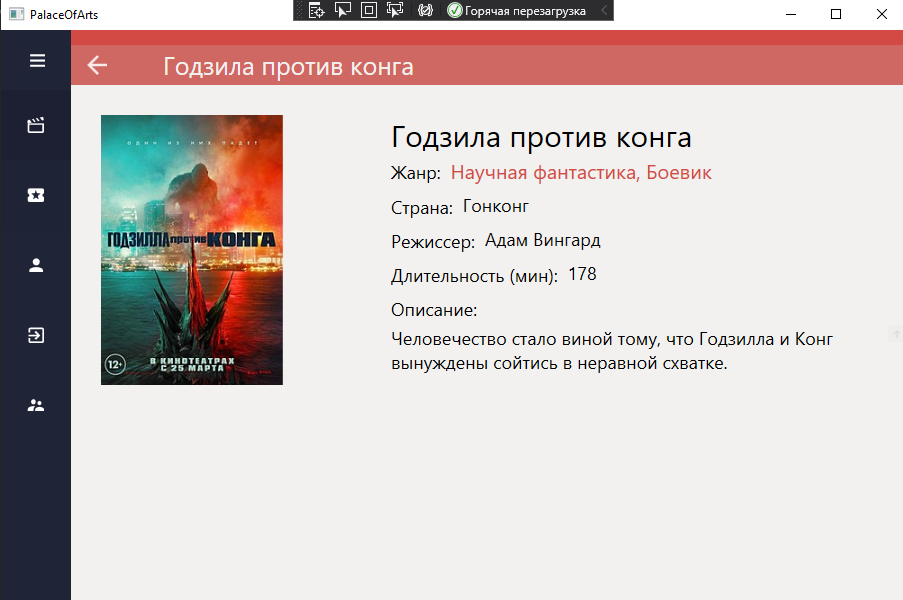


Рисунок 6.10 – Подробная информация о фильме

Выбор места осуществляется при помощи двух комбобоксов внутри которых доступные фильмы и даты. После успешного заказа пользователю на почту прихоидт уведомление. Представлено на рисунке 6.11.

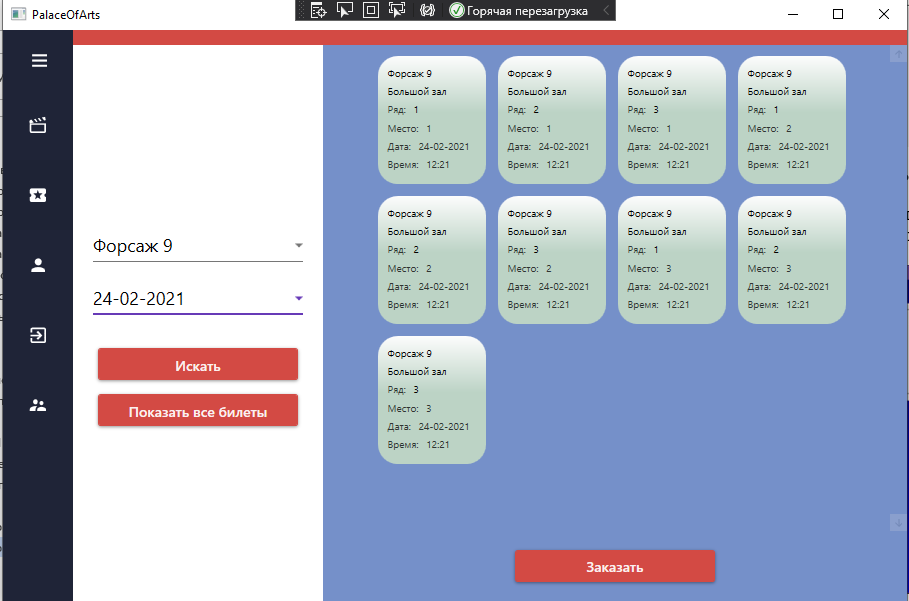


Рисунок 6.12 – Выбор билета

Посмотреть все забронированные рейсы можно на специальной странице, на которую можно попасть из меню. Данная страница показана на рисунке 6.13

