СОДЕРЖАНИЕ

[Введение 3](#_Toc102465893)

[1 Экономическая характеристика объекта 4](#_Toc102465894)

[2 Постановка задачи 6](#_Toc102465895)

[2.1 Экономическое содержание задачи 7](#_Toc102465896)

[2.2 Входная информация 9](#_Toc102465897)

[2.3 Выходная информация 10](#_Toc102465898)

[3 Технические и инструментальные средства для решения задачи 11](#_Toc102465899)

[3.1 Обоснование выбора языка программирования 11](#_Toc102465900)

[3.2 Технические требования к компьютеру для решения задачи 12](#_Toc102465901)

[4 Информационное обеспечение задачи 16](#_Toc102465902)

[4.1 Классификаторы, используемые для решения задачи 16](#_Toc102465903)

[5 Программное обеспечение задачи 20](#_Toc102465904)

[5.1 Описание методов 20](#_Toc102465905)

[5.2 Алгоритм решения задачи 22](#_Toc102465906)

[5.3 Описание блок – схемы 22](#_Toc102465907)

[Заключение 25](#_Toc102465908)

[Список использованных источников 26](#_Toc102465909)

[Приложение А 22](#_Toc102465910)

[Приложение Б 23](#_Toc102465911)

# Введение

В данной курсовой работе рассматривается проблема отсутствия свободных мест на нужный сеанс во время прихода в кинотеатр. Следствием чего является впустую потраченное время и испорченное настроение. Целью данной курсовой работы является решение данной проблемы путем реализации программного продукта на одном из языков программирования. Задача проектирования состоит в том, чтобы максимально просто добиться результата поставленной задачи. Было предложено реализовать программный продукт, который позволял бы пользователям покупать билеты онлайн – “Система покупки билетов онлайн”.

Для решения данной проблемы были выдвинуты следующие цели:

* пользователи (посетители кинотеатра) могли приобретать билеты онлайн, не выходя из дому;
* пользователи могли просматривать расписание сеансов на неделю вперёд, в том числе занятость зала на интересующий их сеанс;
* администратор (работник кинотеатра) мог бы добавлять или удалять фильмы, сеансы; изменять количество мест в зале, цены билетам; а также продавать билеты на сеансы.

# 1 Экономическая характеристика объекта

Состав и взаимосвязь помещений

Кинотеатр представляет собой учреждение, занимающееся организацией кинопоказов и проведением иных развлекательных мероприятий, связанных с кино. Возведение отдельно стоящего однозального кинотеатра стоит примерно 2-3 млн долларов. Эта сумма вполне приемлема при наличии инвестиционных источников и учитывая перспективы будущего бизнеса. Она быстро окупится, и через два года проект начнет приносить значительную прибыль.

Все помещения кинотеатра подразделяют на следующие комплексы и группы: помещения зрительского комплекса;. помещения демонстрационного комплекса — зрительный зал, помещения технологического обеспечения кинопоказа; административно-хозяйственные и производственные помещения; технические помещения.

Каждая из этих групп помещений получает развитие, соответствующее характеру и вместимости здания. Расчет площадей помещений приводится в нормах проектирования культурно-зрелищных учреждений ВСН 45-86. Взаиморасположение групп должно создавать простой и удобный график движения зрителей, оптимальные условия технологического процесса демонстрации фильмов и пожарную безопасность. Помещения зрительского комплекса  
  
 Неспециализированных кинотеатров включают: кассовый вестибюль с помещениями касс, входной вестибюль, фойе или распределительные кулуары, буфет с подсобными помещениями, курительную и санитарные узлы. В неспециализированных кинотеатрах для строительства в 1A, 1Б, 1Г климатических подрайонах допускается проектирование выходного вестибюля, гардероба, комнаты для переодевания и независимо от климатических подрайонов — комнаты игровых автоматов и детской комнаты.  
  
 Проектирование специализированных кинотеатров ведется по индивидуальным заданиям, в которых определяется состав помещении, отвечающий назначению кинотеатра и конкретным условиям места строительства.  
  
 Премьерный кинотеатр — это, как правило, крупнейший кинотеатр города с большим залом универсального назначения, возможно, и с малыми специализированными залами. В кинотеатрах этого типа помимо основного состава помещений зрительского комплекса рекомендуется предусматривать гостиную, аудиторию (просмотровый зал), выставочный зал.  
Досуговый кинотеатр проектируют с расчетом на посетителей, желающих не только посмотреть кинофильм, но и отдохнуть, потанцевать. Такой кинотеатр может иметь гардероб при вестибюле, развитые фойе и буфет или кооперироваться с кафе и танцзалом, иметь гостиную, бар, зимний сад, зал игровых автоматов. Этот тип кинотеатра может быть двухзальным, причем больший зал рекомендуется делать универсальным, а при эстраде проектировать артистические.  
  
  
 Из входного вестибюля, пройдя пост контроля билетов, зрители попадают в фойе или распределительные кулуары — помещения, в которых зрители накапливаются в ожидании начала сеанса. При фойе в стороне от входов в зрительный зал располагают курительные и санитарные узлы либо на уровне фойе, либо уровнем ниже.  
Помещения фойе и кулуаров проектируют в зависимости от типа кинотеатра. В неспециализированных кинотеатрах площадь фойе определяют из расчета 0,4 м2 на место. В кинотеатрах с универсальными залами, где на площади фойе могут быть расположены эстрада и места для зрителей, площадь принимают 0,45 м2 на место, в детских кинотеатрах — 0,6 м2, кинотеатрах с непрерывным кинопоказом — 0,25 м2 на место (включая буфет).  
  
 Помещения демонстрационного комплекса. В кинотеатрах демонстрационный комплекс включает зрительный зал и помещения технологического обеспечения кинопоказа, в кинотеатрах с универсальными залами, кроме того, эстраду и помещения ее технологического обеспечения.  
  
 Зрительный зал является основным помещением кинотеатра. Планировка и оборудование зрительного зала должны создавать условия для хорошей видимости со всех мест, нормальной акустики, удобного распределения зрителей по местам и быстрой их эвакуации. Этими основными задачами и определяются соотношения размеров зала, его форма и объем, размеры и расположение экрана, размещение мест и проходов между ними.  
  
 Размеры, форма и объем зала зависят в первую очередь от его вместимости. Площадь зала принимают из расчета 1,0 м2 на одного зрителя в кинотеатрах круглогодичного действия и 0,9 м2 — в кинотеатрах сезонного действия (площадь определяют в пределах ограждающих конструкций, включая эстраду).  
  
 Основой формообразования зала являются условия кинопроекции и требование расположения зрительских мест в зоне оптимальной видимости. Именно это определяет габариты зала (длину, ширину, высоту) и профиль его пола, размеры и размещение экрана, характер расположения мест.

