СОДЕРЖАНИЕ

[Введение 4](file:///C:\Users\U-19K\Desktop\Курсач\Курсовой%20Проект.docx#_heading=h.gjdgxs)

[1 Экономическая характеристика объекта 5](file:///C:\Users\U-19K\Desktop\Курсач\Курсовой%20Проект.docx#_heading=h.30j0zll)

[2 Постановка задачи 6](file:///C:\Users\U-19K\Desktop\Курсач\Курсовой%20Проект.docx#_heading=h.1fob9te)

[2.1 Экономическое содержание задачи 6](file:///C:\Users\U-19K\Desktop\Курсач\Курсовой%20Проект.docx#_heading=h.3znysh7)

[2.2 Входная информация 6](file:///C:\Users\U-19K\Desktop\Курсач\Курсовой%20Проект.docx#_heading=h.2et92p0)

[3 Технические и инструментальные средства для решения задачи 7](file:///C:\Users\U-19K\Desktop\Курсач\Курсовой%20Проект.docx#_heading=h.tyjcwt)

[3.1 Обоснование выбора языка программирования 7](file:///C:\Users\U-19K\Desktop\Курсач\Курсовой%20Проект.docx#_heading=h.3dy6vkm)

[3.2 Технические требования к компьютеру для решения задачи 9](file:///C:\Users\U-19K\Desktop\Курсач\Курсовой%20Проект.docx#_heading=h.1t3h5sf)

[4 Информационное обеспечение задачи 10](file:///C:\Users\U-19K\Desktop\Курсач\Курсовой%20Проект.docx#_heading=h.4d34og8)

[4.1 Классификаторы, используемые для решения задачи 10](file:///C:\Users\U-19K\Desktop\Курсач\Курсовой%20Проект.docx#_heading=h.2s8eyo1)

[5 Программное обеспечение задачи 13](file:///C:\Users\U-19K\Desktop\Курсач\Курсовой%20Проект.docx#_heading=h.17dp8vu)

[5.1 Описание методов 13](file:///C:\Users\U-19K\Desktop\Курсач\Курсовой%20Проект.docx#_heading=h.3rdcrjn)

[5.2 Алгоритм решения задачи 15](file:///C:\Users\U-19K\Desktop\Курсач\Курсовой%20Проект.docx#_heading=h.26in1rg)

[5.3 Описание блок – схемы 15](file:///C:\Users\U-19K\Desktop\Курсач\Курсовой%20Проект.docx#_heading=h.lnxbz9)

[Заключение 19](file:///C:\Users\U-19K\Desktop\Курсач\Курсовой%20Проект.docx#_heading=h.35nkun2)

[Список использованных источников 20](file:///C:\Users\U-19K\Desktop\Курсач\Курсовой%20Проект.docx#_heading=h.1ksv4uv)

[Приложение А 21](file:///C:\Users\U-19K\Desktop\Курсач\Курсовой%20Проект.docx#_heading=h.44sinio)

[Приложение Б 22](file:///C:\Users\U-19K\Desktop\Курсач\Курсовой%20Проект.docx#_heading=h.2jxsxqh)

# Введение

Кинотеатр — общественное здание или его часть с оборудованием для публичной демонстрации [кинофильмов](https://ru.wikipedia.org/wiki/%D0%9A%D0%B8%D0%BD%D0%B5%D0%BC%D0%B0%D1%82%D0%BE%D0%B3%D1%80%D0%B0%D1%84). Главное помещение кинотеатра — зрительный зал с экраном большого размера и системой воспроизведения звука. В современных кинотеатрах система звуковоспроизведения состоит из нескольких [громкоговорителей](https://ru.wikipedia.org/wiki/%D0%93%D1%80%D0%BE%D0%BC%D0%BA%D0%BE%D0%B3%D0%BE%D0%B2%D0%BE%D1%80%D0%B8%D1%82%D0%B5%D0%BB%D1%8C), обеспечивающих [объёмный звук](https://ru.wikipedia.org/wiki/%D0%9E%D0%B1%D1%8A%D1%91%D0%BC%D0%BD%D1%8B%D0%B9_%D0%B7%D0%B2%D1%83%D0%BA).

