МИНИСТЕРСТВО ВЫСШЕГО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ ФЕДЕРАЛЬНОЕ

ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

«РЯЗАНСКИЙ ГОСУДАРСТВЕННЫЙ РАДИОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ ИМЕНИ В.Ф. УТКИНА»

Факультет вычислительной техники

Кафедра электронных вычислительных машин

К защите

Руководитель проекта

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**ПОЯСНИТЕЛЬНАЯ ЗАПИСКА**

**К КУРСОВОМУ ПРОЕКТУ**

по дисциплине «Промышленное программирование»

по теме «Разработка информационной системы с клиент-серверной архитектурой»

Выполнил:

студент группы 845

Кулаков Д. А. \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

дата сдачи на проверку,

подпись

Руководитель проекта:

ассистент кафедры ЭВМ

Хизриева Н.И. \_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

оценка дата защиты, подпись

Рязань, 2021 г

**Содержание**

ВВЕДЕНИЕ………………………………………………………………………3

1 Концептуальное описание предметной области……………………….……4

2 Анализ существующих решений для предметной области………………...5

3 Разработка клиентского приложения

3.1 Выбор программных компонентов клиентской части…………………....6

3.2 Разработка интерфейса пользователя……………………………………...7

4 Разработка программной документации…………………………………….9

ЗАКЛЮЧЕНИЕ………………………………………………………………...10

СПИСОК ИСПОЛЬЗУЕМОЙ ЛИТЕРАТУРЫ………………………………11

ПРИЛОЖЕНИЕ А: исходный текст клиентской программы……………….12

**Введение**

В современном мире с повсеместном развитием способов получения информации для людей острым встает вопрос затраты времени. Поэтому разрабатываются системы, позволяющие быстро получить необходимую информацию. Сейчас существуют крупные системы, обладающие большой скоростью, значительным объемом поисковых данных и их доступностью.

В данной курсовой работе, призванной ознакомить студентов с проектированием подобной системы, но куда более меньших размеров, была разработана система для кинотеатра, которая позволяет управлять основными объектами. Использование веб-приложения для кинотеатра позволяет узнавать о расписаниях и покупать билеты фильмы там, где удобно зрителю. То есть нет необходимости в посещении кинотеатра, кроме как для просмотра киноленты.

База данных разработана в среде Microsoft SQL Server 2018. Клиентское приложение (сайт) разработано в среде Microsoft Visual Studio. Использованные языки: Transact-SQL, C#, JS.

1. **Концептуальное описание предметной области**

Предметная область – кинотеатр, предоставляющий прокат различных фильмов.

Кинотеатр должен обладать определенным административным и оперативным штатом, в который входят: администраторы, менеджеры, киномеханики, кассиры (работники у залов), уборщики.

Основная деятельность – показ фильмов, поэтому у кинотеатра должна быть база кинолент, со временем меняющая свой состав. В состав так же входят кинозалы, кассы.

Кинотеатр имеет возможность предоставление проката реклам перед сеансами.

Покупка билетов может осуществляться через кассы, либо через интернет-сайт.

1. **Анализ существующих решений для предметной области**

Для курсовой предметной области уже существует множество решений, позволяющих получить обширный объем информации не только о фильме, но и дополнительную информацию, такую как рецензии, события, связанные с фильмом, сборы и другие.

Подобные системы сходны по функционалу, поэтому для рассмотрения был взят портал «КиноПоиск».

На сайте в список расположены фильмы, находящиеся в прокате. Для каждого фильма выводится название, изображение, режиссер и жанр. Также рейтинг, дата премьеры и кнопка для перехода к покупке билетов. Подобное представление позволяет пользователю быстро оценить текущую афишу и определиться с выбором.

Если этой информации недостаточно, то пользователь может перейти к расширенной информации о фильме.

Когда пользователь определится с выбором, то он может перейти к сеансам, выбрать подходящий, далее забронировать места.

На портале доступно большое количество функций, однако для курсового проекта выбор остановился на описанном выше наборе.

1. **Разработка клиентского приложения**

**3.1 Выбор программных компонентов клиентской части**

Клиентское приложение для данной курсовой работы было разработано в среде разработки Microsoft Visual Studio на языке С# с использованием фреймворков ASP.NET и ADO.NET.

Фреймворк ASP.NET обладает несколькими модулями для разработки веб-приложений. В работе был использован ASP.NET MVC, который предоставляет создавать веб-страницы с использованием шаблона проектирования MVC.

Платформа ASP.NET MVC базируется на взаимодействии трех компонентов: контроллера, модели и представления.

Контроллер принимает запросы, обрабатывает пользовательский ввод, взаимодействует с моделью и представлением и возвращает пользователю результат обработки запроса.