2. Постановка задачи

Предметная область, подлежащая изучению –«кинотеатр». В сферу этой предметной области попадают предприятия, занимающиеся оказанием услуг проведения досуга. Цель функционирования предприятий этой предметной области – оказание услуг, таких как: показ кинофильмов, реклама и анонс будущих фильмов и получение прибыли, извлекаемой из оплаты этих услуг клиентами. Для оказания услуг необходимо наличие квалифицированных специалистов и помещений, удовлетворяющих нормам санитарных и других требований в соответствии с действующим законодательством.

Целью курсовой работы являетсясоздание информационно-программной системы для решения следующих задач:

1. автоматизации операций осуществления продажи билетов.
2. централизованного хранения информации о фильмах с целью:
   1. их вторичного показа,
   2. продажи прокатного удостоверения.
3. обеспечения хранения информации о количестве посетителей заведения с целью:
   1. сбора информации о посещаемости фильма,
   2. подсчета вырученных средств от продажи билетов,
   3. возможности увеличения или уменьшения количества сеансов.

К разрабатываемой системе предъявлены следующие требования.

1. наличие централизованного хранилища данных для всех объектов автоматизированного учета,
2. ведение операций информационного учета с компьютеров локальной сети предприятия,
3. создание унифицированного доступа к данным с использованием специализированного программного интерфейса пользователя, который бы отражал специфику этой предметной области.

Для решения поставленных задач планируется выполнение следующих действий:

1. проектирование данных для отражения специфики данной предметной области,
2. разработка и создание базы данных для хранения информации на едином сервере данных под управлением MS SQL Server ,
3. разработка компьютерной программы для выполнения базовых операций над данными предметной области, которая выполняет следующие функции:
   1. добавление информации об основных объектах (фильмах и мультфильмах) редактирование информации по основным объектам,
   2. поиск данных по основным объектам учёта:
      1. фильмов;
      2. посетителей;
      3. сеансов;
   3. получение итоговой информации по отчетным характеристикам системы:
      1. получение суммарной выручки от продажи всех билетов,
      2. наиболее посещаемые фильмы.

## 2.1 Экономическое содержание задачи

Деятельность кинотеатра направлена на удовлетворение развлекательной и культурно-просветительной потребности населения. На основе миссии определяются цели деятельности. Следовательно, целью деятельности данного кинотеатра является максимально большее привлечение кинозрителей к просмотру максимально большего количества фильмов разнообразных жанров и тематик, для развлечения кинозрителей и оказания ни них культурно-научно-просветительского воздействия.

Рассмотрим достоинства и недостатки существующей организационной структуры компании. Управление компанией носит линейный характер, основанный на единоначалии. К основным недостаткам такого метода управления можно отнести следующие:

1. Высокие требования к руководителю, который должен иметь багаж всесторонних знаний, необходимых для управления компанией. Остановимся на этой проблеме управления компанией более подробно. Управление компанией на уровне генерального директора предполагает решение вопросов по позиционированию на рынке реализуемого продукта или услуги, уменьшение затрат на закупку и реализацию, подбор нужных кадров, т.е. объем знаний и умений, в котором сложно быть компетентным одному человеку.
2. Линейная структура лишена необходимых промежуточных звеньев по планированию и подготовке решений, что приводит к отсутствию видения ближайшей перспективы развития компании и негативно отражается на ее будущей конкурентоспособности.
3. Средние уровни управления компанией перегружены лишней организационной информацией, что мешает им решать на должном уровне конкретные необходимые для развития компании задачи.
4. Излишняя перегруженность средних уровней ослабляет связи между ними, что влияет на качество принимаемых на этих уровнях управленческих решений.
5. Так как общее управление компанией организовано по линейному принципу, при отсутствии одного из ключевых руководителей нарушается работа как одного из звеньев компании, так и всей компании целиком. Это способствует дополнительным рискам в работе компании и делает ее менее конкурентоспособной. Ещё одним недостатком в управлении компанией является программная автономия ее отдельных подразделений, занимающихся реализацией билетов.

Для устранения описанных недостатков управления компанией предлагается сделать ее более интегрированной и постепенно перевести механизм управления с линейного на функциональный, используя для этого средства дополнительной автоматизации процесса продажи билетов.

## 

## 2.2 Входная информация

Входной является информация, которую пользователь вносит в файл базы данных, заполняя необходимые поля выбранной таблицы, а также вводя информацию в базу данных с помощью запросов SQL.

Для разрабатываемого программного продукта входной будет служить следующая информация:

- информация о сеансах (время проведения сеанса, стоимость билета на данный сеанс);

- информация о купленных билетах (дата реализации билета, место и ряд в зрительном зале);

- информация об имеющихся фильмах (название фильма, жанр, возрастные ограничения на просмотр данного фильма).

В ИС используется несколько справочников, которые формируют нормативно-справочную, входную и оперативную информацию.

В информационной системе используются следующие справочники:

* Фильмы;
* Сеансы;
* Билеты.

В системе за счет использования запросов сформированы документы:

* Список сеансов;
* Список ID билетов;
* Статистика.

2.3 Выходная информация

Выходная информация - результат выполнения запросов, фильтрации данных, вывод необходимой информации в отчет, печать информации. Информация, которая несет вывод либо обобщает указанные данные в общем виде или по определенному критерию.

Выходной информацией для данного проекта является информация, которая позволяет произвести вывод на печать отчетной формы: список реализованных билетов. Вывод информации о выручке кинотеатра за определенный период.

Количество экранных форм и их содержание спроектировано так, чтобы обеспечить возможность вывода всей результатной информации, а также результатов запросов пользователя к базе данных. Предусмотрены выдача результатов как на экран ПЭВМ, так и на принтер по желанию пользователя. Вся результатная информация проекта автоматизации представляется в виде отчетов и экранных форм. Ниже рассмотрены детали каждого из результатных документов.

Выходными документами разрабатываемой программы являются:

* Отчет по продажам;
* Отчет по сеансам;
* Отчет по возвратам.

# 3 Технические и инструментальные средства для решения задачи

## 3.1 Обоснование выбора языка программирования

Данное приложение сделано с помощью языка C# - обьектно ориентированный язык программирования со статической типизацией, разработанный компанией Microsoft для платформы Microsoft .NET Framework и .NET Core. Программы на C# выполняются в .NET, виртуальной системе выполнения, вызывающей общеязыковую среду выполнения (CLR) и набор библиотек классов. Среда CLR – это реализация общеязыковой инфраструктуры языка (CLI), являющейся международным стандартом, от корпорации Microsoft.