Первые стационарные кинотеатры назывались в [России](https://ru.wikipedia.org/wiki/%D0%A0%D0%BE%D1%81%D1%81%D0%B8%D1%8F) «электротеатрами».

В современных кинотеатрах часто имеется несколько зрительных залов, обязательна система [кондиционирования воздуха](https://ru.wikipedia.org/wiki/%D0%9A%D0%BE%D0%BD%D0%B4%D0%B8%D1%86%D0%B8%D0%BE%D0%BD%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5_%D0%B2%D0%BE%D0%B7%D0%B4%D1%83%D1%85%D0%B0), а современные звуковоспроизводящие системы состоят из множества раздельных звуковых каналов. При проектировании зданий для кинотеатров одной из главных задач считается достижение наилучших [акустических](https://ru.wikipedia.org/wiki/%D0%90%D1%80%D1%85%D0%B8%D1%82%D0%B5%D0%BA%D1%82%D1%83%D1%80%D0%BD%D0%B0%D1%8F_%D0%B0%D0%BA%D1%83%D1%81%D1%82%D0%B8%D0%BA%D0%B0) характеристик кинозала для качественного звучания. В кинотеатрах обычно также имеются фойе для зрителей, гардероб, буфет, служебные помещения. [Киноустановка](https://ru.wikipedia.org/wiki/%D0%9A%D0%B8%D0%BD%D0%BE%D1%83%D1%81%D1%82%D0%B0%D0%BD%D0%BE%D0%B2%D0%BA%D0%B0) кинотеатра, использующего [киноплёнку](https://ru.wikipedia.org/wiki/%D0%9A%D0%B8%D0%BD%D0%BE%D0%BF%D0%BB%D1%91%D0%BD%D0%BA%D0%B0), состоит не менее, чем из двух [кинопроекторов](https://ru.wikipedia.org/wiki/%D0%9A%D0%B8%D0%BD%D0%BE%D0%BF%D1%80%D0%BE%D0%B5%D0%BA%D1%82%D0%BE%D1%80) для обеспечения непрерывности демонстрации фильма. Каждый кинопроектор называется «[постом](https://ru.wikipedia.org/wiki/%D0%9F%D0%BE%D1%81%D1%82_%D0%BA%D0%B8%D0%BD%D0%BE%D0%BF%D1%80%D0%BE%D0%B5%D0%BA%D1%86%D0%B8%D0%B8)» и все имеющиеся посты централизованно управляются специальной автоматизированной системой кинопоказа, обеспечивающей незаметность переходов между отдельными [частями фильма](https://ru.wikipedia.org/wiki/%D0%A7%D0%B0%D1%81%D1%82%D1%8C_%D1%84%D0%B8%D0%BB%D1%8C%D0%BC%D0%B0) для зрителя.

Цель курсового проекта заключается в создании настольного приложения «Резервирование билетов» на базе ОС Windows.

В курсовом проекте решаются задачи:

-Формирование базы данных;

-Резервирование билетов в реальном времени;

# 1 Экономическая характеристика объекта

Кинотеатр «Комсомолец» в Волгодонск.

В кинотеатре имеется Малый зал (3D) на 180 мест и Большой (2D) на 480 мест, в котором установлен самый большой в городе экран. Кинозалы оснащены передовым проекционным оборудованием, многоканальными звуковыми системами. В кинозалах стильный интерьер, отличная акустика, установлены кондиционеры, удобные кресла расположены амфитеатром, на последних рядах установлены диванчики на двоих.  
В просторном фойе открыт кино-бар с попкорном, кофе, десертами, напитками.  
В кинотеатре «Комсомолец» демонстрируются фильмы в разных жанрах — от блокбастеров и детективов до семейного и детского кино.

# 2 Постановка задачи

2.1 Экономическое содержание задачи

Цель курсового проекта - создание оконного приложения «Резервирование билетов» на базе ОС Windows.

