МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

Учреждение образования «БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ»

Факультет Информационных технологий

Кафедра Информационные системы и технологии

Специальность 1–40 01 01 «Программное обеспечение информационных

технологий»

Специализация Программирование интернет-приложений

**ПОЯСНИТЕЛЬНАЯ ЗАПИСКА**

**К КУРСОВОМУ ПРОЕКТУ НА ТЕМУ:**

«Разработка базы данных сети кинотеатров с применением мультимедийных типов данных в базе данных MS SQL Server»

Выполнил студент Хадасевич Юлиана Андреевна

(Ф.И.О.)

Руководитель проекта пр.-ст. Бондарчик Е.Н

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

Заведующий кафедрой к.т.н., доц. Смелов В.В.

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

Консультант: пр.-ст. Бондарчик Е.Н

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

Нормоконтролер: пр.-ст. Бондарчик Е.Н

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

Курсовой проект защищен с оценкой

Минск 2019

**Реферат**

ФИО

Подпись

Дата

Лист

2

КП 00.00.ПЗ

Разраб.

*Хадасевич Ю. А*

Провер.

Бондарчик Е.Н.

Н. контр.

.

Утверд.

Реферат

Лит.

Листов

1

*БГТУ 71191005, 2019*

Пояснительная записка курсового проекта содержит 32 страницы пояснительной записки, 13 таблиц, 32 иллюстрации, 9 источников литературы, 3 приложения.

C#, WPF, MICROSOFT SQL SERVER, БАЗА ДАННЫХ, SQL, T-SQL, ADO.NET

Целью курсового проекта является создание базы данных сети кинотеатров, ознакомление и применение мультимедийных типов данных в базе данных.

В первой главе проводится аналитический обзор прототипов по тематике курсового проекта и содержится описание технологий, использованных во время выполнения проекта.

Вторая глава посвящена процессу проектирования модели базы данных.

В третьей главе описывается процесс разработки, принципы функционирования и назначение созданных компонент проекта.

В четвёртой главе описаны процедуры для работы с XML.

В пятой главе содержатся сведения об использованной технологии.

Шестая глава посвящена тестированию производительности приложения и самого приложения.

В седьмой главе описывается руководство пользователя

В заключении приведены итоги курсового проектирования.

ФИО

Подпись

Дата

Лист

3

КП 00.00.ПЗ

Разраб.

*Хадасевич Ю. А*

Провер.

Бондарчик Е.Н.

Н. контр.

.

Утверд.

Содержание

Лит.

Листов

1

*БГТУ 71191005, 2019*

Содержание

[Введение 6](#_Toc27209854)

[1 Постановка задачи 7](#_Toc27209855)

[1.1 Обзор прототипов 7](#_Toc27209856)

[1.2 Анализ прототипов 9](#_Toc27209857)

[1.3 Описание используемых технологий 9](#_Toc27209858)

[2 Проектирование модели базы данных 11](#_Toc27209859)

[3 Разработка необходимых объектов 12](#_Toc27209860)

[3.1 Таблицы 12](#_Toc27209861)

[3.2 Пользователи 17](#_Toc27209862)

[3.3 Процедуры 17](#_Toc27209863)

[4 Описание процедур импорта и экспорта данных 19](#_Toc27209864)

[4.1 Процедура импорта данных из XML-файла 19](#_Toc27209865)

[4.2 Процедура экспорта данных в XML-файл 19](#_Toc27209866)

[5 Описание технологии 20](#_Toc27209867)

[6 Тестирование 21](#_Toc27209868)

[6.1 Тестирование производительности базы данных 21](#_Toc27209869)

[6.2 Тестирование работоспособности приложения 22](#_Toc27209870)

[7 Руководство пользователя 25](#_Toc27209871)

[Заключение 31](#_Toc27209872)

[Список литературных источников 32](#_Toc27209873)

[Приложение А 33](#_Toc27209874)

[Приложение Б 34](#_Toc27209875)

[Приложение В 42](#_Toc27209876)

# Введение

ФИО

Подпись

Дата

Лист

6

КП 00.00.ПЗ

Разраб.

*Хадасевич Ю.А.*

Провер.

Бондарчик Е.Н.

Н. контр.

.

Утверд.

Введание

Лит.

Листов

1

*БГТУ 71191005, 2019*

Сегодня трудно себе представить информационную систему, которая не имела бы в качестве основы или важной составляющей базу данных. Сфера развлечений также не может обойтись без внедрения различных технологий для повышения удобства пользователей.

Одним из примеров задач, в которых мы бы хотели совершать операции автономно и без дополнительных затрат времени является покупка билетов в кинотеатр. Теперь возможно покупать билеты в кино в любое удобное время и в любом месте.

Целью моего курсового проекта является создание базы данных сети кинотеатров, ознакомление и применение мультимедийных типов данных в базе данных, поскольку визуальная информация — это именно то, что ассоциируется с сеансом в кинотеатре.

Основными задачами курсовой работы являются:

* провести аналитический обзор литературы;
* спроектировать базу данных;
* реализовать функциональность базы данных;
* провести тестирование используемой технологии в базе данных;
* разработать приложение для работы с базой данных;
* написать руководство пользователя.

В соответствии с заданием курсового проекта для проектирования базы данных сети кинотеатров используется система управления базами данных Microsoft SQL Server.

Также в задачу разработчика входит создание дружелюбного пользовательского интерфейся для совершения действий над базой данных сети кинотеатров. Для этого создаётся приложение, которое будет облегчать пользователю работу с базой данных.

Для разработки приложения использовалась технология WPF и язык программирования C#. Для взаимодействия приложения с базой данных использовалась технология ADO.NET.

# Постановка задачи

ФИО

Подпись

Дата

Лист

7

КП 01.00.ПЗ

Разраб.

*Хадасевич Ю. А*

Провер.

Бондарчик Е.Н.

Н. контр.

.

Утверд.

1 Постановка задачи

Лит.

Листов

4

*БГТУ 71191005, 2019*

В соответствии с заданием курсового проекта следует не только создать базу данных, но и разработать программное средство, которое должно в полной мере показать возможности базы данных. Для того чтобы сформировать окончательные требования к проектируемому программному средству сначала рассмотрим прототипы программных средств того же направления.

## 1.1 Обзор прототипов

На сегодняшний день существует несколько сайтов, которые дают своим клиентам возможность узнать информацию о киносеансах и приобрести билеты. Рассмотрим примеры web-приложений подобных сайтов.

ByCard[1]

Одним из самых популярных сайтов для покупки билетов является именно ByCard. На этом сайте можно не только выбрать, но и приобрести понравившийся билет. Подробная информация о фильме, возможность поиска по дате и наличие трейлеров делают этот сайт лидером среди других.

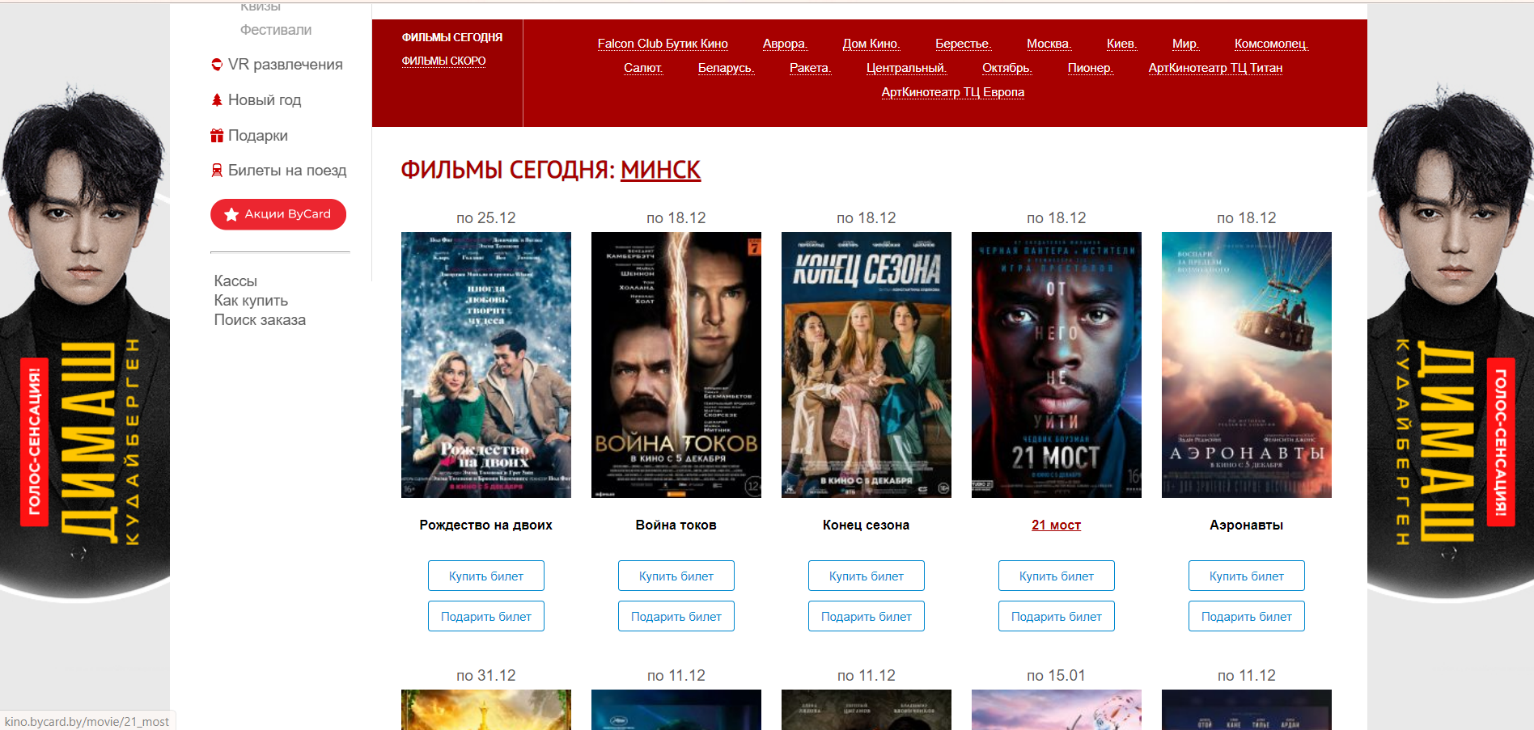


Рисунок 1.1 – Интерфейс сайта ByCard

Афиша TUT.by[2]

Данный сайт является самым популярным среди сайтов для выбора досуга. TUT.by считается одним из самых популярных новостных источников для белорусов и имеет множество дополнительных ответвлений, таких как афиша TUT.by. Удобный выбор дат сеансов, сортировка по жанрам и кинотеатрам – всё это

положительные стороны сайта. С другой стороны, он имеет небольшой минус, а именно невозможность покупки билета. Для этого происходит переход на сторонний ресурс.