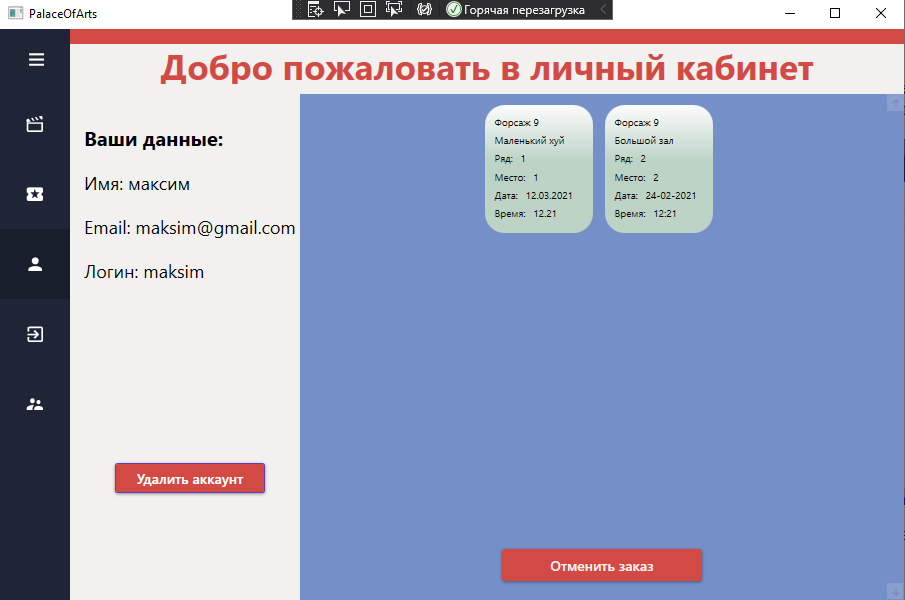


Рисунок 6.13 – Просмотр забронированных пользователем билетов

При нажатии на иконку выхода из аккаунта пользователь может вернуться к окну входа.

ЗАКЛЮЧЕНИЕ

В итоге выполнения данного курсового проекта было разработано программное средство «Бронирование билетов в кинотеатр». При разработке были выполнены все пункты из указанного списка предполагаемого основного функционала приложения.

В программном средстве были реализованы следующие функции:

* регистрация пользователя в системе;
* вход пользователя в систему;
* просмотр данных о фильмах, билетах и забронированных билетах;
* добавление и удаление данных о фильмах, сессиях и билетах;
* поиск необходимого фильма;
* бронирование билетов;
* уведомление пользователей о успешном заказе;
* сортировка фильмов по новизне и по алфавиту.

В соответствии с полученным результатом работы программы можно сделать вывод, что разработанная программа работает, верно, а требования технического задания выполнены в полном объеме.

СПИСОК ЛИТЕРАТУРЫ

1. METANIT.COM Сайт о программировании [Электронный ресурс] / Режим доступа: <https://metanit.com> . Дата доступа: 20.04.2020

2. ProfessorWeb .NET & Web Programming [Электронный ресурс] / Режим доступа: <https://professorweb.ru> Дата доступа: 13.04.2020

3. Форум для программистов или разработчиков [Электронный ресурс] – <https://stackoverflow.com/> – Дата доступа: 5.05.2020

ПРИЛОЖЕНИЕ А

Листинг кода регистрации

public ICommand RegisterCommand { get; }

private bool CanRegisterCommandExecute(object p) => Name?.Length > 0 && registerLogin?.Length > 0

&& registerPassword?.Length > 0;

private void OnRegisterCommandExecuted(object p)

{

string pattern = @"^(?("")(""[^""]+?""@)|(([0-9a-z]((\.(?!\.))|[-!#\$%&'\\*\+/=\?\^`\{\}\|~\w])\*)(?<=[0-9a-z])@))" +

@"(?(\[)(\[(\d{1,3}\.){3}\d{1,3}\])|(([0-9a-z][-\w]\*[0-9a-z]\*\.)+[a-z0-9]{2,17}))$";

if (context.Users.FirstOrDefault(u => u.Login == registerLogin) == null)

{

var window = Application.Current.Windows[0];

if (Regex.IsMatch(Email, pattern, RegexOptions.IgnoreCase))

{

if (context.Users.FirstOrDefault(u => u.Email == Email) == null)

{

if (context.Users.Count() == 0)

{

User user1 = new User(Name, Email, registerLogin, registerPassword);

user1.Role = 1;

context.Users.Add(user1);

context.SaveChanges();

var MainWindowViewModel = new MainWindowViewModel(user1, "AdminPage");

var MainWindow = new MainWindow

{

DataContext = MainWindowViewModel

};

MainWindow.Show();

window.Close();

}

else

{

User user = new User(Name, Email, registerLogin, registerPassword);

context.Users.Add(user);

context.SaveChanges();

var MainWindowViewModel = new MainWindowViewModel(user, "Catalog");

var MainWindow = new MainWindow

{

DataContext = MainWindowViewModel

};

MainWindow.Show();

window.Close();

}

}

else

{

dialogText = "Пользователь с таким email уже зарегистрирован";

dialog = true;