Разработанный модуль позволяет резервировать билеты в кинотеатре с помощью компьютера.

Программа создана на языке C#, графический интерфейс приложения дружественный и понятный, сама программа проста в использовании и не требует специального образования или прохождение курсов.

2.2 Входная информация и Выходная информация

Входная информация представляет собой список фильмов хранящиеся в базе данных. Выходной информацией является зарезервированное место.

# 3.Технические и инструментальные средства для решения задачи

3.1 Обоснование выбора языка программирования

C# — объектно-ориентированный, **ориентированный на компоненты** язык программирования. C# предоставляет языковые конструкции для непосредственной поддержки такой концепции работы. Благодаря этому C# подходит для создания и применения программных компонентов. С момента создания язык C# обогатился функциями для поддержки новых рабочих нагрузок и современными рекомендациями по разработке ПО. В основном C# — **объектно-ориентированный** язык. Вы определяете типы и их поведение.

Вот лишь несколько функций языка C#, которые позволяют создавать надежные и устойчивые приложения. [**Сборка мусора**](https://docs.microsoft.com/ru-ru/dotnet/standard/garbage-collection/) автоматически освобождает память, занятую недостижимыми неиспользуемыми объектами. [**Типы, допускающие значение null,**](https://docs.microsoft.com/ru-ru/dotnet/csharp/nullable-references) обеспечивают защиту от переменных, которые не ссылаются на выделенные объекты. [**Обработка исключений**](https://docs.microsoft.com/ru-ru/dotnet/csharp/fundamentals/exceptions/) предоставляет структурированный и расширяемый подход к обнаружению ошибок и восстановлению после них. [**Лямбда-выражения**](https://docs.microsoft.com/ru-ru/dotnet/csharp/language-reference/operators/lambda-expressions) поддерживают приемы функционального программирования. [**Синтаксис LINQ**](https://docs.microsoft.com/ru-ru/dotnet/csharp/linq/) создает общий шаблон для работы с данными из любого источника. Поддержка языков для [**асинхронных операций**](https://docs.microsoft.com/ru-ru/dotnet/csharp/programming-guide/concepts/async/) предоставляет синтаксис для создания распределенных систем. В C# имеется [**Единая система типов**](https://docs.microsoft.com/ru-ru/dotnet/csharp/fundamentals/types/). Все типы C#, включая типы-примитивы, такие как int и double, наследуют от одного корневого типа object. Все типы используют общий набор операций, а значения любого типа можно хранить, передавать и обрабатывать схожим образом. Более того, C# поддерживает как определяемые пользователями [ссылочные типы](https://docs.microsoft.com/ru-ru/dotnet/csharp/language-reference/builtin-types/reference-types), так и [типы значений](https://docs.microsoft.com/ru-ru/dotnet/csharp/language-reference/builtin-types/value-types). C# позволяет динамически выделять объекты и хранить упрощенные структуры в стеке. C# поддерживает универсальные методы и типы, обеспечивающие повышенную безопасность типов и производительность. C# предоставляет итераторы, которые позволяют разработчикам классов коллекций определять пользовательские варианты поведения для клиентского кода.

C# подчеркивает **Управление версиями,** чтобы обеспечить совместимость программ и библиотек с течением времени. Вопросы управления версиями существенно повлияли на такие аспекты разработки C#, как раздельные модификаторы virtual и override, правила разрешения перегрузки методов и поддержка явного объявления членов интерфейса.