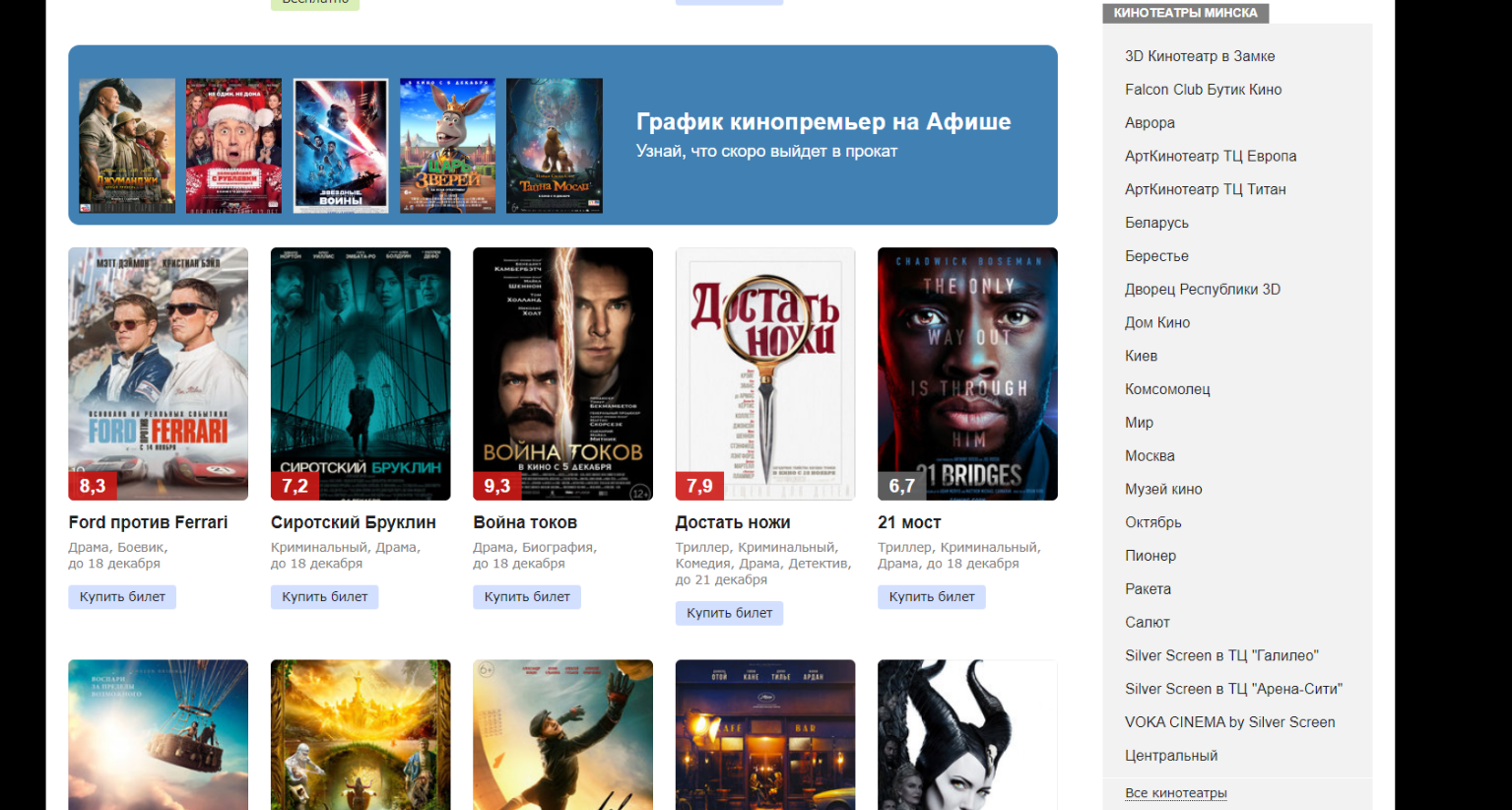


Рисунок 1.2 – Интерфейс сайта афиша TUT.by

КиноМинска[3]

КиноМинска – это государственный сайт для просмотра сеансов кинотеатров. Несомненным преимуществом является то, что данный ресурс полностью посвящён такому виду развлечения, как кинотеатр, в отличие от вышеприведённых сайтов. Также можно пересылать информацию о сеансе посредством социальных сетей.

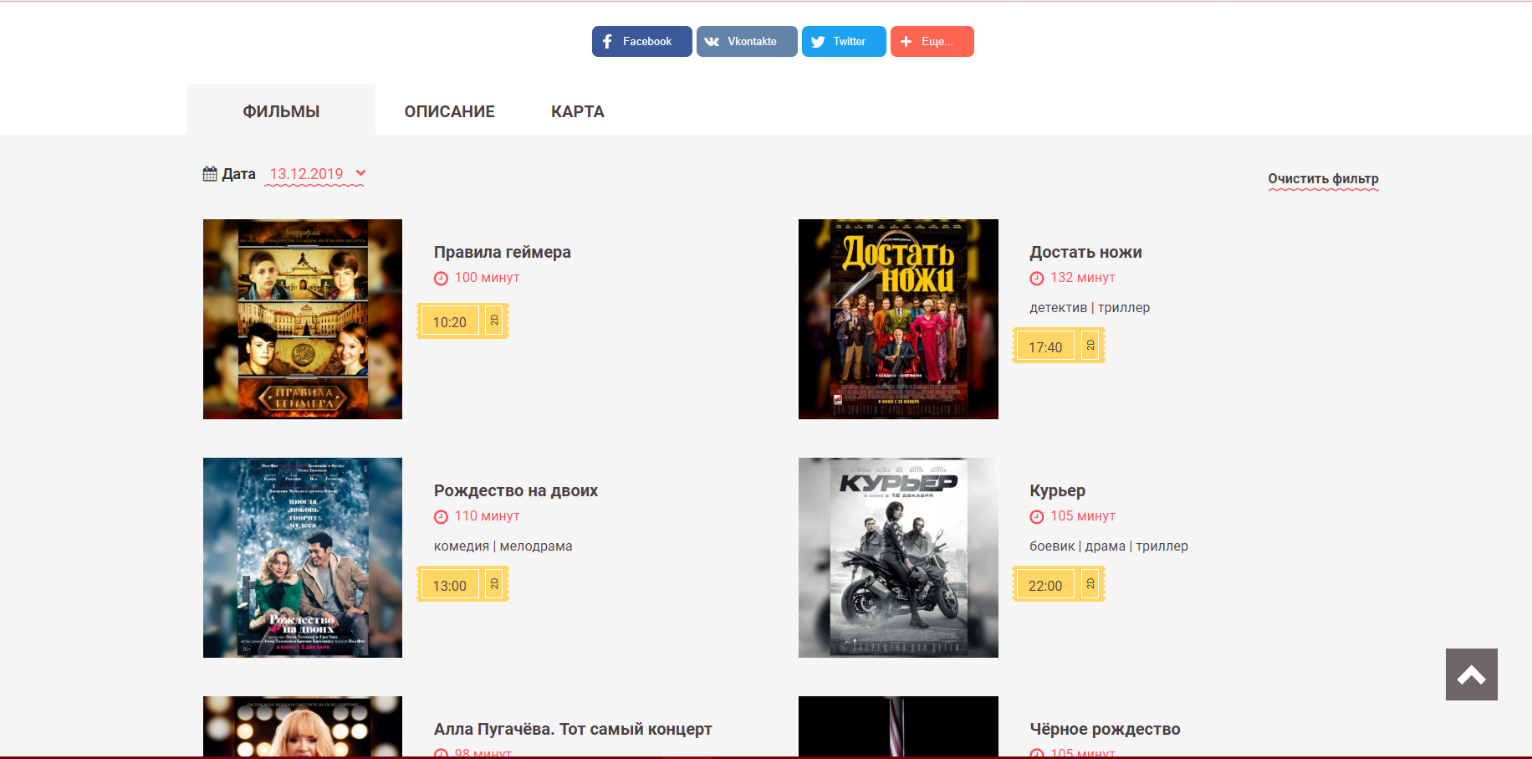


Рисунок 1.3 – Интерфейс сайта КиноМинска

**1.2 Анализ прототипов**

Проанализировав прототипы сайтов, можно отметить ожидаемую схожесть в их функциональности и интерфейсе.

Исходя из полученных результатов, можно сформулировать основные функциональные требования курсовой работы:

* регистрация и авторизация клиентов;
* авторизация администратора;
* представление списка фильмов клиенту, где в качестве визуального ориентира выступает постер фильма;
* покупка клиентом билетов;
* возможности поиска фильмов по различным критериям, таким как жанры, кинотеатры;
* просмотр клиентом информации о фильмах;
* просмотр клиентом трейлеров к фильмам;
* добавление администратором фильмов;
* добавление администратором сессий и кинотеатров;
* редактирование информации о кинотеатрах и фильмах;
* удаление информации о кинотеатрах и фильмах;

Данные требования будут реализованы в базе данных в виде хранимых процедур, которые описаны в главе 3 данной пояснительной записки.

**1.3 Описание используемых технологий**

СУБД существует огромное множество: Oracle, MS SQL Server, Microsoft Access, MySql и так далее. Для организации работы с базой данных в данной работе было решено использовать одну из наиболее популярных СУБД «Microsoft SQL Server» [4], так как она предоставляет необходимые возможности оперирования объектами базы данных.

В качестве интерфейса прикладного программирования был выбран API-интерфейс — Windows Presentation Foundation (WPF) [5], платформа уровня представления для построения графических интерфейсов. Основывается на векторной системе визуализации и ориентирована на разработку клиентских Windows приложений, базирующихся на технологии Microsoft.NET.

WPF предоставляет средства для создания визуального интерфейса, включая язык XAML (eXtensible Application Markup Language), элементы управления, привязку данных, макеты, двухмерную и трёхмерную графику, анимацию, стили, шаблоны, документы, текст, мультимедиа и оформление. Что является преимуществом перед более ранней технологией создания пользовательских интерфейсов — Windows Forms.

Для работы с WPF использовался объектно-ориентированный язык программирования с С-подобным синтаксисом — С#, разработанный для создания приложений на платформе Microsoft .NET Framework [6].

Чтобы осуществлять связь между базой данных и приложением на C# необходим посредник. И именно таким посредником является технология ADO.NET [7]. ADO.NET – интерфейс прикладного уровня Microsoft (набор классов, предоставляющих службы доступа к данным). Является не развитием более ранней технологии ADO, а самостоятельной технологией, частью фреймворка .NET. Для обеспечения защиты паролей клиентов применяется технология хеширования.

Двоичный тип данных «Microsoft SQL Server» – это одна из тех функций, которую большинство разработчиков часто не используют, но оказывается, что данный тип является достаточно полезным для хранения мультимедийной информации. В проекте для хранения графической информации я использую тип данных varbinary(max).