В качестве среды для разработки приложения была выбрана программа Microsoft Visual Studio. Она является одним из линейки продуктов компании Microsoft, включающих интегрированную среду разработки программного обеспечения и не только. Данная среда разработки была выбрана в ввиду ее тесного взаимодействия с языком C#, а также ее удобным графическим интерфейсом и средств отладки. Основными особенностями является Visual Studio является вcтроенный инструмент редактора форм для упрощения создания графического интерфейса, отладчик уровня исходного кода и машинного уровня. Также включается в себя богатый по возможностям редактор исходного кода с поддержкой технологии IntelliSense. Вдобавок ко всему, в среде разработки встроена поддержка системы контроля версий Git, что упрощает сопровождение системы. Еще одна полезная возможность в Microsoft Visual Studio – это возможность использовать множество плагинов для расширения функциональных возможностей системы практически на каждом уровне работы. Это позволяет настраивать рабочую среду под конкретного человека, что улучшает взаимодействие конкретного пользователя со средой разработки. Помимо всего, следует отметить поддержку большого количества платформ, из которых наиболее популярны: Microsoft Windows, .NET Framework, .NET Core, Xbox, Microsoft Silverlight, и др.

Главные возможности среды разработки Microsoft Visual Studio:

* Кроссплатформенность среды разработки
* Различные редакции для различного пользования
* Поддержка популярных языков программирования
* Быстрая сборка и компиляция проекта
* Наличие менеджера пакетов NuGet
* Наличие нескольких средств создания приложений с графическим интерфейсом
* Поддержка создания и использования тестов для проектов
* Мониторинг используемой памяти
* Поддержка совместной работы над кодом в проектах
* Наличие бесплатной версии среды разработки
* Наличие поддержки платформ .NET Core и .NET Framework

- Простая установка среды разработки на любой поддерживаемой платформе.

- Наличие средств для простой разработки и развертывания баз данных SQL Server

## 

## 3.2 Технические требования к компьютеру для решения задачи

Рекомендуемые системные требования к компьютеру на которой будет установлена IDE Visual Studio 2022 следующие:

### Операционные системы:

* Windows 11 версии 21H2 или выше: домашняя, Pro, Pro образования, Pro для рабочих станций, Enterprise и образовательных учреждений.
* Windows 10 версии 1909 или более поздних версий: Домашняя, Профессиональная, для образовательных учреждений и Корпоративная.
* Windows Server 2022: Standard и Datacenter.
* Windows Server 2019: Standard и Datacenter.
* Windows Server 2016: Standard и Datacenter
* Windows Server 2022: Standard и Datacenter.
* Windows Server 2019: Standard и Datacenter.
* Windows Server 2016: Standard и Datacenter.

### Оборудование :

64-разрядный процессор 1,8 ГГц или более мощный. Рекомендуется четырехъядерный или с большим количеством ядер. Процессоры ARM не поддерживаются.

* + - Не менее 4 ГБ ОЗУ. На используемые ресурсы влияет множество факторов; для стандартных профессиональных решений рекомендуется 16 ГБ ОЗУ.
    - Windows 365: не менее 2 виртуальных ЦП и 8 ГБ ОЗУ. Рекомендуется 4 виртуальных ЦП и 16 ГБ ОЗУ.
    - Место на жестком диске: от 850 ГБ до 210 ГБ свободного места в зависимости от установленных компонентов, обычно для установки требуется от 20 до 50 ГБ свободного места. Для повышения производительности рекомендуется устанавливать Windows и Visual Studio на твердотельном накопителе (SSD).
    - Видеоадаптер с минимальным разрешением WXGA (1366 на 768 пикселей); для оптимальной работы Visual Studio рекомендуется разрешение 1920 на 1080 пикселей или выше.
    - Минимальное разрешение предполагает масштабирование, параметры DPI и масштабирование текста на уровне 100%. Если не задано значение 100%, необходимо соответствующим образом масштабировать минимальное разрешение. Например, если задать для дисплея Windows значение параметра "Масштаб и макет" на устройстве Surface Book, которое имеет физический дисплей 3000x2000, значение 200%, то для Visual Studio логическое разрешение экрана будет 1500x1000, что соответствует минимальным требованиям в 1366x768.

### Поддерживаемые языки:

Visual Studio доступна на следующих языках: английский, китайский (упрощенное и традиционное письмо), чешский, французский, немецкий, итальянский, японский, корейский, польский, португальский (Бразилия), русский, испанский и турецкий. Язык Visual Studio можно выбрать во время установки. Установщик Visual Studio доступен на тех же четырнадцати языках и будет соответствовать языку Windows (если он доступен).

### Дополнительные требования и инструкции:

Для установки или обновления Visual Studio требуются права администратора.

* Дополнительные рекомендации и инструкции по установке, развертыванию, обновлению и настройке Visual Studio в организации см. в руководстве администратора Visual Studio.
* Для установки Visual Studio необходима платформа .NET Framework 4.5.2 или выше. Для работы Visual Studio требуется .NET Framework 4.8. Если платформа .NET Framework 4.8 еще не установлена, то будет выполнена ее установка.
* Для разработки универсальных приложений для Windows, включая проектирование, редактирование и отладку, требуется Windows 10. Windows Server 2019 и Windows Server 2016 могут использоваться для создания универсальных приложений для Windows из командной строки.
* Для установки Visual Studio необходима среда выполнения WebView2. Если она еще не установлена, то будет выполнена ее установка. Установка Visual Studio завершится сбоем, если не удастся успешно установить WebView2 (из-за ошибки установки или ограничений групповой политики на установку или обновление).
* Для интеграции Team Foundation Server 2019 с Office требуется Office 2016, Office 2013 или Office 2010.

## Требования к системе для инструментов удаленной отладки и изолированного сборщика IntelliTrace для Visual Studio 2022

Для инструментов удаленной отладки и изолированного сборщика IntelliTrace предъявляются те же требования к системе, что и для Visual Studio, со следующими отличиями:

* Также поддерживаются в Windows 7, Windows Server 2008 с пакетом обновления 2 (SP2), Windows Server 2008 R2 с пакетом обновления 1 (SP1), Windows 10 в ARM, Windows 10 Корпоративная LTSC, Windows Server 2012 R2, Windows 8.1 и Windows Server 2012.
* При использовании архитектуры x86 или AMD64/x64 требуется процессор с тактовой частотой не ниже 1,6 ГГц
* Требуется 1 ГБ ОЗУ (1,5 ГБ при запуске на виртуальной машине)
* Требуется 1 ГБ доступного пространства на жестком диске
* Требуется разрешение экрана не ниже 1024 на 768
* Для получения наилучших результатов используйте средства диагностики с самым последним обновлением для вашей версии Visual Studio.

# 4. Информационное обеспечение задачи

## 4.1 Классификаторы, используемые для решения задачи

Классификатор – это сгруппированный, по различным признакам, перечень  наименованных объектов, где каждому из них присвоен уникальный код, в соответствии с их общими признаками или различиями. Классификаторы применяются для упрощения работы по сбору информации и систематизации данных, а также используются для указания кодов в различных унифицированных документах.