Плюсы:

* Поддержка Microsoft. В отличии от Java, которой не пошел на пользу переход в собственность Oracle, C# хорошо развивается благодаря усилиям Microsoft.
* В последнее время много совершенствуется. Так как C# был создан позже, чем Java и другие языки, то требовалось очень много доработать. Также это касается популяризации и бесплатности - было обещано открыть исходный код, а инструменты (Visual Studio, Xamarin) стали бесплатными для частных лиц и небольших компаний.
* Много синтаксического сахара. Синтаксический сахар - это такие конструкции, которые созданы для облегчения написания и понимания кода (особенно если это код другого программиста) и не играют роли при компиляции.
* Средний порог вхождения. Синтаксис похожий на C, С++ или Java облегчает переход для других программистов. Для новичков это также один из самых перспективных языков для изучения.
* Xamarin. Благодаря покупке Xamarin на C# теперь можно писать под Android и iOS. Это, несомненно, большой плюс, так как их собственная мобильная ОС (Windows Phone) не завоевала большой популярности.
* Добавлено функциональное программирование (F#).
* Большое сообщество программистов.
* Много вакансий на должность C# программиста в любом регионе.

Минусы:

* Бесплатность только для небольших компании, учащихся и программистов-одиночек. Для больших команд покупка лицензий обойдется недешево. Поэтому если у вас есть своя фирма, то придется раскошелиться.
* Сохранили оператор go to.

3.2 Технические требования к компьютеру на которой будет установлена Visual Studio

* Операционная система – Microsoft® Windows® 7/8/10 (32- или 64-bit);
* оперативная память – 3 ГБ минимум, рекомендуется 8 ГБ;
* пространство на жестком диске – рекомендуется 4 ГБ;
* монитор – 1280 х 800 минимальное разрешение экрана.

# 4 Информационное обеспечения задачи

4.1 Классификаторы, используемые для решения задачи

Классификатор представляет собой набор таблиц (справочников).

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 работнике приведенная в таблице 4.1, документе показанных на таблице 4.2 и местонахождении документа на таблице 4.3.

Таблица 4.1 – Атрибуты Зал

|  |  |
| --- | --- |
| Объект | Атрибуты |
| Зал | Идентификационный код  Название  Местоположение  Количество мест |

Таблица 4.2 – Атрибуты Кино

|  |  |
| --- | --- |
| Объект | Атрибуты |
| Кино | Идентификационный код  Название фильма  Описание  Жанр  Длительность  Режиссер  Начало |

Таблица 4.3 – Атрибуты Билет

|  |  |
| --- | --- |
| Объект | Атрибуты |
| Билет | Идентификационный код  Идентификационный код сеанса  Идентификационный код Места |

Таблица 4.4 – Атрибуты Сеанс

|  |  |
| --- | --- |
| Объект | Атрибуты |
| Сеанс | Идентификационный код  Идентификационный код Кино  Идентификационный код Зала |

Таблица 4.5 – Атрибуты Тип места

|  |  |
| --- | --- |
| Объект | Атрибуты |
| Тип места | Идентификационный код  Идентификационный код Места  Название |

Таблица 4.6 – Атрибуты Место

|  |  |
| --- | --- |
| Объект | Атрибуты |
| Место | Идентификационный код  Идентификационный код Зала |

Для выполнения проекта была выбрана СУБД SQL Server. Схема БД представлена на рисунке 4.1.