Также в процессе выбора технологии была рассмотрена технология FileStream, где данные хранятся в файловой системе.

По сведениям из официальной документации [8] объекты FILESTREAM следует использовать в том случае, когда средний размер сохраняемых объектов превышает 1 МБ. При работе с объектами меньшего размера сохранение больших двоичных объектов (BLOB) типа varbinary(max) в базе данных часто позволяет добиться лучшей производительности потоков. А поскольку хранящиеся в базе данных картинки не превышают 1 МБ, то было принято решение не использовать технологию FileStream.

# 

# Проектирование модели базы данных

ФИО

Подпись

Дата

Лист

11

КП 02.00.ПЗ

Разраб.

*Хадасевич Ю. А*

Провер.

Бондарчик Е.Н.

Н. контр.

.

Утверд.

2 Проектирование модели базы данных

Лит.

Листов

1

*БГТУ 71191005, 2019*

Для реализации поставленной задачи была создана база данных Cinema. Диаграмма структуры полученной базы данных, разработанной в СУБД «Microsoft SQL Server», представлена в приложении А.

Для базы данных было разработано 13 таблиц, которые связаны друг с другом внешними ключами.

Таблица Users хранит данные о клиентах.

Таблица Purchase хранит данные о произведённой покупке билетов.

Таблица Tickets содержит информацию о билетах.

Таблица Session предназначена для данных о сеансе.

Таблица Hall хранит данные о зале кинотеатра.

Таблица Cinema хранит данные о кинотеатрах.

Таблица Movie создана для информации о фильме.

Таблица Genre содержит набор используемых жанров.

Таблица Genre\_Movie предназначена для создания связи многие ко многим между таблицей Movie и Genre.

Таблица Actor содержит информацию об актёрах.

Таблица Actor\_Movie предназначена для создания связи многие ко многим между таблицей Movie и Actor.

Таблица Studio хранит информацию о киностудиях.

Таблица Country хранит набор страна.

Создание перечисленных таблиц с установлением внешних связей описано в главе 3.

# Разработка необходимых объектов

ФИО

Подпись

Дата

Лист

12

КП 03.00.ПЗ

Разраб.

*Хадасевич Ю. А*

Провер.

Бондарчик Е.Н.

Н. контр.

.

Утверд.

3 Разработка необходимых объектов

Лит.

Листов

7

*БГТУ 71191005, 2019*

## Таблицы

Таблицы являются неотъемлемой частью любой реляционной базы данных. Краткая характеристика каждой из таблиц была предоставлена в разделе 2, а код их создания можно увидеть в Приложении Б. Ниже мы рассмотрим каждую таблицу подробнее.

Таблица Cinema состоит из семи столбцов:

– ID– хранит идентификатор кинотеатра;

– NAME – хранит название кинотеатра;

– ADDRESS – хранит адрес кинотеатра;

– WEBSITE – хранит адрес сайта кинотеатра;

– CITY – хранит город в котором кинотеатр расположен;

– NUMBER\_OF\_HALLS – хранит количество залов в кинотеатре;

– TIMETABLE – хранит расписание работы кинотеатра.

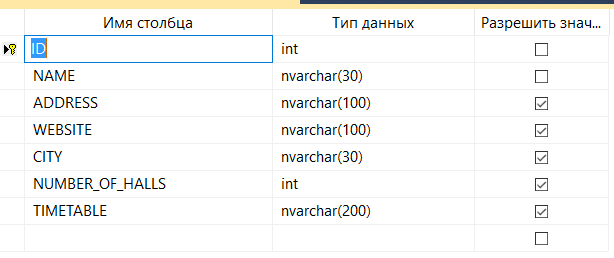


Рисунок 3.1 – Описание таблицы Cinema

Таблица Cinema состоит из пяти столбцов:

– ID– хранит идентификатор зала;

– NAME – хранит название зала;

–CINEMA\_ID – идентификатор кинотеатра, внешний ключ к таблице CINEMA;

– ROWS – хранит количество рядов для этого зала;

– SEATS – хранит количество мест в ряду для этого зала.

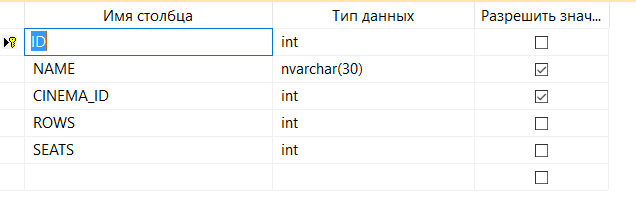


Рисунок 3.2 – Описание таблицы Hall

Таблица Session состоит из семи столбцов:

– ID– хранит идентификатор сеанса;

– MOVIE\_ID –идентификатор фильма, внешний ключ к таблице MOVIE;

– HALL\_ID – идентификатор зала, в котором будет проходить сеанс, внешний ключ к таблице HALL;

– DATE – дата сеанса;

– TIME – время сеанса;

– COST – стоимость одного билета на этом сеансе;

– FREESEATS – количество свободных мест у сеанса.

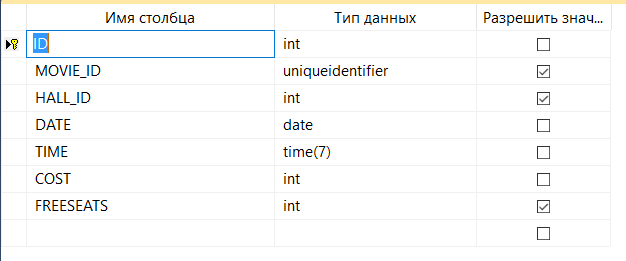


Рисунок 3.3 – Описание таблицы Session

Таблица Tickets состоит из пяти столбцов:

– ID– хранит идентификатор билета;

– SESSION\_ID – идентификатор сеанса на который куплен билет, внешний ключ к таблице SESSION;

–PURCHASE\_ID – идентификатор покупки, внешний ключ к таблице PURCHASE;

– ROW – номер ряда для этого билета;

– SEAT – номер места для этого билета.

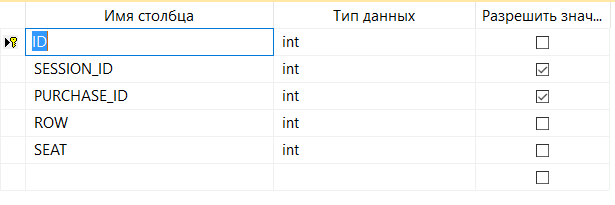


Рисунок 3.4 – Описание таблицы Tickets

Таблица Purchase состоит из пяти столбцов:

– ID– хранит идентификатор покупки;

–USER\_ID – идентификатор пользователя, который совершил покупку, внешний ключ к таблице USERS;

– DATE –дата покупки;

– UNICK\_TICKET – случайно сгенерированный код покупки;

– PRICE – стоимость покупки.

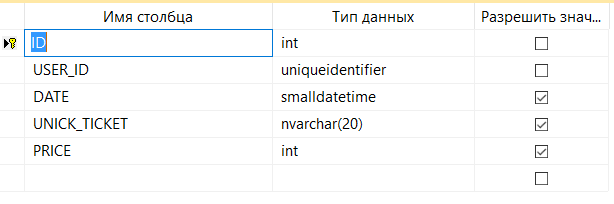


Рисунок 3.5 – Описание таблицы Purchase

Таблица Users состоит из четырёх столбцов:

– ID – хранит идентификатор пользователя;

–LOGIN – логин пользователя;

– PASSWORD –хешированный пароль;

– EMAIL – адрес электронной почты.

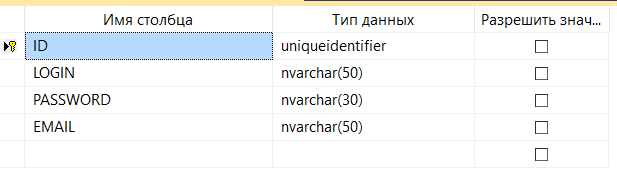


Рисунок 3.6 – Описание таблицы Users

Таблица Users состоит из четырёх столбцов:

– ID – хранит идентификатор страны;

–NAME– название страны.

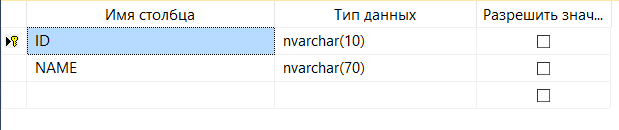


Рисунок 3.7 – Описание таблицы Country

Таблица Actor состоит из четырёх столбцов:

– ID – хранит идентификатор актёра;

– NAME – имя актёра;

– SURNAME –фамилия актёра;

– COUNTRY\_ID – идентификатор страны проживания актёра, внешний ключ к таблице COUNTRY.

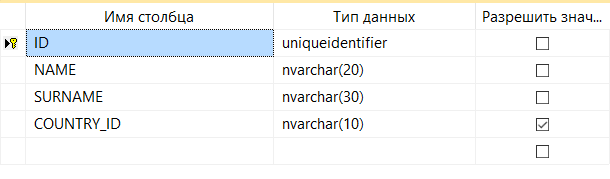


Рисунок 3.8 – Описание таблицы Actor

Таблица Actor\_Movie состоит из двух столбцов:

– ACTOR\_ID – хранит идентификатор актёра;

– MOVIE\_ID – идентификатор фильма.

Служит для создания связи многие ко многим между таблицами MOVIE и ACTOR.

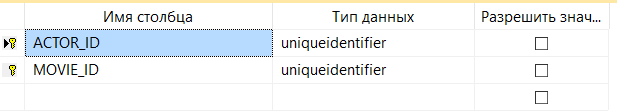


Рисунок 3.9 – Описание таблицы Actor\_Movie

Таблица Movie состоит из девяти столбцов:

– ID –идентификатор фильма;

– NAME –название фильма;

– RELEASE – дата выхода фильма;

– COUNTRY\_ID – идентификатор страны в которой снят фильм, внешний ключ к таблице COUNTRY;

– RUNNING\_TIME – длительность фильма;

– STUDIO\_ID – идентификатор студии, внешний ключ к таблице STUDIO;

– PLOT –описание фильма;

– IMAGE – постер фильма;

– TRAILER – трейлер фильма.

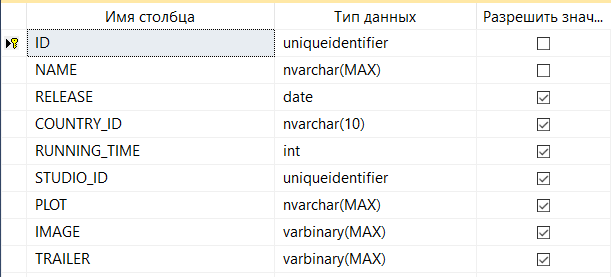


Рисунок 3.10 – Описание таблицы Movie

Таблица Genre состоит из двух столбцов:

–ID – хранит идентификатор жанра;

– NAME – название жанра.