Модель представляет слой, описывающий логику организации данных в приложении. В модель могут быть включены несколько экземпляров разных классов, что позволяет использовать лишь одну модель для получения данных.

Представление получает данные из контроллера и генерирует элементы пользовательского интерфейса для отображения информации. Для этого применяется смесь языка C# и HTML – cshtml. Это позволяет получать и обрабатывать данные посредством как языка C#, так и JavaScript, а также отрисовывать страницы с помощью CSS.

Фреймворк ADO.NET - технология, предоставляющая доступ и управление данным, хранящимся в базе данных. В данной курсовой используется доступ к Microsoft SQL Server.

**3.2 Разработка интерфейса пользователя**

Для данной курсовой работы был разработан веб-сайт на фреймворке ASP.NET с использованием языков C#, JS, HTML, а также CSS.

Подключение к базе данных производится автоматически при запуске сайта.

После запуска сайта пользователю выводится информация о текущих фильмах в кинотеатре: изображение, название, режиссер и кнопка для перехода к сеансам на соответственный фильм. Стартовая страница приведена на рисунке 6.2.1

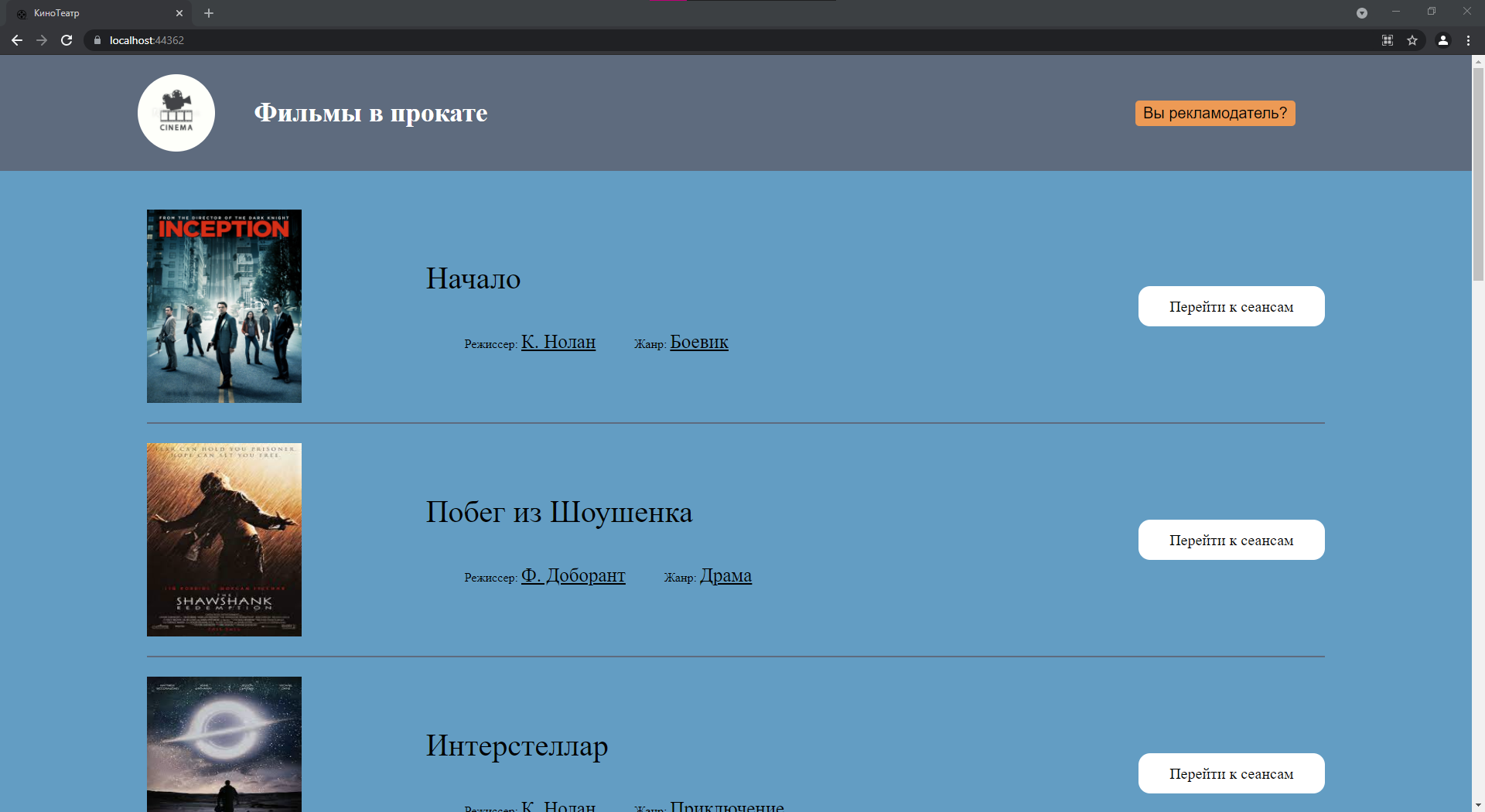


Рисунок 6.2.1 – Стартовая страница

После перехода к сеансам пользователь получает информацию о доступных сеансах, а также развернутую информацию о фильмах. Данная страница приведена на рисунке 6.2.2.