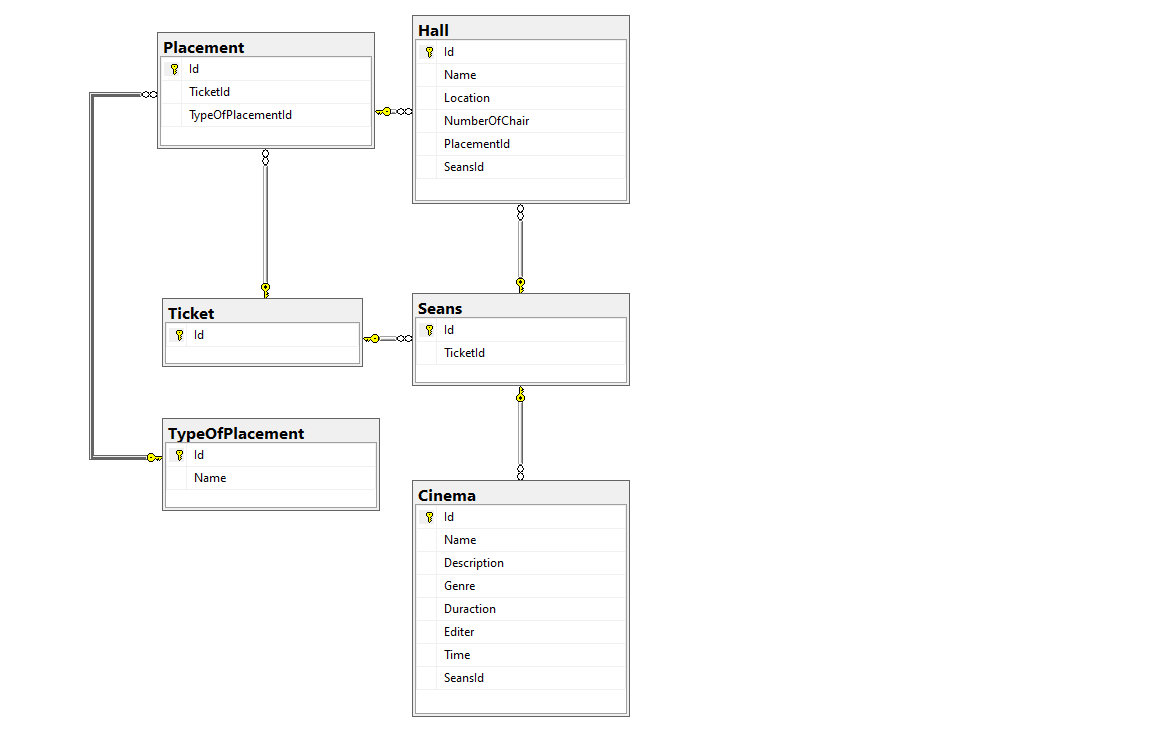


Рисунок 4.1 – Связи таблиц в БД

Структура таблиц Билет, место, зал, сеанс, фильм и тип места базы данных представлена на рисунках 4.2 – 4.7 соответственно.

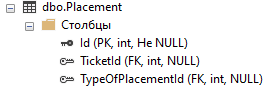


Рисунок 4.2 – Таблица «Место»

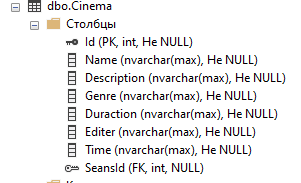


Рисунок 4.3 – Таблица «Кино»



Рисунок 4.4 – Таблица «Билет»

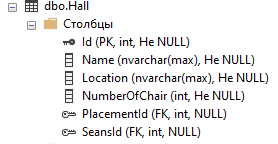


Рисунок 4.5 – Таблица «Зал»

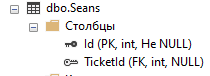


Рисунок 4.6 – Таблица «Сеанс»

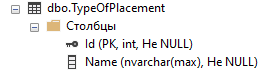


Рисунок 4.7 – Таблица «Тип места»

# 5 Программное обеспечение задачи

5.1 Описание методов

Для примера взята часть кода основного окна после входа в учетную область.

<Window x:Class="KontrolWork.MainWindow"

xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"

xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"

xmlns:d="http://schemas.microsoft.com/expression/blend/2008"

xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"

xmlns:local="clr-namespace:KontrolWork"

mc:Ignorable="d"

Title="MainWindow" Height="450" Width="800">

<Grid Background="AliceBlue">

<Grid.ColumnDefinitions >

<ColumnDefinition Width="\*">

</ColumnDefinition>

<ColumnDefinition>

</ColumnDefinition>

</Grid.ColumnDefinitions>

<Grid.RowDefinitions>

<RowDefinition>

</RowDefinition>

</Grid.RowDefinitions>

<StackPanel Grid.Column="0" Grid.Row="0" Name="Stack">

<TextBlock Text="Сеанс" FontSize="35" HorizontalAlignment="Center" ></TextBlock>

<Button Name="First" Content="Титаник" Height="66" FontSize="30" Click="First\_Click" Background="Bisque"/>

<Button Name="Second" Content="Брат 2" Height="66" FontSize="30" Click="Second\_Click" Background="Bisque"/>