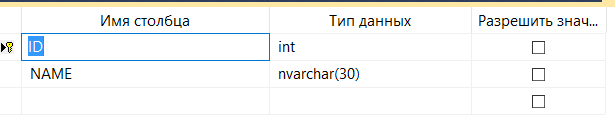


Рисунок 3.11 – Описание таблицы Genre

Таблица Genre\_Movie состоит из двух столбцов:

– GENRE\_ID – хранит идентификатор жанра;

– MOVIE\_ID – идентификатор фильма.

Служит для создания связи многие ко многим между таблицами MOVIE и GENRE.

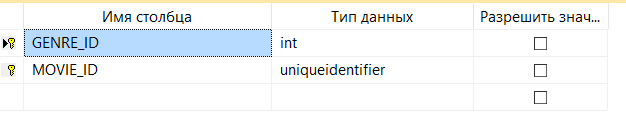


Рисунок 3.12 – Описание таблицы Genre\_Movie

Таблица Studio состоит из пяти столбцов:

– ID– хранит идентификатор киностудии;

–NAME – название киностудии;

– COUNTRY\_ID –идентификатор страны, в которой находится киностудия, внешний ключ к таблице COUNTRY;

– YEAR\_OF\_FOUNDATION – год основания студии;

– IMAGE – логотип студии.

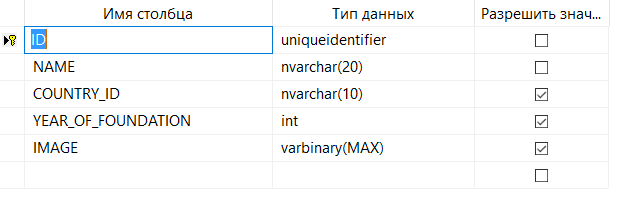


Рисунок 3.13 – Описание таблицы Studio

## Пользователи

Пользователь базы данных – это физическое или юридическое лицо, которое имеет доступ к БД и пользуется услугами информационной системы для получения информации. Пользователи базы данных получают права доступа для чтения, вставки, обновления и удаления конкретных объектов.

Администратор кинотеатра наделён привилегией на выполнение всех хранимых процедур, разработанных для базы данных.

Клиенту разрешено выполнять только процедуры для авторизации и регистрации, процедуры для получения информации из БД, процедуры для работы с XML данным, процедуры для добавления покупки и билета, а также процедуру для удаления билета.

Скрипты для создания логинов и пользователей базы данных, а также выделение привилегий на осуществление определённых операций с базой данных представлены в приложении Б.

## Процедуры

Использование хранимых процедур позволяет ограничить либо вообще исключить непосредственный доступ пользователей к таблицам базы данных, оставив пользователям только разрешения на выполнение хранимых процедур, обеспечивающих косвенный и строго регламентированный доступ к данным.

Всего было разработано 42 процедуры для работы с данными. Краткое описание процедур описано далее.

Пример создания хранимой процедуры CheckSeats, которая проверяет можно ли пользователю купить билеты на это место представлен в приложении Б.

Данная процедура принимает входной параметр row (ряд, выбранный пользователем), seat (место на этом ряду, выбранное пользователем), зал, в котором пользователь выбирает место и формирует выходной параметр message, который содержит в случае ошибки уточняющее сообщение о ней. Тело процедуры содержит проверку на существование данного места и его занятость другим билетом.

Authorisation – процедура для авторизации пользователя;

Registration – процедура для регистрации пользователя;

DeleteCinema – процедура для удаления кинотеатра;

DeleteMovie – процедура для удаления фильма;

DeleteSession – процедура для удаления сеанса;

DeleteTicket – процедура для удаления билета;

ExportIntoXML – процедура для уточнения данных, используемых для экспорта в XML.

ExportToXML – процедура для экспорта кинотеатра в файл \*.xml;

TextSearch – процедура для поиска фильма по названию или описанию;

GetActorInfo – процедура для вывода информации об актёрах;

GetCinemaInfo – для вывода информации о кинотеатрах;

GetMovieInfo – процедура для вывода информации о фильме;

GetSessionInfo – процедура для вывода информации о сеансах;

GetStudioInfo – процедура для вывода информации о киностудиях;

GetTicketInfo – процедура для вывода информации о билетах;

GetCountry – процедура для получения стран;

GetGenres – процедура для получения жанров;

GetHallInfo – процедура для вывода информации о зале;

GetMovies – процедура для вывода информации о фильмах;

GetSetActors – процедура для получения строки всех актёров фильма;

GetSetGenres – процедура для получения строки всех жанров фильма;

ImportFromXML –процедура для импорта кинотеатров из файла \*.xml;

InsertActor – процедура для добавления актёров;

InsertCinema – процедура для добавления кинотеатра;

InsertStudio – процедура для добавления киностудии;

InsertGenre – процедура для добавления жанра;

InsertHall – процедура для добавления зала;

InsertCountry – процедура для добавления страны;

InsertMovie – процедура для добавления фильма;

InsertPurchase – процедура для покупки билетов;

InsertSession – процедура для добавления сеанса;

InsertTicket– процедура для добавления билетов;

SelectFreeSeats – процедура для вывода свободных мест;

SelectPrice – процедура для вывода стоимости заказа;

UpdateCinema – процедура для обновления информации о кинотеатрах;

UpdateMovie – процедура для обновления информации о фильмах;

UserInfo – процедура получения адреса электронной почты и идентификатора пользователя по его логину и паролю;

AddActor – процедура добавления актёра к фильму;

AddGenre – процедура добавления жанра к фильму;

AllowRows – процедура получения набора рядов зала в сеансе;

AllowSeats – процедура получения набора свободных мест в определённом ряду в сеансе;

CheckSeats – процедура на проверку места на доступность.

Все скрипты хранимых процедур приложены в отдельных файлах в корне директории прилагаемого диска.

# Описание процедур импорта и экспорта данных

ФИО

Подпись

Дата

Лист

1

КП 04.00.ПЗ

Разраб.

*Хадасевич Ю. А*

Провер.

Бондарчик Е.Н.

Н. контр.

.

Утверд.

4 Описание процедур импорта и экспорта данных

Лит.

Листов

1

*БГТУ 71191005, 2019*

База данных обычно имеет не самостоятельную ценность, является частью информационной системы. Независимо от того, как устроена эта система, на противоположном от БД конце находится интерфейс взаимодействия с пользователем, и задача программиста предоставить простой и понятный способ работы с хранящимися в БД данными и объектами.

При всей своей отлаженности и очевидности, классический способ хранения и представления объектов развитой структуры имеет и вполне определенные недостатки и может вызывать проблемы, с которыми сталкивался любой разработчик, пытавшийся реализовать таким способом достаточно сложную систему. В некоторых ситуациях, решить эти проблемы позволяет хранение объекта в виде XML.

## Процедура импорта данных из XML-файла

Для импорта используется стандартная функция, входящая в набор OLE DB – OPENROWSET, в которую передаются параметры о типе импортируемых данных и пути файла, где они находятся. С помощью процедуры ImportFromXML пользователь может добавить билет в свой кабинет из XML-файла. Листинг данной процедуры можно увидеть в приложении Б.

* 1. Процедура экспорта данных в XML-файл

Для работы процедуры экспорта данных в XML-файл необходимо изменить параметры конфигурации dbo при помощи дополнительного скрипта. Также в приложении Б представлена процедура ExportToXML, которая вызывает дополнительную процедуру ExportIntoXML и записывает её результат по пути, указанному в path.

Процедура ExportIntoXML по входным параметрам возвращает данные, которые затем вставляются в XML-файл. Её реализацию можно увидеть в приложении Б.

# Описание технологии

ФИО

Подпись

Дата

Лист

20

КП 05.00.ПЗ

Разраб.

*Хадасевич Ю. А*

Провер.

Бондарчик Е.Н.

Н. контр.

.

Утверд.

5 Описание технологии

Лит.

Листов

1

*БГТУ 71191005, 2019*

Для начала можно сравнить два типа файлов, ASCII и Binary. Большинство значений, хранящихся в SQL Server, состоят из символов ASCII. Двоичные файлы, однако, содержат как символы ASCII, так и специальные управляющие символы, и комбинации байтов. BLOB (Binary Large Object) – это двоичные файлы большого размера. Хотя ограничение в 8 КБ более чем достаточно для большинства типов данных, оно является проблемой для большинства BLOB-объектов.

Сначала нужно ответить на вопрос где хранить мультимедийные объекты. Либо приложение может хранить информацию о пути файловой системы, ведущую к документу, например, путь к изображению внутри столбца varchar, оставляя документ в файловой системе, или мы могли бы поместить файл изображения внутри столбца с типом данных varbinary(max). Max говорит о максимально допустимом размере для хранения данных с помощью типа varbinary равном 231-1 байт. Также возможно указать тип данных varbinary(n), где n – количество байт(1–8000). Несколько ключевых вопросов помогут определить оптимальный вариант.

Насколько сильно объект будет мешать производительности, например, если это потоковое видео? С такими данными файловая система может работать лучше.

Насколько большие объекты будут храниться в базе данных, если объекты превышают 1 МБ[9], да ещё и существенно, то файловая система обычно будет более эффективна при представлении или чтении объекта. Если двоичные файлы небольшие, то их хранение внутри базы данных будет более чем достаточно.

Если двоичные файлы часто изменяются и являются большими, файловая система может обрабатывать фрагментацию лучше, чем SQL Server.

В функции InsertMovie2 производится добавление фильма, где одним из параметров является путь к сохраняемому в базе данных файлу. Затем с помощью OPENROWSET (функция позволяет получить доступ к удаленным данным путем подключения к удаленному источнику данных), где мы указываем путь к файлу с параметром BULK и формат файла, вставляем файл. Листинг функции приведён в приложении Б.

Функция InsertMovie2 использовалась для процесса разработки объектов базы данных и была заменена на функцию InsertMovie.

В качестве параметров image и video процедуре InsertMovie из приложения указывается массив байт, что позволяет вставить мультимедийные данные в базу данных Cinema другим способом. Функция InsertMovie представлена в приложении Б.

# Тестирование

ФИО

Подпись

Дата

Лист

21

КП 06.00.ПЗ

Разраб.

*Хадасевич Ю. А*

Провер.

Бондарчик Е.Н.

Н. контр.

.

Утверд.

6 Тестирование

Лит.

Листов

4

*БГТУ 71191005, 2019*

* 1. Тестирование производительности базы данных

Для тестирования производительности была взята за основу таблица USERS, которую мы заполнили 100001 строками. После этого был применён SELECT-запрос к данной таблице на столбец ID и при помощи стандартных средств IDE MS SQL Server Management Studio оценена цена выборки к таблице. Результат данной оценки запроса приведён на рисунке 6.1.