Объект – некоторая сущность в цифровом пространстве, обладающая определённым состоянием и поведением в данной предметной области.

Атрибут – элемент, описывающий любую характеристику объекта, важную для данной предметной области. Объектом в данной базе данных будет: информация о фильмах приведенная в таблице 4.1, залы на таблице 4.2 жанры на таблице 4.3, сеансы на таблице 4.4 и информация о выданных билетах в таблице 4.5

Таблица 4.1 – Атрибуты фильма

|  |  |
| --- | --- |
| Объект | Атрибуты |
| Фильмы | Идентификационный код  Фильм  Жанр  Дата сеанса  Год выхода  Зал |

Таблица 4.2 – Атрибуты зала

|  |  |
| --- | --- |
| Объект | Атрибуты |
| Залы | Идентификационный код  Зал |

Таблица 4.3 – Атрибуты жанров

|  |  |
| --- | --- |
| Объект | Атрибуты |
| жанры | Идентификационный код  жанры |

Таблица 4.4 – Атрибуты сеансов

|  |  |
| --- | --- |
| Объект | Атрибуты |
| Сеансы | Идентификационный код  Фильм  Цена  Дата сеанса  Зал | |

Таблица 4.5 – Атрибуты билетов

|  |  |
| --- | --- |
| Объект | Атрибуты |
| Билеты | Идентификационный код  Фильм  Дата сеанса  Зал  Ряд  Место |

Для выполнения проекта была выбрана СУБД SQL Server. Схема БД представлена на рисунке 4.6.

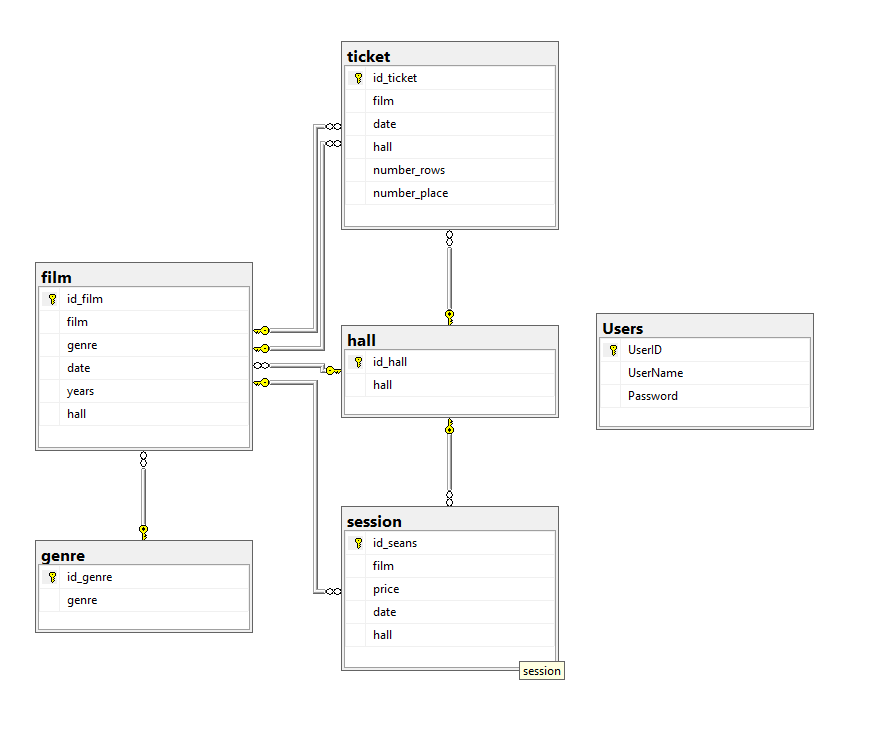


Рисунок 4.6 – Связи таблиц в БД

Структура таблиц базы данных представлена на рисунках 4.7 – 4.11 соответственно.

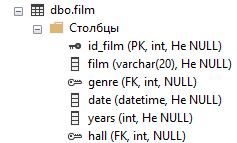


Рисунок 4.7 – Таблица «Фильмы»

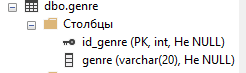


Рисунок 4.8 – Таблица «Жанры»

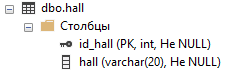


Рисунок 4.9 – Таблица «Залы»

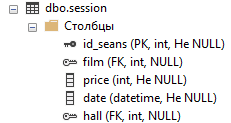


Рисунок 4.10 – Таблица «Сеансы»

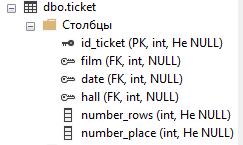


Рисунок 4.11 – Таблица «Билеты»

# 5 Программное обеспечение задачи

## 5.1 Описание методов

Для примера взята часть кода для входа в приложение под своей учетной записью. При нажатии на кнопку вызывается активити, для дальнейшей работы с приложением.

|  |
| --- |
| using System; |
| using System.Collections.Generic; |
|  |
| using System.Text; |
| using System.Threading.Tasks; |
|  |
| using System.Windows.Controls; |
| using System.Windows.Data; |
| using System.Windows.Documents; |
| using System.Windows.Media.Imaging; |
| using System.Windows.Navigation; |
| using System.Windows.Shapes; |
| using System.Data; |
| using System.Data.SqlClient; |
|  |
| namespace Cinema |
| { |
|  |
| public partial class MainWindow :Window |
| { |
| public MainWindow() |
| { |
| InitializeComponent(); |
| } |
|  |
| private void btnSubmit\_Click(object sender, RoutedEventArgs e) |
| { |
| SqlConnection sqlConnection = new SqlConnection("Data Source=MSI-AEGIS-TI3;Initial Catalog=Cinema2.0;Integrated Security=True"); |
| try |
| { |
| if (sqlConnection.State == ConnectionState.Closed) |
| sqlConnection.Open(); |
| String query = "SELECT COUNT(1) FROM Users WHERE UserName = @UserName AND Password = @Password"; |
| SqlCommand sqlCmd = new SqlCommand(query, sqlConnection); |
| sqlCmd.CommandType = CommandType.Text; |
| sqlCmd.Parameters.AddWithValue("@UserName", txtUsername.Text); |
| sqlCmd.Parameters.AddWithValue("@Password", txtPassword.Password); |
| int count = Convert.ToInt32(sqlCmd.ExecuteScalar()); |
| if (count == 1) |
| { |
| List list = new List(); |
| list.Show(); |
| this.Close(); |
| } |
| else |
| { |
| MessageBox.Show("Логин или пароль неверны !"); |
| } |
| } |
| catch (Exception ex) |
| { |
| MessageBox.Show(ex.Message); |
| } |
| finally |
| { |
| sqlConnection.Close(); |
| } |
| } |
|  |
| private void StackPanel\_MouseDown(object sender, MouseButtonEventArgs e) |
| { |
| if (e.ChangedButton == MouseButton.Left) |
| { |
| this.DragMove(); |
| } |
| } |
| } |
| } |

## 5.2 Алгоритм решения задачи

Для проведения каких-либи действий с кинотеатром сначало необходмо войти в учётную запись, а затем выбрать кнопку «купить билет»

## 5.3 Описание блок – схемы

Последовательное описание блок – схемы приложения приведено ниже:

Шаг 1. Запуск программы.