<Button Name="Third" Content="Люди в черном" FontSize="30" Height="66" Background="Bisque"/>

</StackPanel>

<!-- <Button Height="50" Grid.Column="1" HorizontalAlignment="Center" VerticalAlignment="Center" Width="300" Content="Авторизация" FontSize="30" Click="Button\_Click" Background="Bisque"/> -->

<Button Height="50" Grid.Column="1" HorizontalAlignment="Center" VerticalAlignment="Center" Width="300" Content="Добавить фильм" FontSize="30" Click="Button\_Click\_1" Visibility="Visible" Name="AddFilm" Background="Bisque"/>

</Grid>

</Window>

5.2 Алгоритм решения задачи

Для проведения каких-либо действий с резервированием билетов сначала нужно добавить фильм в программу после чего зарезервировать место в выбранном зале.

5.3 Описание блок – схемы

Последовательное описание блок – схемы приложения приведено ниже:

Шаг 1. Запуск программы.

Шаг 2. Вопрос: Верно ли указан логин и пароль? Если да, то переходим к следующему шагу, иначе остаемся на 2 шаге.

Шаг 3. Вход в приложение под учетной записью

Шаг 4. Вопрос: Выбрано ли действие? Если да, то переходим к 5 шагу, если нет, то переходим к 7 шагу.

Шаг 5. Выполнение

Шаг 6. Возвращаемся к 4 шагу.

Шаг 7. Закрытие приложения.

Блок-схема Приведена в приложении А.

Листинг программы приведен в приложении Б.

5.4 Описание интерфейса программы

В активити входа в приложение, представленном на рисунке 5.1, находится два текстовых поля логин и пароль, кнопка авторизации. Для дальнейшей работы нужно верно указать логин и пароль. Если логин и пароль верны, то откроется основное активити, представленное на рисунке 5.2 с возможностью выбора фильма. При нажатии на кнопку открывается окно с выбором места в конкретном зале которое показано на рисунке 5.3