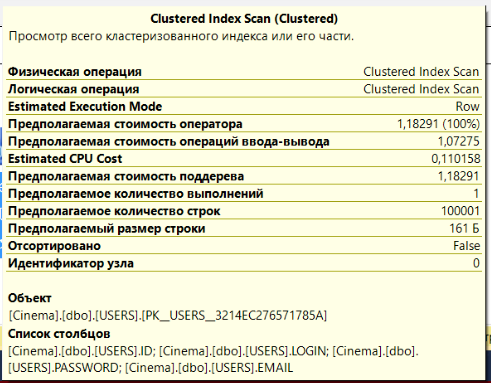


Рисунок 6.1 – Оценка запроса к таблице USERS с кластеризованным индексом

После проведения первоначальной оценки был построен некластеризованный индекс к таблице USERS, создание которого можно увидеть в приложении Б, и проведена оценка такого же SELECT-запроса к таблице USERS. Результаты, полученные во время оценки, представлены на рисунке 6.2.

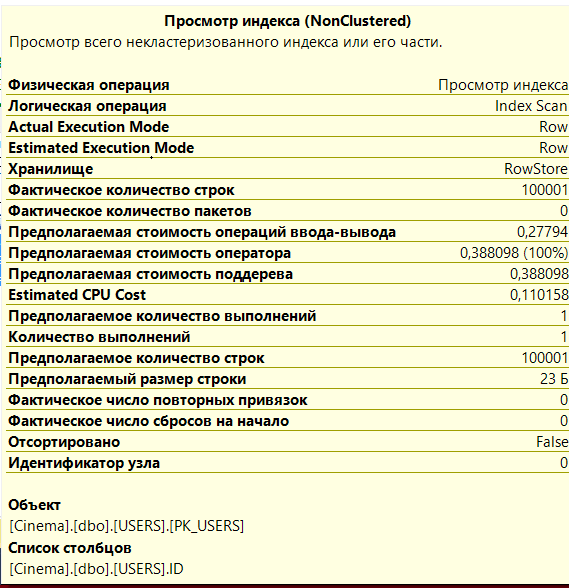


Рисунок 6.2 – Оценка запроса к таблице с построенным некластеризованным индексом

По результатам проведённых оценок, можно сделать вывод, что после создания некластеризованного индекса:

– стоимость по параметру “Предполагаемая стоимость операций ввода-вывода” стала в 3 раза меньше;

– стоимость по параметру “Предполагаемая стоимость оператора” снизилась в 3 раза;

– стоимость по параметру “Предполагаемая стоимость поддерева” уменьшилась в 3 раза.

Таким образом, постройка индекса к таблице была более чем оправдана, так как мы получили прирост производительности в 3 и более раза.

## Тестирование работоспособности приложения

Тестирование приложения предназначено для выявления возможных ошибок при разработке и для того, чтобы исключить действия пользователя, которые могут повлечь за собой аварийное завершение программы.

Рассмотрим регистрацию и авторизацию пользователя. Если пользователь не заполнит какие-нибудь поля при регистрации или авторизации, то ему будет предложено заполнить все поля.

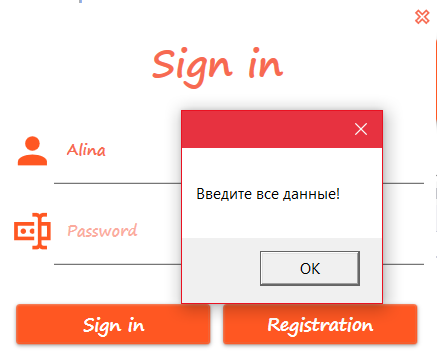
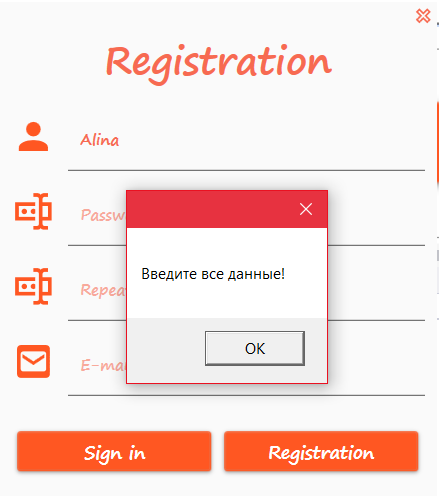


Рисунок 6.3 – Проверка заполнения полей при регистрации и авторизации

В окне авторизации пользователя, если пользователь введёт данные, которых нет в базе данных, то пользователю предложат ввести корректные данные.

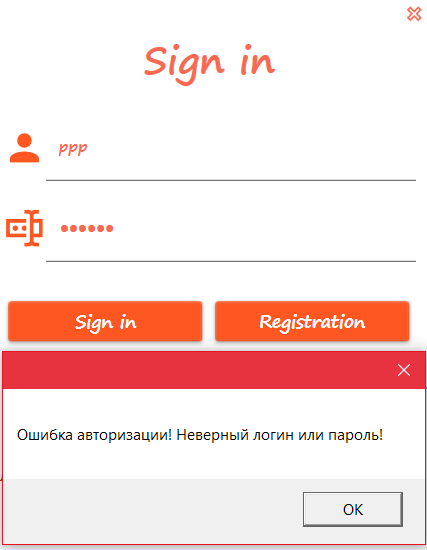


Рисунок 6.4 – Проверка ошибки авторизации

При добавлении администратором каких-либо данных, например, информации о фильме, также проводится контроль заполнения всех обязательных полей.

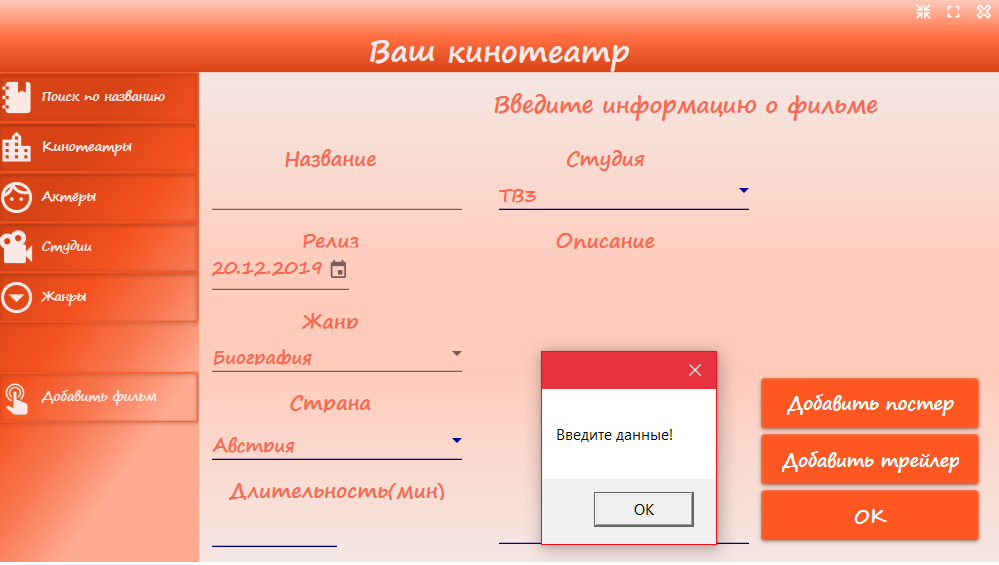


Рисунок 6.5 – Проверка заполнения полей при добавлении нового объекта

Если пользователь с правами администратора хочет купить билет, то ему сообщают, что он не может купить билет.

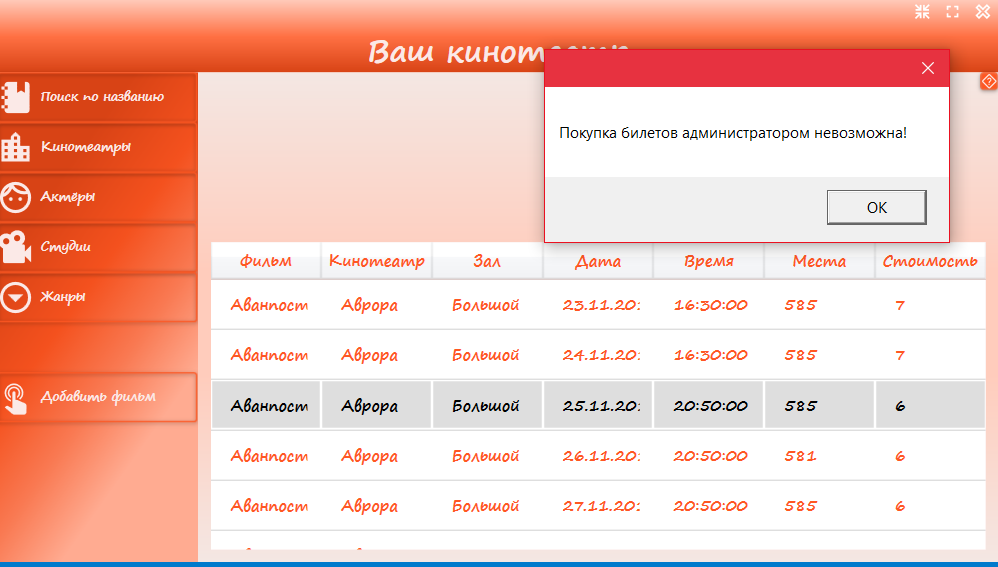


Рисунок 6.6 – Проверка заполнения полей при добавлении нового объекта

# Руководство пользователя

ФИО

Подпись

Дата

Лист

25

КП 07.00.ПЗ

Разраб.

*Хадасевич Ю. А*

Провер.

Бондарчик Е.Н.

Н. контр.

.

Утверд.

7 Руководство пользователя

Лит.

Листов

6

*БГТУ 71191005, 2019*

Первое, что должен сделать пользователь после открытия приложения — это войти, если он не зарегистрирован, то он может перейти соответственно в окно регистрации и сначала зарегистрироваться.

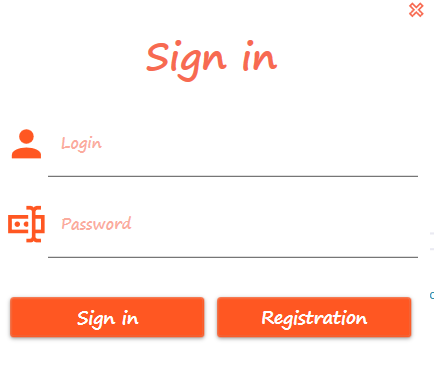
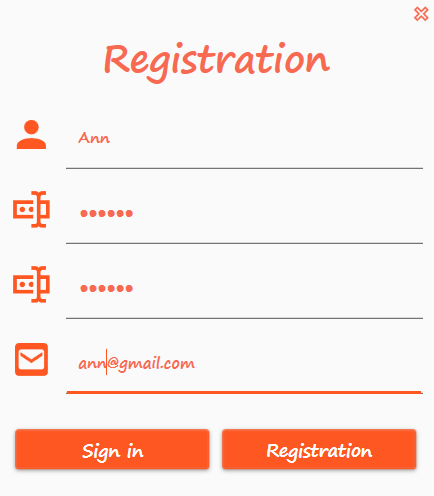
 

Рисунок 7.1 – Окна регистрации или авторизации

После авторизации пользователь попадает в окно поиска фильмов, где можно увидеть все фильмы, которые находятся в базе данных и краткую информацию о них. Также, введя в строку поиска часть названия или описания фильма, можно найти желаемый фильм с такими данными.