Шаг 2. Вход в приложение под своей учетной записью.

Шаг 3. Вопрос: Выбрано ли действие? Если да, то переходим к шагу 4, если нет, то переходим к 2 шагу.

Шаг 4. Выполнение

Шаг 5. Возвращаемся к 3 шагу.

Шаг 6 Закрытие приложения.

Блок-схема приведена в приложении А.

Листинг программы приведен в приложении Б.

5.4 Описание интерфейса программы

В окне входа в приложение, представленном на рисунке 5.1, находятся два текстовых поля логин и пароль, и кнопка входа в приложение. Для дальнейшей работы нужно верно указать логин и пароль. Если логин и пароль верны, то откроется основное окно, представленное на рисунке 5.2 с возможностью выбора действий. При нажатии на кнопки таблиц откроется окно с таблицами БД, показанное на рисунке 5.3.

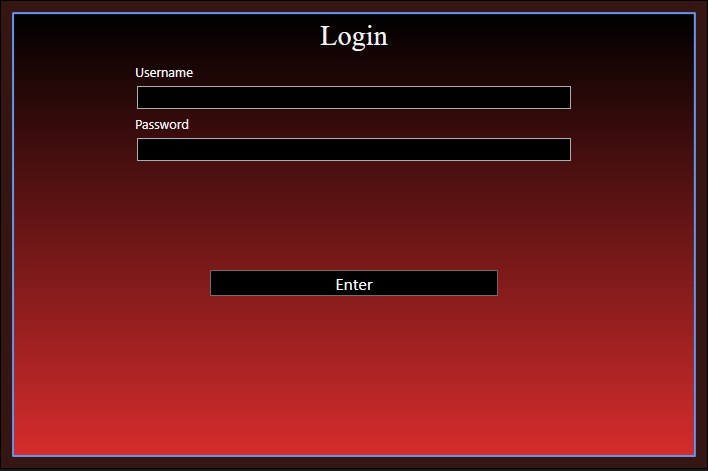


Рисунок 5.1 – Активити входа



Рисунок 5.2 – Выбор действий

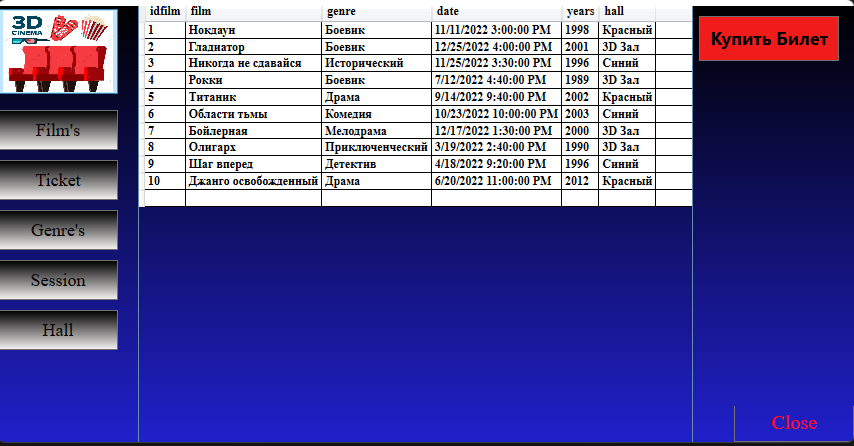


Рисунок 5.3 – Таблицы БД

Для успешной работы с программой необходимо:

1. Подключение к интернету.
2. Учетная запись.

# Заключение

В процессе работы над программным продуктом автором были выявлены и решены проблемы доступа из основного интерфейса к дополнительным меню и формам программы путем использования оригинального дизайна интерфейса и его эргономичного решения.

Также была разработана и реализована необходимая система доступа и перехода к основным и дополнительным блокам программного продукта, обеспечивающего точное описание необходимых в работе подразделения ИТ серверных и сетевых механизмов доступа каждого рабочего места сотрудника.

Автором в процессе работы над программным продуктом был произведен необходимый научный и технический анализ для проектирования и реализации программного продукта «Автоматизация продажи билетов».

Цель курсового проекта достигнута. Задачи выполнены.

# Список использованных источников

Интернет ресурсы:

1. Документация по Entity Framework. [Электронный ресурс] – Режим доступа: https:docs.microsoft.com/ru-ru/ef/
2. Википедия. Язык программирования C# [Электронный ресурс] – Режим доступа: https:ru.wikipedia.org/wiki/C\_Sharp
3. Шаблон Model-View-ViewModel. [Электронный ресурс] – https:docs.microsoft.com/ru-ru/xamarin/xamarin-forms/enterprise-application-patterns/mvvm
4. SQL Server 2019 [Электронный ресурс] – https:www.microsoft.com/ru-ru/sql-server/sql-server-2019

# Приложение А

(обязательное)

Блок – схема программы

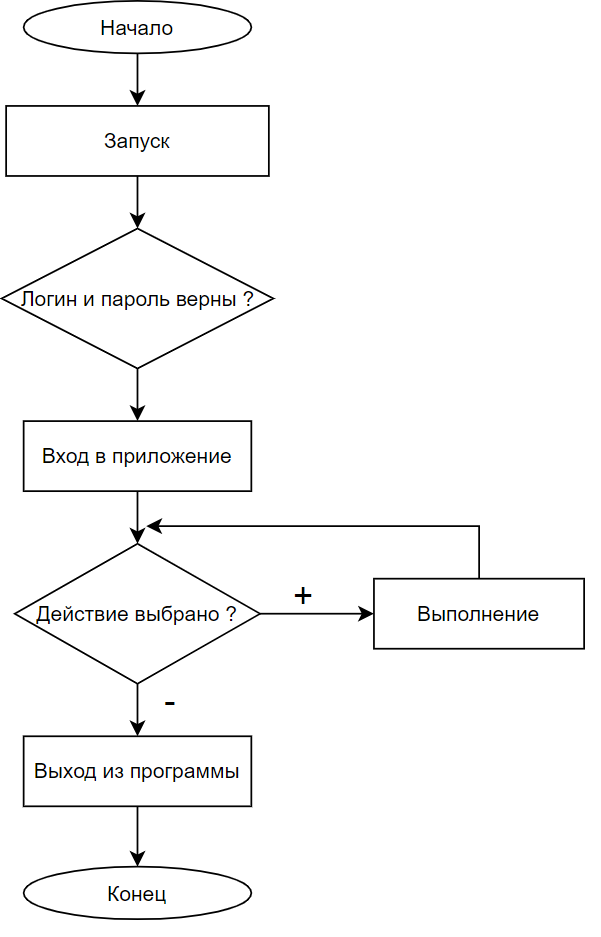


Рисунок А.1–Блок схема программы

# Приложение Б

(обязательное)