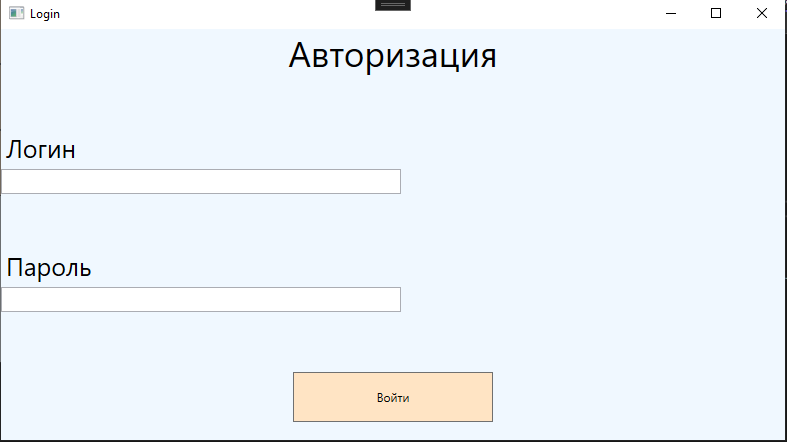


Рисунок 5.1 – Активити входа

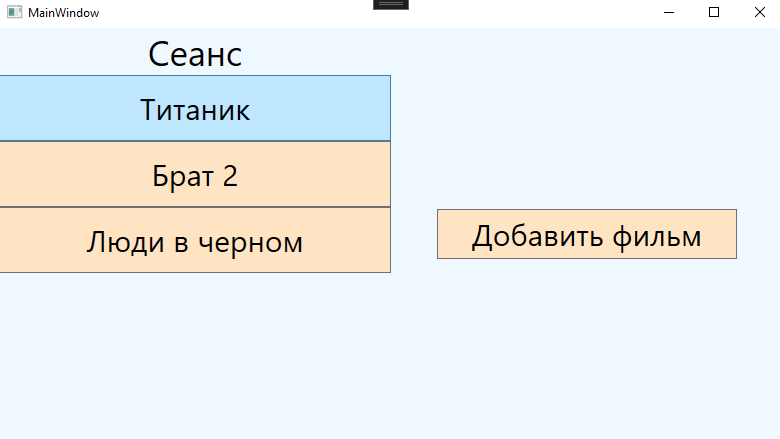


Рисунок 5.2 – Основное активити

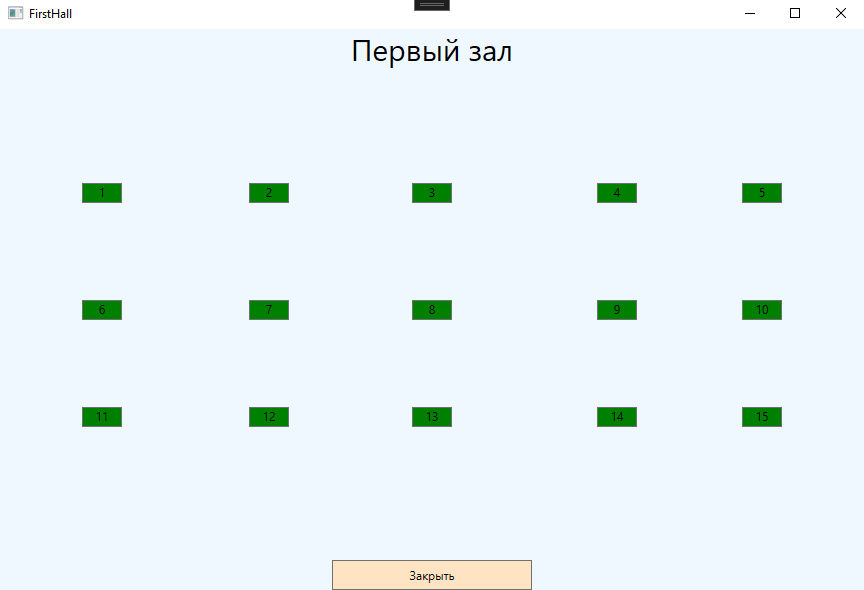


Рисунок 5.3 – Активити выбора места

Для успешной работы с программой необходимо:

* 1. Версия Windows 10.
  2. Вход в учетную запись

# Заключение

Программа разработана на языке C# в интегрированной середе разработки Visual Studio с использованием СУБД SQL Server. Приложение интуитивно понятно даже неопытному пользователю и не затрудняет его работу с документами с помощью компьютера.

В дальнейшем приложение будет обновлятся и будут добавлятся новые функции необходимые для работы с резервированием билетов.

В процессе написания кода для данной программы были приобретены практические навыки в работе в СУБД SQL Server и языке программирования C#.

Цель курсового проекта достигнута. Задачи выполнены.

# Список использованных источников

Интернет ресурсы:

1 Википедия. Язык программирование СИ [Электронный ресурс] – режим доступа: https://ru.wikipedia.org/wiki/Си\_(язык\_программирования)

2 Microsoft. Документация по C#. [Электронный ресурс] – режим доступа: https://docs.microsoft.com/ru-ru/dotnet/csharp/

3 Stackoverflow. Ответы на вопросы программирования [Электронный ресурс] – режим доступа https://ru.stackoverflow.com/

4 Metanit. Язык программирования C# и платформа .NET [Электронный ресурс] – режим доступа https://metanit.com/sharp/tutorial/1.1.php

5 Habr. Российский блог информационных технологий [Электронный ресурс] – Режим доступа: https://habr.com/ru/

# Приложение А

(обязательное)

Блок – схема программы

Запуск программы

Начало

Введены верные логин и пароль?

Да

Вход в приложение

Нет

Выбрано действие?