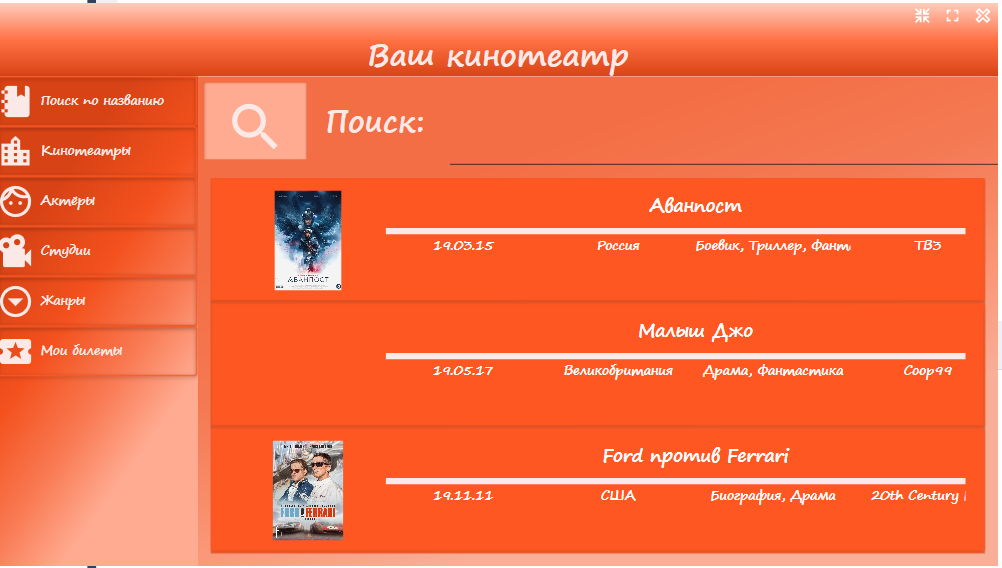


Рисунок 7.2 – Окно поиска фильмов по названию

Для того чтобы выбрать фильм нужно на него нажать. После этого откроется окно с подробной информацией о фильме, где кроме прочего можно посмотреть трейлер, нажав на соответствующую кнопку.

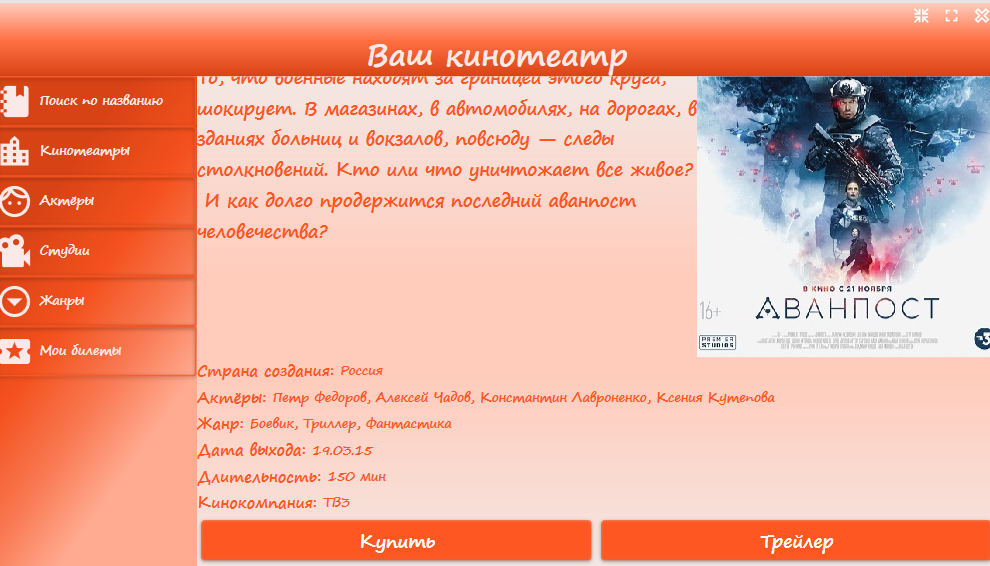


Рисунок 7.3 – Окно с выбранным фильмом

При нажатии кнопки «Купить» будет открыто окно со списком сеансов этого фильма и информацией об этих сеансах. Информацию можно отсортировать, нажав на заголовок столбца.



Рисунок 7.4 – Окно с сеансами на фильм

На выбранный сеанс нужно кликнуть два раза после чего появится элемент для выбора мест. За один раз можно приобрести пять билетов. Чтобы появился список мест нужно сначала установить желаемый ряд, потом два раза кликнуть на список с местами и после этого выбрать место. После выбора всех мест нажать «Ок».

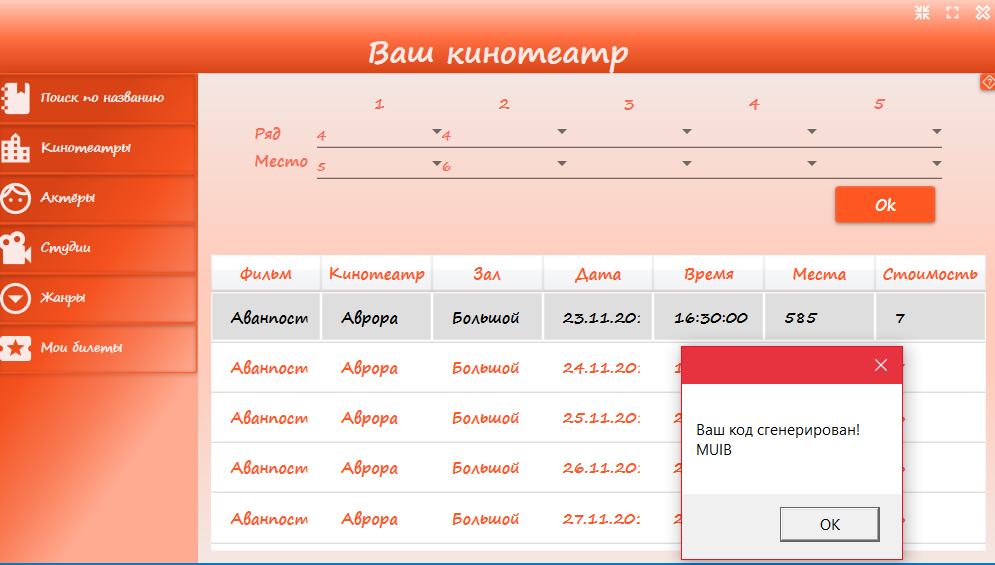


Рисунок 7.5 – Совершение покупки

Также можно совершить поиск фильма по жанрам. Чтобы это произошло нужно выбрать жанр из выпадающего меню. Если поиск совершается только по одному жанру, то он должен быть вписан в первое поле для выбора жанра. Для поиска фильма можно выбрать несколько жанров.

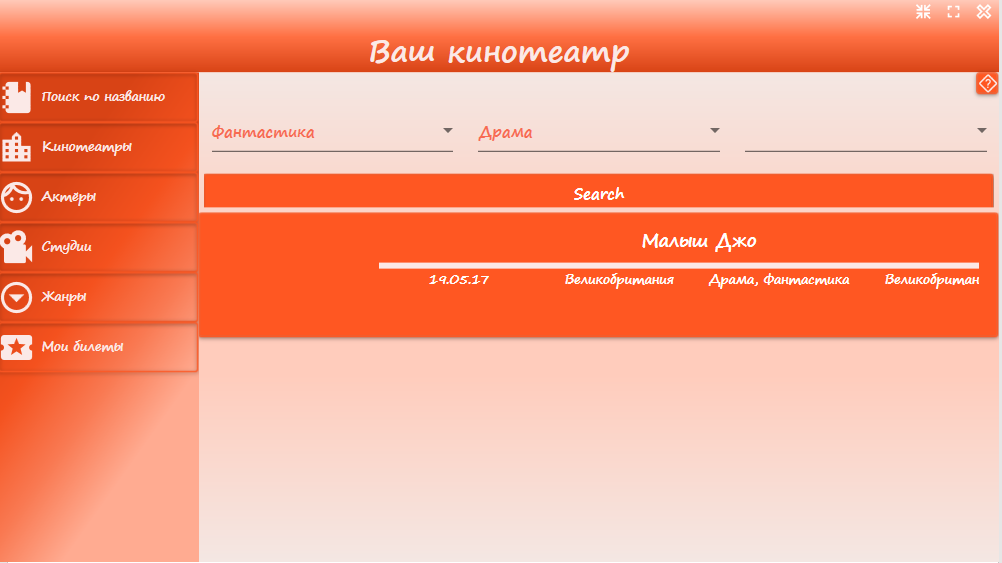


Рисунок 7.6 – Поиск фильма по жанрам

При нажатии соответствующих кнопок можно посмотреть информацию об актёрах, киностудиях и кинотеатрах. При двойном нажатии на кинотеатр можно увидеть все сеансы, которые идут в этом кинотеатре и приобрести билет также, как и ранее.

На вкладке «Мои билеты». Пользователь может увидеть свои билеты, сохранить из в XML-файл двойным нажатием на билете или удалить билет, нажав на него правой кнопкой мыши. Также можно добавить билет из XML-файла или сменить пользователя.

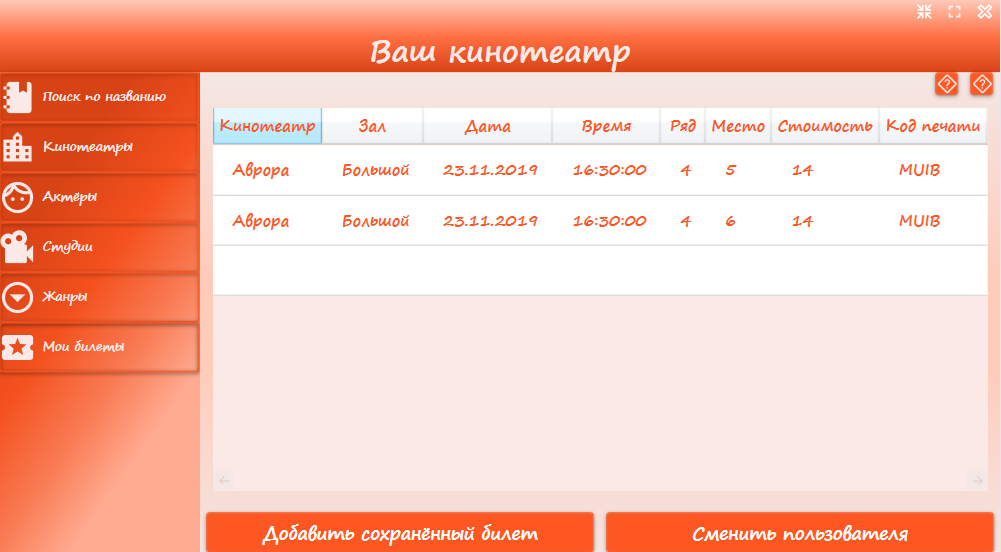


Рисунок 7.7 – Страница пользователя

Если пользователь вошёл как администратор, то он может осуществлять дополнительные операции. Кнопки для работы с кинотеатрами появляются на вкладке «Кинотеатры».