}

}

else

{

dialogText = "Email введен некоректно";

dialog = true;

}

}

else

{

dialogText = "Пользователь с данным псевдонимом уже зарегистрирован.";

dialog = true;

}

}

ПРИЛОЖЕНИЕ Б

Листинг кода входа в систему

public ICommand LoginCommand { get; }

private bool CanLoginCommandExecute(object p) => Login?.Length > 0 && loginPassword?.Length > 0;

AuthWindow window { get => Application.Current.MainWindow as AuthWindow; }

private void OnLoginCommandExecuted(object p)

{

if (context.Users.FirstOrDefault(u => u.Login == Login && u.Password == User.getHash(loginPassword)) != null)

{

User user = context.Users.FirstOrDefault(u => u.Login == Login);

var MainWindowViewModel = new MainWindowViewModel(user, "Catalog");

var MainWindow = new MainWindow

{

DataContext = MainWindowViewModel

};

window.Close();

MainWindow.Show();

}

else

{

dialogText = "Неверный логин или пароль.";

dialog = true;

}

}

ПРИЛОЖЕНИЕ В

Листинг кода отправки сообщения на почту пользователя

private static bool \_CanPingGoogle()

{

const int timeout = 1000;

const string host = "google.com";

var ping = new Ping();

var buffer = new byte[32];

var pingOptions = new PingOptions();

try

{

var reply = ping.Send(host, timeout, buffer, pingOptions);

return (reply != null && reply.Status == IPStatus.Success);

}

catch (Exception)

{

return false;

}

}

public ICommand AddOrderTicketCommand { get; }

private bool CanAddOrderTicketCommandExecute(object p) => true;

private void OnAddOrderTicketCommandExecuted(object p)

{

if (context.OrderTickets.FirstOrDefault(o => o.TicketsId == selectedOutput.TicketsId) == null)

{

context.OrderTickets.Add(new OrderTicket { TicketsId = selectedOutput.TicketsId, UserId = user.UserId });

context.SaveChanges();

//tickets.Remove(selectedOutput);

if (\_CanPingGoogle())

{

try

{

MailAddress from = new MailAddress("unc7447@gmail.com", "Palace of Arts");

MailAddress to = new MailAddress($"{user.Email}");

MailMessage m = new MailMessage(from, to);

m.Subject = "Palace Of Arts";

m.Body = $"<h2>Здравствуйте, {user.Name}, спасибо за то, что воспользовались нашим приложением. <br> Фильм:{selectedOutput.Session.Films.FilmsName}, Зал: {selectedOutput.Session.Halls.HallsName} Дата: {selectedOutput.Session.Date}, Время: {selectedOutput.Session.Time}. Ряд и место: {selectedOutput.Row}; {selectedOutput.Place} </h2>";

m.IsBodyHtml = true;

SmtpClient smtp = new SmtpClient("smtp.gmail.com", 587);

smtp.Credentials = new NetworkCredential("unc7447@gmail.com", "Macbook2019");

smtp.EnableSsl = true;

smtp.Send(m);

Console.Read();

dialogText = $"Спасибо за заказ.\n Письмо отправлено по адресу {user.Email}.";

dialog = true;

}

catch

{

dialogText = "Спасибо за заказ\nСообщение не было отправлено";

dialog = true;

}

}

else

{

dialogText = "Спасибо за заказ, наш менеджер свяжется с вами";

dialog = true;

}

tickets.Remove(selectedOutput);

}

else

{

dialogText = "Этот билет уже заказан";

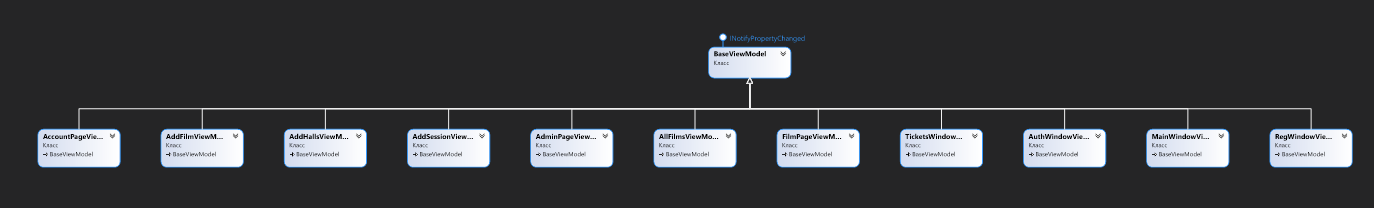
dialog = true;

}

}

ПРИЛОЖЕНИЕ Г

Диаграмма UML классов модуля ViewModel



ПРИЛОЖЕНИЕ Д