Нет

Закрытие

Закрепление билета

Выполнение

Да

Рисунок А.1–Блок схема программы

# Приложение Б

(обязательное)

Листинг программы

using KontrolWork;

using Microsoft.EntityFrameworkCore;

using Microsoft.EntityFrameworkCore.Infrastructure;

using Microsoft.EntityFrameworkCore.Metadata;

using Microsoft.EntityFrameworkCore.Storage.ValueConversion;

#nullable disable

namespace KontrolWork.Migrations

{

[DbContext(typeof(DatabaseContext))]

partial class DatabaseContextModelSnapshot : ModelSnapshot

{

protected override void BuildModel(ModelBuilder modelBuilder)

{

#pragma warning disable 612, 618

modelBuilder

.HasAnnotation("ProductVersion", "6.0.4")

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

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

modelBuilder.Entity("KontrolWork.Chair", b =>

{

b.Property<int>("Id")

.ValueGeneratedOnAdd()

.HasColumnType("int");

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

b.Property<int>("Hall")

.HasColumnType("int");

b.Property<int>("Number")

.HasColumnType("int");

b.HasKey("Id");

b.ToTable("Chair");

});

modelBuilder.Entity("KontrolWork.Models.Cinema", b =>

{

b.Property<int>("Id")

.ValueGeneratedOnAdd()

.HasColumnType("int");

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

b.Property<string>("CinemaName")

.IsRequired()

.HasColumnType("nvarchar(max)");

b.Property<string>("Time")

.IsRequired()

.HasColumnType("nvarchar(max)");

b.HasKey("Id");

b.ToTable("Cinema");

});

modelBuilder.Entity("KontrolWork.Tickets", b =>

{

b.Property<int>("Id")

.ValueGeneratedOnAdd()

.HasColumnType("int");

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

b.Property<int>("ChairId")

.HasColumnType("int");

b.Property<int>("CinemaId")

.HasColumnType("int");

b.HasKey("Id");

b.HasIndex("ChairId");

b.HasIndex("CinemaId");

b.ToTable("Ticket");

});

modelBuilder.Entity("KontrolWork.Tickets", b =>

{

b.HasOne("KontrolWork.Chair", "Chair")

.WithMany()

.HasForeignKey("ChairId")

.OnDelete(DeleteBehavior.Cascade)

.IsRequired();

b.HasOne("KontrolWork.Models.Cinema", "Cinema")

.WithMany()

.HasForeignKey("CinemaId")

.OnDelete(DeleteBehavior.Cascade)

.IsRequired();

b.Navigation("Chair");

b.Navigation("Cinema");

});

#pragma warning restore 612, 618

}

}

}

public partial class film : Migration

{

protected override void Up(MigrationBuilder migrationBuilder)

{

migrationBuilder.CreateTable(

name: "Chair",

columns: table => new

{

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

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

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

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

},

constraints: table =>

{

table.PrimaryKey("PK\_Chair", x => x.Id);

});

migrationBuilder.CreateTable(

name: "Cinema",

columns: table => new

{

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

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

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

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

},

constraints: table =>

{

table.PrimaryKey("PK\_Cinema", x => x.Id);

});

migrationBuilder.CreateTable(

name: "Ticket",

columns: table => new

{

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

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

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

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

},

constraints: table =>

{

table.PrimaryKey("PK\_Ticket", x => x.Id);

table.ForeignKey(

name: "FK\_Ticket\_Chair\_ChairId",

column: x => x.ChairId,

principalTable: "Chair",

principalColumn: "Id",

onDelete: ReferentialAction.Cascade);

table.ForeignKey(

name: "FK\_Ticket\_Cinema\_CinemaId",

column: x => x.CinemaId,

principalTable: "Cinema",

principalColumn: "Id",

onDelete: ReferentialAction.Cascade);

});

migrationBuilder.CreateIndex(

name: "IX\_Ticket\_ChairId",

table: "Ticket",

column: "ChairId");

migrationBuilder.CreateIndex(

name: "IX\_Ticket\_CinemaId",

table: "Ticket",

column: "CinemaId");

}

protected override void Down(MigrationBuilder migrationBuilder)

{

migrationBuilder.DropTable(

name: "Ticket");

migrationBuilder.DropTable(

name: "Chair");

migrationBuilder.DropTable(

name: "Cinema");

}

}