Рисунок 7.8 – Страница кинотеатров у администратора

Например, можно отредактировать кинотеатр, указав название кинотеатра, отметив галочками изменяемые поля, выбрав в них новое значение и нажав кнопку «Ок».

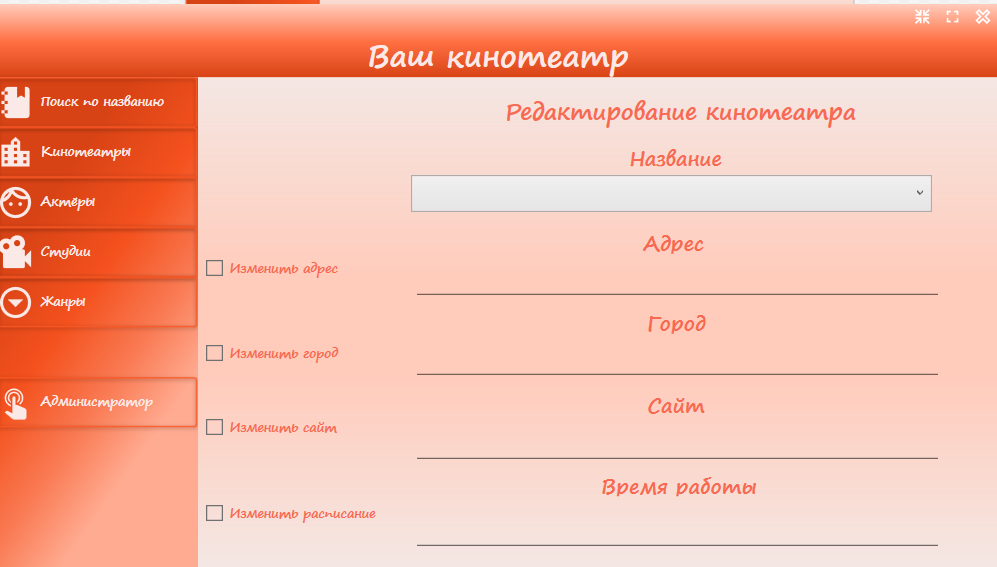


Рисунок 7.9 – Страница редактирования кинотеатра

Также у администратора есть вкладка «Администратор», где он может совершить дополнительные действия. Также администратор может удалить фильм, кинотеатр или сессию, нажав на ней правой кнопкой мыши.



Рисунок 7.10 – Страница администратора

Перечень всех действий пользователя можно увидеть на UML-диаграмме, которая осуществляет визуализацию разрабатываемой системы с разных точек зрения. Варианты использования отражают функциональность ПС с точки зрения получения значимого результата для пользователя. UML-диаграмма вариантов использования представлена в приложении В.

# 

# Заключение

ФИО

Подпись

Дата

Лист

31

КП 00.00.ПЗ

Разраб.

*Хадасевич Ю. А*

Провер.

Бондарчик Е.Н.

Н. контр.

.

Утверд.

Заключение

Лит.

Листов

1

*БГТУ 71191005, 2019*

Перед началом разработки базы данных был произведен аналитический обзор прототипов приложений подобной тематики и определение функциональных возможностей моей базы данных.

В процессе выполнения курсовой работы была спроектирована база данных для сети кинотеатров. База данных была разработана с помощью системы управления базами данных «Microsoft SQL Server 2012». Помимо этого, для хранения мультимедийной информации я ознакомилась с использованием BLOB файлов и таким типом данных, как varbinary.

При разработке базы данных было созданы следующие объекты:

– 13 таблиц с внешними связями;

– 42 хранимых процедур;

– 1 индекс;

– 2 пользователя базы данных.

Также было разработано приложение, демонстрирующее работу с базой данных. Приложение написано с помощью технологии WPF и языка программирования C#. Связь между базой данных и приложением осуществляется с помощью технологии ADO.NET, которая предоставляет набор классов, через которые мы можем установить подключение к базе данных и производить ряд других операций. Также было написано руководство пользователя для созданного приложения.

В соответствии с полученным результатом, можно сказать, что разработанная база данных функционирует верно, требования технического задания реализованы в полном объеме, в созданном приложении предоставлены возможности базы данных, поэтому цель курсового проекта можно считать достигнутой.

**Список литературных источников**

ФИО

Подпись

Дата

Лист

32

КП 00.00.ПЗ

Разраб.

*Хадасевич Ю. А*

Провер.

Бондарчик Е.Н.

Н. контр.

.

Утверд.

Список литературных источников

Лит.

Листов

1

*БГТУ 71191005, 2019*

1. Официальный сайт «ByCard» [Электронный ресурс] – Режим доступа: <http://www.bycard.by/>
2. Официальный сайт «TUT.by» раздел Афиша [Электронный ресурс] – Режим доступа: <https://afisha.tut.by/>
3. Официальный сайт «КиноМинска» ОАО [Электронный ресурс] – Режим доступа: <http://kinominska.by/>
4. Документация СУБД «Microsoft SQL Server» [Электронный ресурс] – Режим доступа: <https://docs.microsoft.com/en-us/sql/?view=sql-server-ver15>
5. Мэтью Макдональд — WPF: Windows Presentation Foundation в .NET 4.0 с примерами на C#
6. Язык C# и .NET Framework [Электронный ресурс] — Режим доступа: <http://professorweb.ru/my/csharp/charp_theory/level1/>
7. Руководство по ADO.NET и работе с базами данных [Электронный ресурс] – Режим доступа: <https://metanit.com/sharp/adonet/>
8. FileStream – SQL Server [Электронный ресурс] – Режим доступа: <https://docs.microsoft.com/en-us/sql/relational-databases/blob/filestream-sql-server?view=sql-server-ver15>
9. Хранение больших объектов в базе данных или файловой системе в SQL Server [Электронный ресурс] – Режим доступа: <https://www.microsoft.com/en-us/research/publication/to-blob-or-not-to-blob-large-object-storage-in-a-database-or-a-filesystem/>

# Приложение А

# Приложение Б

Листинг создания таблиц

--Country

create table COUNTRY(

ID nvarchar(10) primary key not null,

NAME nvarchar(70) unique not null);

--Studio

create table STUDIO(

ID uniqueidentifier primary key rowguidcol DEFAULT NEWSEQUENTIALID(),

NAME nvarchar(20) not null,

COUNTRY\_ID nvarchar(10) foreign key references COUNTRY(ID),

YEAR\_OF\_FOUNDATION int,

IMAGE varbinary(max));

--Genre

create table GENRE(

ID int primary key IDENTITY(1, 1),

NAME nvarchar(30) unique not null);

--Actor

create table ACTOR(

ID uniqueidentifier primary key rowguidcol DEFAULT NEWSEQUENTIALID(),

NAME nvarchar(20) not null,

SURNAME nvarchar(30) not null,

COUNTRY\_ID nvarchar(10) foreign key references Country(ID)

);

--Movie

create table MOVIE(

ID uniqueidentifier rowguidcol DEFAULT NEWSEQUENTIALID(),

NAME nvarchar(max) not null,

RELEASE date,

COUNTRY\_ID nvarchar(10) foreign key references COUNTRY(ID),

RUNNING\_TIME int,

STUDIO\_ID uniqueidentifier foreign key references STUDIO(ID),

PLOT nvarchar(max),

IMAGE varbinary(max),

CONSTRAINT PK\_Movie PRIMARY KEY CLUSTERED(ID));

alter table MOVIE ADD TRAILER varbinary(max) NULL

--Genre AND Movie (many-to-many)

CREATE TABLE GENRE\_MOVIE (

GENRE\_ID INT NOT NULL,

MOVIE\_ID uniqueidentifier NOT NULL,

PRIMARY KEY (GENRE\_ID, MOVIE\_ID),

CONSTRAINT FK\_MOVIE\_GENRE FOREIGN KEY (MOVIE\_ID)

REFERENCES MOVIE (ID) ON DELETE CASCADE,

CONSTRAINT FK\_GENRE FOREIGN KEY (GENRE\_ID)

REFERENCES GENRE (ID) ON DELETE CASCADE

)

--Actor AND Movie (many-to-many)

CREATE TABLE ACTOR\_MOVIE (

ACTOR\_ID uniqueidentifier NOT NULL,

MOVIE\_ID uniqueidentifier NOT NULL,

PRIMARY KEY (ACTOR\_ID, MOVIE\_ID),

CONSTRAINT FK\_MOVIE\_ACTOR FOREIGN KEY (MOVIE\_ID)

REFERENCES MOVIE (ID) ON DELETE CASCADE,

CONSTRAINT FK\_ACTOR FOREIGN KEY (ACTOR\_ID)

REFERENCES ACTOR (ID) ON DELETE CASCADE

)

--Cinema

create table CINEMA(

ID int primary key IDENTITY(1, 1),

NAME nvarchar(30) not null,

ADDRESS nvarchar(100),

WEBSITE nvarchar(100),

CITY nvarchar(30),

NUMBER\_OF\_HALLS int,

TIMETABLE nvarchar(200));

--Hall

create table HALL(

ID int primary key IDENTITY(1, 1),

NAME NVARCHAR(30),

CINEMA\_ID int foreign key references CINEMA(ID),

ROWS int not null,

SEATS int not null);

--Session

create table SESSION(

ID int primary key IDENTITY(1, 1),

MOVIE\_ID uniqueidentifier foreign key references MOVIE(ID),

HALL\_ID int foreign key references HALL(ID),

DATE date not null,

TIME time(7) not null,

COST int not null,

FREESEATS int);

--Users

create table USERS(

ID uniqueidentifier primary key rowguidcol DEFAULT NEWSEQUENTIALID(),

LOGIN nvarchar(50) not null,

PASSWORD nvarchar(30) not null,

EMAIL nvarchar(50) not null);

alter table USERS

add constraint PK\_USERS primary key nonclustered (ID asc);