Листинг программы

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows;

using System.Windows.Controls;

using System.Windows.Data;

using System.Windows.Documents;

using System.Windows.Input;

using System.Windows.Media;

using System.Windows.Media.Imaging;

using System.Windows.Navigation;

using System.Windows.Shapes;

using System.Data;

using System.Data.SqlClient;

namespace Cinema

{

public partial class MainWindow : Window

{

public MainWindow()

{

InitializeComponent();

}

private void btnSubmit\_Click(object sender, RoutedEventArgs e)

{

SqlConnection sqlConnection = new SqlConnection("Data Source=MSI-AEGIS-TI3;Initial Catalog=Cinema2.0;Integrated Security=True");

try

{

if (sqlConnection.State == ConnectionState.Closed)

sqlConnection.Open();

String query = "SELECT COUNT(1) FROM Users WHERE UserName = @UserName AND Password = @Password";

SqlCommand sqlCmd = new SqlCommand(query, sqlConnection);

sqlCmd.CommandType = CommandType.Text;

sqlCmd.Parameters.AddWithValue("@UserName", txtUsername.Text);

sqlCmd.Parameters.AddWithValue("@Password", txtPassword.Password);

int count = Convert.ToInt32(sqlCmd.ExecuteScalar());

if (count == 1)

{

List list = new List();

list.Show();

this.Close();

}

else

{

MessageBox.Show("Логин или пароль неверны !");

}

}

catch (Exception ex)

{

MessageBox.Show(ex.Message);

}

finally

{

sqlConnection.Close();

}

}

private void StackPanel\_MouseDown(object sender, MouseButtonEventArgs e)

{

if (e.ChangedButton == MouseButton.Left)

{

this.DragMove();

}

}

}

}

namespace Cinema

{

/ <summary>

/ Логика взаимодействия для List.xaml

/ </summary>

public partial class List : Window

{

public List()

{

InitializeComponent();

}

SqlConnection sqlConnection = new SqlConnection("Data Source=MSI-AEGIS-TI3;Initial Catalog=Cinema2.0;Integrated Security=True");

private void Button\_Click(object sender, RoutedEventArgs e)

{

sqlConnection.Open();

string cmd = "SELECT film.id\_film,film.film,genre.genre,film.[date],film.years,hall.hall FROM film left join genre on genre.id\_genre = film.genre left join hall on hall.id\_hall = film.hall";

SqlCommand createCommand = new SqlCommand(cmd, sqlConnection);

createCommand.ExecuteNonQuery();

SqlDataAdapter dataAdp = new SqlDataAdapter(createCommand);

DataTable table = new DataTable("film");

dataAdp.Fill(table);

ListGrid.ItemsSource = table.DefaultView;

sqlConnection.Close();

}

private void Button\_Click\_1(object sender, RoutedEventArgs e)

{

sqlConnection.Open();

string cmd = "SELECT \* FROM hall";

SqlCommand createCommand = new SqlCommand(cmd, sqlConnection);

createCommand.ExecuteNonQuery();

SqlDataAdapter dataAdp = new SqlDataAdapter(createCommand);

DataTable table = new DataTable("hall");

dataAdp.Fill(table);

ListGrid.ItemsSource = table.DefaultView;

sqlConnection.Close();

}

private void Button\_Click\_2(object sender, RoutedEventArgs e)

{

sqlConnection.Open();

string cmd = "SELECT \* FROM genre";

SqlCommand createCommand = new SqlCommand(cmd, sqlConnection);

createCommand.ExecuteNonQuery();

SqlDataAdapter dataAdp = new SqlDataAdapter(createCommand);

DataTable table = new DataTable("genre");

dataAdp.Fill(table);

ListGrid.ItemsSource = table.DefaultView;

sqlConnection.Close();

}

private void Button\_Click\_3(object sender, RoutedEventArgs e)

{

sqlConnection.Open();

string cmd = "SELECT [session].id\_seans,film.film,[session].price,film.[date],hall.hall FROM [session] left join film on film.id\_film = [session].film left join hall on hall.id\_hall = [session].hall";

SqlCommand createCommand = new SqlCommand(cmd, sqlConnection);

createCommand.ExecuteNonQuery();

SqlDataAdapter dataAdp = new SqlDataAdapter(createCommand);

DataTable table = new DataTable("session");

dataAdp.Fill(table);

ListGrid.ItemsSource = table.DefaultView;

sqlConnection.Close();

}

private void Button\_Click\_4(object sender, RoutedEventArgs e)

{

sqlConnection.Open();

string cmd = "SELECT ticket.id\_ticket, film.film,film.date,hall.hall,ticket.number\_rows,ticket.number\_place FROM ticket left join film on film.id\_film = ticket.film left join hall on hall.id\_hall = ticket.hall";

SqlCommand createCommand = new SqlCommand(cmd, sqlConnection);

createCommand.ExecuteNonQuery();

SqlDataAdapter dataAdp = new SqlDataAdapter(createCommand);

DataTable table = new DataTable("ticket");

dataAdp.Fill(table);

ListGrid.ItemsSource = table.DefaultView;

sqlConnection.Close();

}

private void Button\_Click\_7(object sender, RoutedEventArgs e)

{

this.Close();

}

private void Grid\_MouseDown(object sender, MouseButtonEventArgs e)

{

if (e.ChangedButton == MouseButton.Left)

{

this.DragMove();

}

}

private void Button\_Click\_8(object sender, RoutedEventArgs e)

{

SaleTicket saleTicket = new SaleTicket();

saleTicket.Show();

this.Close();

}

}

}

using System;

using System.Collections.Generic;

using System.ComponentModel.DataAnnotations;

using System.ComponentModel.DataAnnotations.Schema;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Cinema.Model

{

public class Film

{

[Key]

public int id { get; set; }

[Required]

public int? Genreid { get; set; }

[ForeignKey("Genreid")]

public Genre Genre { get; set; }

[Required]

public int? hallid { get; set; }

[ForeignKey("hallid")]

public hall hall { get; set; }

[Required]

public int? sessionid { get; set; }

[ForeignKey("sessionid")]

public session session { get; set; }

[Required]

public DateTime Date { get; set; }

[Required]

public DateTime Years { get; set; }

[NotMapped]

public Genre genre

{

get

{

return DataWorker.GetGenreById((int)Genreid);