--Purchase

create table PURCHASE(

ID int primary key IDENTITY(1, 1),

USER\_ID uniqueidentifier foreign key references USERS(ID) not null,

DATE smalldatetime);

ALTER TABLE PURCHASE ADD UNICK\_TICKET nvarchar(20);

ALTER TABLE PURCHASE ADD PRICE int null;

ALTER TABLE PURCHASE

ADD FOREIGN KEY (USER\_ID) REFERENCES USERS (ID);

--Tickets

create table TICKETS(

ID int primary key IDENTITY(1, 1),

SESSION\_ID int foreign key references SESSION(ID),

PURCHASE\_ID int foreign key references PURCHASE(ID),

ROW int not null,

SEAT int not null);

Листинг создания логинов и пользователей

CREATE LOGIN ADMINISTRATOR\_LOG

WITH PASSWORD = 'P@$$w0rd',

DEFAULT\_DATABASE = [Cinema],

DEFAULT\_LANGUAGE = [русский];

USE Cinema;

CREATE USER ADMINISTRATOR

FOR LOGIN ADMINISTRATOR\_LOG

WITH DEFAULT\_SCHEMA = [dbo];

GRANT EXECUTE TO ADMINISTRATOR;

CREATE LOGIN CLIENT\_LOG

WITH PASSWORD = '1111',

DEFAULT\_DATABASE = [Cinema],

DEFAULT\_LANGUAGE = [русский];

CREATE USER CLIENT

FOR LOGIN CLIENT\_LOG

WITH DEFAULT\_SCHEMA = [dbo];

GRANT EXECUTE ON OBJECT::Authorisation TO CLIENT;

GRANT EXECUTE ON OBJECT::Registration TO CLIENT;

GRANT EXECUTE ON OBJECT::TextSearch TO CLIENT;

GRANT EXECUTE ON OBJECT::GetActorInfo TO CLIENT;

GRANT EXECUTE ON OBJECT::GetCinemaInfo TO CLIENT;

GRANT EXECUTE ON OBJECT::GetMovieInfo TO CLIENT;

GRANT EXECUTE ON OBJECT::GetSessionInfo TO CLIENT;

GRANT EXECUTE ON OBJECT::SelectFreeSeats TO CLIENT;

GRANT EXECUTE ON OBJECT::SelectPrice TO CLIENT;

GRANT EXECUTE ON OBJECT::GetGenres TO CLIENT;

GRANT EXECUTE ON OBJECT::UserInfo TO CLIENT;

GRANT EXECUTE ON OBJECT::GetTicketInfo TO CLIENT;

GRANT EXECUTE ON OBJECT::GetStudiosInfo TO CLIENT;

GRANT EXECUTE ON OBJECT::GetSetActors TO CLIENT;

GRANT EXECUTE ON OBJECT::GetSetGenres TO CLIENT;

GRANT EXECUTE ON OBJECT::AllowRows TO CLIENT;

GRANT EXECUTE ON OBJECT::AllowSeats TO CLIENT;

GRANT EXECUTE ON OBJECT::CheckSeats TO CLIENT;

GRANT EXECUTE ON OBJECT::ImportFromXML TO CLIENT;

GRANT EXECUTE ON OBJECT::ExportToXML TO CLIENT;

GRANT EXECUTE ON OBJECT::ExportIntoXML TO CLIENT;

GRANT EXECUTE ON OBJECT::ExportIntoXML TO CLIENT;

GRANT EXECUTE ON OBJECT::DeleteTicket TO CLIENT;

GRANT EXECUTE ON OBJECT::InsertPurchase TO CLIENT;

GRANT EXECUTE ON OBJECT::InsertTicket TO CLIENT;

Листинг процедуры CheckSeats

CREATE PROCEDURE CheckSeats

@row int,

@seat int,

@hall nvarchar(30),

@message nvarchar(50) = NULL output

AS BEGIN

IF @row>(SELECT HALL.ROWS FROM HALL WHERE NAME=@hall) or @row<0

set @message='В зале не существует такого ряда';

else IF @seat>(SELECT HALL.SEATS FROM HALL WHERE NAME=@hall) or @seat<0

set @message='В зале не существует такого места';

IF @row=any(SELECT T.ROW FROM TICKETS T INNER JOIN SESSION S ON S.ID=T.SESSION\_ID where SEAT=@seat)

and @seat=any(SELECT SEAT FROM TICKETS T INNER JOIN SESSION S ON S.ID=T.SESSION\_ID where T.ROW=@row)

set @message='Это место уже занято';

END;

Процедура импорта данных из XML-файла

CREATE PROCEDURE ImportFromXML

@path nvarchar(256),

@rc bit output

AS BEGIN

SET @rc = 0;

SET NOCOUNT ON;

SET XACT\_ABORT ON;

BEGIN TRAN

declare @result table(x xml);

declare @sql nvarchar(300);

set @sql = 'SELECT CAST(REPLACE(CAST(x AS VARCHAR(MAX)), ''encoding="utf-16"'', ''encoding="utf-8"'') AS XML)

FROM OPENROWSET(BULK '''+@path+''', SINGLE\_BLOB) AS T(x)';

INSERT INTO @result EXEC(@sql);

declare @xml XML;

set @xml = (SELECT TOP 1 x FROM @result);

declare @hall\_id int=(select id from HALL where NAME=(select C3.value('Hall[1]', 'nvarChar(30)') AS TICKET\_OFFICE

FROM @xml.nodes('Root/Ticket') AS T3(C3)));

declare @cinema\_id int=(select id from CINEMA where NAME=(select C3.value('Cinema[1]', 'nvarChar(30)') AS TICKET\_OFFICE

FROM @xml.nodes('Root/Ticket') AS T3(C3)));

declare @movie\_id uniqueidentifier=(select id from MOVIE where NAME=(select C3.value('Name[1]', 'nvarChar(max)') AS TICKET\_OFFICE

FROM @xml.nodes('Root/Ticket') AS T3(C3)));

declare @session\_id int=(select id from SESSION where MOVIE\_ID=@movie\_id AND HALL\_ID=@hall\_id AND

DATE=(select C3.value('Date[1]', 'date') AS TICKET\_OFFICE

FROM @xml.nodes('Root/Ticket') AS T3(C3)) and TIME=(select C3.value('Time[1]', 'time') AS TICKET\_OFFICE

FROM @xml.nodes('Root/Ticket') AS T3(C3)));

declare @purchase\_id int=(select id from PURCHASE where UNICK\_TICKET=(select C3.value('Unick[1]', 'nvarChar(20)') AS TICKET\_OFFICE

FROM @xml.nodes('Root/Ticket') AS T3(C3)));

INSERT INTO TICKETS(SESSION\_ID, PURCHASE\_ID, ROW, SEAT)

SELECT

@session\_id, @purchase\_id,

C3.value('ROW[1]', 'int') AS ROW,

C3.value('SEAT[1]', 'int') AS SEAT

FROM @xml.nodes('Root/Ticket') AS T3(C3);

SET @rc = 1;

COMMIT;

END;

Процедура экспорта данных в XML-файл

exec master.dbo.sp\_configure 'show advanced options', 1;

RECONFIGURE;

exec master.dbo.sp\_configure 'xp\_cmdshell', 1;

RECONFIGURE;

GO

CREATE PROCEDURE ExportToXML

@path nvarchar(256),

@log nvarchar(50),

@r int,

@s int,

@u nvarchar(20),

@rc bit output

AS BEGIN

set @rc = 0;

BEGIN TRANSACTION

declare @sql nvarchar(500);

SET @sql = 'bcp "Exec ExportIntoXML @login='+@log+', @row='+cast(@r as nvarchar(3))+

', @seat='+cast(@s as nvarchar(3))+', @unik='+@u+'" queryout "'+@path+'" -S .\SQLEXPRESS -E -dCinema -w -C1251 -r -T';

EXEC xp\_cmdshell @sql;

set @rc = 1;

COMMIT;

END;

Процедура получения полных данных для экспорта в XML-файл

CREATE PROCEDURE ExportIntoXML

@login nvarchar(50),

@row nvarchar(3),

@seat nvarchar(3),

@unik nvarchar(20)

AS BEGIN

SELECT M.NAME[Name], C.NAME[Cinema], H.NAME[Hall], S.DATE[Date], TIME[Time], ROW, SEAT, PRICE, UNICK\_TICKET[Unick]

FROM USERS U INNER JOIN

PURCHASE P ON U.ID=P.USER\_ID INNER JOIN

TICKETS T ON P.ID=T.PURCHASE\_ID INNER JOIN

SESSION S ON S.ID=T.SESSION\_ID INNER JOIN

MOVIE M ON M.ID = S.MOVIE\_ID INNER JOIN

HALL H ON H.ID = S.HALL\_ID INNER JOIN

CINEMA C ON C.ID = H.CINEMA\_ID

where U.Login=@login AND UNICK\_TICKET=@unik AND SEAT=cast(@seat as int) and ROW=cast(@row as int) FOR XML PATH('Ticket'), ROOT('Root');

END;

Процедура добавления изображения в базу данных

CREATE PROCEDURE InsertMovie2

@name nvarchar(max),

@plot nvarchar(max),

@path nvarchar(max),

@rc int output

AS BEGIN

INSERT INTO movie(Name, Plot, image)

SELECT

@name AS Name

, @plot AS plot

, \*

FROM OPENROWSET(BULK N'''+ @path +''', SINGLE\_BLOB) AS image;

end;

Процедура InsertMovie

CREATE PROCEDURE InsertMovie

@name nvarchar(max),

@release date,

@country nvarchar(70),

@genre nvarchar(30),

@runningtime int,

@studioname nvarchar(20),

@plot nvarchar(max),

@image varbinary(max)=null,

@video varbinary(max)=null,

@rc int output

AS BEGIN

SET @rc = 0;

DECLARE @country\_id varchar(10);

SET @country\_id = (SELECT ID FROM COUNTRY WHERE NAME = @country);

DECLARE @genre\_id int;

SET @genre\_id = (SELECT ID FROM GENRE WHERE NAME = @genre);

DECLARE @studio\_id uniqueidentifier;

SET @studio\_id = (SELECT ID FROM STUDIO WHERE NAME = @studioname);

DECLARE @movie\_id uniqueidentifier;

BEGIN

BEGIN TRAN

INSERT INTO MOVIE(NAME, RELEASE, COUNTRY\_ID, RUNNING\_TIME, STUDIO\_ID, PLOT, IMAGE, TRAILER)

VALUES (@name, @release, @country\_id, @runningtime, @studio\_id, @plot, @image, @video);

SET @movie\_id=(select id from movie where name=@name);

INSERT INTO GENRE\_MOVIE(GENRE\_ID, MOVIE\_ID)

VALUES (@genre\_id, @movie\_id);

SET @rc = 1;

COMMIT TRAN;

IF @@TRANCOUNT>0 ROLLBACK TRAN;

END

END;

# Приложение В