}

}

[NotMapped]

public hall Hall

{

get

{

return DataWorker.GetHallById((int)hallid);

}

}

[NotMapped]

public session Session

{

get

{

return DataWorker.GetSessionById((int)sessionid);

}

}

}

}

public class Genre

{

[Key]

public int id { get; set; }

[Required]

public string genre { get; set; }

}

public class hall

{

[Key]

public int id { get; set; }

[Required]

public string Hall { get; set; }

}

public class session

{

[Key]

public int id { get; set; }

[Required]

public int? filmid { get; set; }

[ForeignKey("filmid")]

public Film film { get; set; }

[Required]

public int? hallid { get; set; }

[ForeignKey("hallid")]

public hall hall { get; set; }

[Required]

public int? price { get; set; }

[Required]

public DateTime Date { get; set; }

[NotMapped]

public hall Hall

{

get

{

return DataWorker.GetHallById((int)hallid);

}

}

[NotMapped]

public Film Film

{

get

{

return DataWorker.GetFilmById((int)filmid);

}

}

}

public class Ticket

{

[Key]

public int id { get; set; }

[Required]

public int? filmid { get; set; }

[ForeignKey("filmid")]

public Film film { get; set; }

[Required]

public int? hallid { get; set; }

[ForeignKey("hallid")]

public hall hall { get; set; }

[Required]

public DateTime Date { get; set; }

[Required]

public string number\_rows { get; set; }

[Required]

public string number\_place { get; set; }

[NotMapped]

public hall Hall

{

get

{

return DataWorker.GetHallById((int)hallid);

}

}

[NotMapped]

public Film Film

{

get

{

return DataWorker.GetFilmById((int)filmid);

}

}

}

using System;

using Microsoft.EntityFrameworkCore.Migrations;

#nullable disable

namespace Cinema.Migrations

{

public partial class Qu : Migration

{

protected override void Up(MigrationBuilder migrationBuilder)

{

migrationBuilder.CreateTable(

name: "Genres",

columns: table => new

{

id = table.Column<int>(type: "int", nullable: false)

.Annotation("SqlServer:Identity", "1, 1"),

genre = table.Column<string>(type: "nvarchar(max)", nullable: false)

},

constraints: table =>

{

table.PrimaryKey("PK\_Genres", x => x.id);

});

migrationBuilder.CreateTable(

name: "halls",

columns: table => new

{

id = table.Column<int>(type: "int", nullable: false)

.Annotation("SqlServer:Identity", "1, 1"),

Hall = table.Column<string>(type: "nvarchar(max)", nullable: false)

},

constraints: table =>

{

table.PrimaryKey("PK\_halls", x => x.id);

});

migrationBuilder.CreateTable(

name: "Films",

columns: table => new

{

id = table.Column<int>(type: "int", nullable: false)

.Annotation("SqlServer:Identity", "1, 1"),

Genreid = table.Column<int>(type: "int", nullable: false),

hallid = table.Column<int>(type: "int", nullable: false),

sessionid = table.Column<int>(type: "int", nullable: false),

Date = table.Column<DateTime>(type: "datetime2", nullable: false),

Years = table.Column<DateTime>(type: "datetime2", nullable: false)

},

constraints: table =>

{

table.PrimaryKey("PK\_Films", x => x.id);

table.ForeignKey(

name: "FK\_Films\_Genres\_Genreid",

column: x => x.Genreid,

principalTable: "Genres",

principalColumn: "id",

onDelete: ReferentialAction.Cascade);

table.ForeignKey(

name: "FK\_Films\_halls\_hallid",

column: x => x.hallid,

principalTable: "halls",

principalColumn: "id",

onDelete: ReferentialAction.Cascade);

});

migrationBuilder.CreateTable(

name: "sessions",

columns: table => new

{

id = table.Column<int>(type: "int", nullable: false)

.Annotation("SqlServer:Identity", "1, 1"),

filmid = table.Column<int>(type: "int", nullable: false),

hallid = table.Column<int>(type: "int", nullable: false),

price = table.Column<int>(type: "int", nullable: false),

Date = table.Column<DateTime>(type: "datetime2", nullable: false)

},

constraints: table =>

{

table.PrimaryKey("PK\_sessions", x => x.id);

table.ForeignKey(

name: "FK\_sessions\_Films\_filmid",

column: x => x.filmid,

principalTable: "Films",

principalColumn: "id",

onDelete: ReferentialAction.Cascade);

table.ForeignKey(

name: "FK\_sessions\_halls\_hallid",

column: x => x.hallid,

principalTable: "halls",

principalColumn: "id",

onDelete: ReferentialAction.Cascade);

});

migrationBuilder.CreateTable(

name: "Tickets",

columns: table => new

{

id = table.Column<int>(type: "int", nullable: false)

.Annotation("SqlServer:Identity", "1, 1"),

filmid = table.Column<int>(type: "int", nullable: false),

hallid = table.Column<int>(type: "int", nullable: false),

Date = table.Column<DateTime>(type: "datetime2", nullable: false),

number\_rows = table.Column<string>(type: "nvarchar(max)", nullable: false),

number\_place = table.Column<string>(type: "nvarchar(max)", nullable: false)

},

constraints: table =>

{

table.PrimaryKey("PK\_Tickets", x => x.id);

table.ForeignKey(

name: "FK\_Tickets\_Films\_filmid",

column: x => x.filmid,

principalTable: "Films",

principalColumn: "id",

onDelete: ReferentialAction.Cascade);

table.ForeignKey(

name: "FK\_Tickets\_halls\_hallid",

column: x => x.hallid,

principalTable: "halls",

principalColumn: "id",

onDelete: ReferentialAction.Cascade);

});

migrationBuilder.CreateIndex(

name: "IX\_Films\_Genreid",

table: "Films",

column: "Genreid");

migrationBuilder.CreateIndex(

name: "IX\_Films\_hallid",

table: "Films",

column: "hallid");

migrationBuilder.CreateIndex(

name: "IX\_Films\_sessionid",

table: "Films",

column: "sessionid");

migrationBuilder.CreateIndex(

name: "IX\_sessions\_filmid",

table: "sessions",

column: "filmid");

migrationBuilder.CreateIndex(

name: "IX\_sessions\_hallid",

table: "sessions",

column: "hallid");

migrationBuilder.CreateIndex(

name: "IX\_Tickets\_filmid",

table: "Tickets",

column: "filmid");

migrationBuilder.CreateIndex(

name: "IX\_Tickets\_hallid",

table: "Tickets",

column: "hallid");

migrationBuilder.AddForeignKey(

name: "FK\_Films\_sessions\_sessionid",

table: "Films",

column: "sessionid",

principalTable: "sessions",

principalColumn: "id",

onDelete: ReferentialAction.Cascade);

}

protected override void Down(MigrationBuilder migrationBuilder)

{

migrationBuilder.DropForeignKey(

name: "FK\_Films\_Genres\_Genreid",

table: "Films");

migrationBuilder.DropForeignKey(

name: "FK\_Films\_halls\_hallid",

table: "Films");

migrationBuilder.DropForeignKey(

name: "FK\_sessions\_halls\_hallid",

table: "sessions");

migrationBuilder.DropForeignKey(

name: "FK\_Films\_sessions\_sessionid",

table: "Films");

migrationBuilder.DropTable(

name: "Tickets");

migrationBuilder.DropTable(

name: "Genres");

migrationBuilder.DropTable(

name: "halls");

migrationBuilder.DropTable(

name: "sessions");

migrationBuilder.DropTable(

name: "Films");

}

}

}

// <auto-generated />

using System;

using Cinema.Model.Data;

using Microsoft.EntityFrameworkCore;

using Microsoft.EntityFrameworkCore.Infrastructure;

using Microsoft.EntityFrameworkCore.Metadata;

using Microsoft.EntityFrameworkCore.Storage.ValueConversion;

#nullable disable

namespace Cinema.Migrations

{

[DbContext(typeof(ApplicationContext))]

partial class ApplicationContextModelSnapshot : ModelSnapshot

{

protected override void BuildModel(ModelBuilder modelBuilder)

{

#pragma warning disable 612, 618

modelBuilder

.HasAnnotation("ProductVersion", "6.0.5")

.HasAnnotation("Relational:MaxIdentifierLength", 128);

SqlServerModelBuilderExtensions.UseIdentityColumns(modelBuilder, 1L, 1);

modelBuilder.Entity("Cinema.Model.Film", b =>

{

b.Property<int>("id")

.ValueGeneratedOnAdd()

.HasColumnType("int");

SqlServerPropertyBuilderExtensions.UseIdentityColumn(b.Property<int>("id"), 1L, 1);

b.Property<DateTime>("Date")

.HasColumnType("datetime2");

b.Property<int?>("Genreid")

.IsRequired()

.HasColumnType("int");

b.Property<DateTime>("Years")

.HasColumnType("datetime2");

b.Property<int?>("hallid")

.IsRequired()

.HasColumnType("int");

b.Property<int?>("sessionid")

.IsRequired()

.HasColumnType("int");

b.HasKey("id");

b.HasIndex("Genreid");

b.HasIndex("hallid");

b.HasIndex("sessionid");

b.ToTable("Films");

});

modelBuilder.Entity("Cinema.Model.Genre", b =>

{

b.Property<int>("id")

.ValueGeneratedOnAdd()

.HasColumnType("int");

SqlServerPropertyBuilderExtensions.UseIdentityColumn(b.Property<int>("id"), 1L, 1);

b.Property<string>("genre")

.IsRequired()

.HasColumnType("nvarchar(max)");

b.HasKey("id");

b.ToTable("Genres");

});

modelBuilder.Entity("Cinema.Model.hall", b =>

{

b.Property<int>("id")

.ValueGeneratedOnAdd()

.HasColumnType("int");

SqlServerPropertyBuilderExtensions.UseIdentityColumn(b.Property<int>("id"), 1L, 1);

b.Property<string>("Hall")

.IsRequired()

.HasColumnType("nvarchar(max)");

b.HasKey("id");

b.ToTable("halls");

});

modelBuilder.Entity("Cinema.Model.session", b =>

{

b.Property<int>("id")

.ValueGeneratedOnAdd()

.HasColumnType("int");

SqlServerPropertyBuilderExtensions.UseIdentityColumn(b.Property<int>("id"), 1L, 1);

b.Property<DateTime>("Date")

.HasColumnType("datetime2");

b.Property<int?>("filmid")

.IsRequired()

.HasColumnType("int");

b.Property<int?>("hallid")

.IsRequired()

.HasColumnType("int");

b.Property<int?>("price")

.IsRequired()

.HasColumnType("int");

b.HasKey("id");

b.HasIndex("filmid");

b.HasIndex("hallid");

b.ToTable("sessions");

});

modelBuilder.Entity("Cinema.Model.Ticket", b =>

{

b.Property<int>("id")

.ValueGeneratedOnAdd()

.HasColumnType("int");

SqlServerPropertyBuilderExtensions.UseIdentityColumn(b.Property<int>("id"), 1L, 1);

b.Property<DateTime>("Date")

.HasColumnType("datetime2");

b.Property<int?>("filmid")

.IsRequired()

.HasColumnType("int");

b.Property<int?>("hallid")

.IsRequired()

.HasColumnType("int");

b.Property<string>("number\_place")

.IsRequired()

.HasColumnType("nvarchar(max)");

b.Property<string>("number\_rows")

.IsRequired()

.HasColumnType("nvarchar(max)");

b.HasKey("id");

b.HasIndex("filmid");

b.HasIndex("hallid");

b.ToTable("Tickets");

});

modelBuilder.Entity("Cinema.Model.Film", b =>

{

b.HasOne("Cinema.Model.Genre", "Genre")

.WithMany()

.HasForeignKey("Genreid")

.OnDelete(DeleteBehavior.Cascade)

.IsRequired();

b.HasOne("Cinema.Model.hall", "hall")

.WithMany()

.HasForeignKey("hallid")

.OnDelete(DeleteBehavior.Cascade)

.IsRequired();

b.HasOne("Cinema.Model.session", "session")

.WithMany()

.HasForeignKey("sessionid")

.OnDelete(DeleteBehavior.Cascade)

.IsRequired();

b.Navigation("Genre");

b.Navigation("hall");

b.Navigation("session");

});

modelBuilder.Entity("Cinema.Model.session", b =>

{

b.HasOne("Cinema.Model.Film", "film")

.WithMany()

.HasForeignKey("filmid")

.OnDelete(DeleteBehavior.Cascade)

.IsRequired();

b.HasOne("Cinema.Model.hall", "hall")

.WithMany()

.HasForeignKey("hallid")

.OnDelete(DeleteBehavior.Cascade)

.IsRequired();

b.Navigation("film");

b.Navigation("hall");

});

modelBuilder.Entity("Cinema.Model.Ticket", b =>

{

b.HasOne("Cinema.Model.Film", "film")

.WithMany()

.HasForeignKey("filmid")

.OnDelete(DeleteBehavior.Cascade)

.IsRequired();

b.HasOne("Cinema.Model.hall", "hall")

.WithMany()

.HasForeignKey("hallid")

.OnDelete(DeleteBehavior.Cascade)

.IsRequired();

b.Navigation("film");

b.Navigation("hall");

});

#pragma warning restore 612, 618

}

}

}