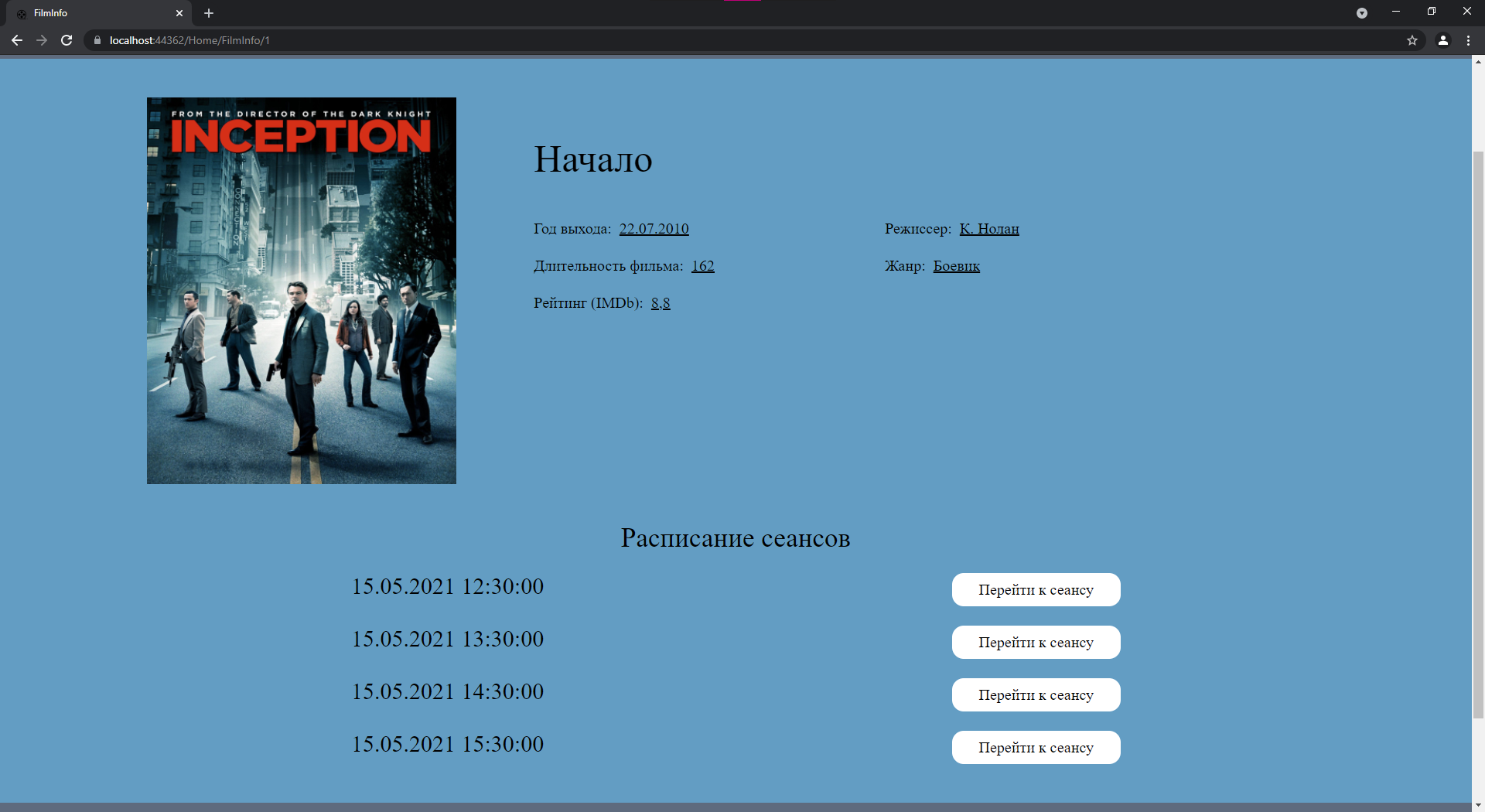


Рисунок 6.2.2 – Страница фильма

Далее, при выборе сеанса будет выведены данные о нем: доступные (полупрозрачные), выбранные (обведенные) и занятые (непрозрачные) места, информация о сеансе и кнопка покупки билета. Страница приведена на рисунке 6.2.3.

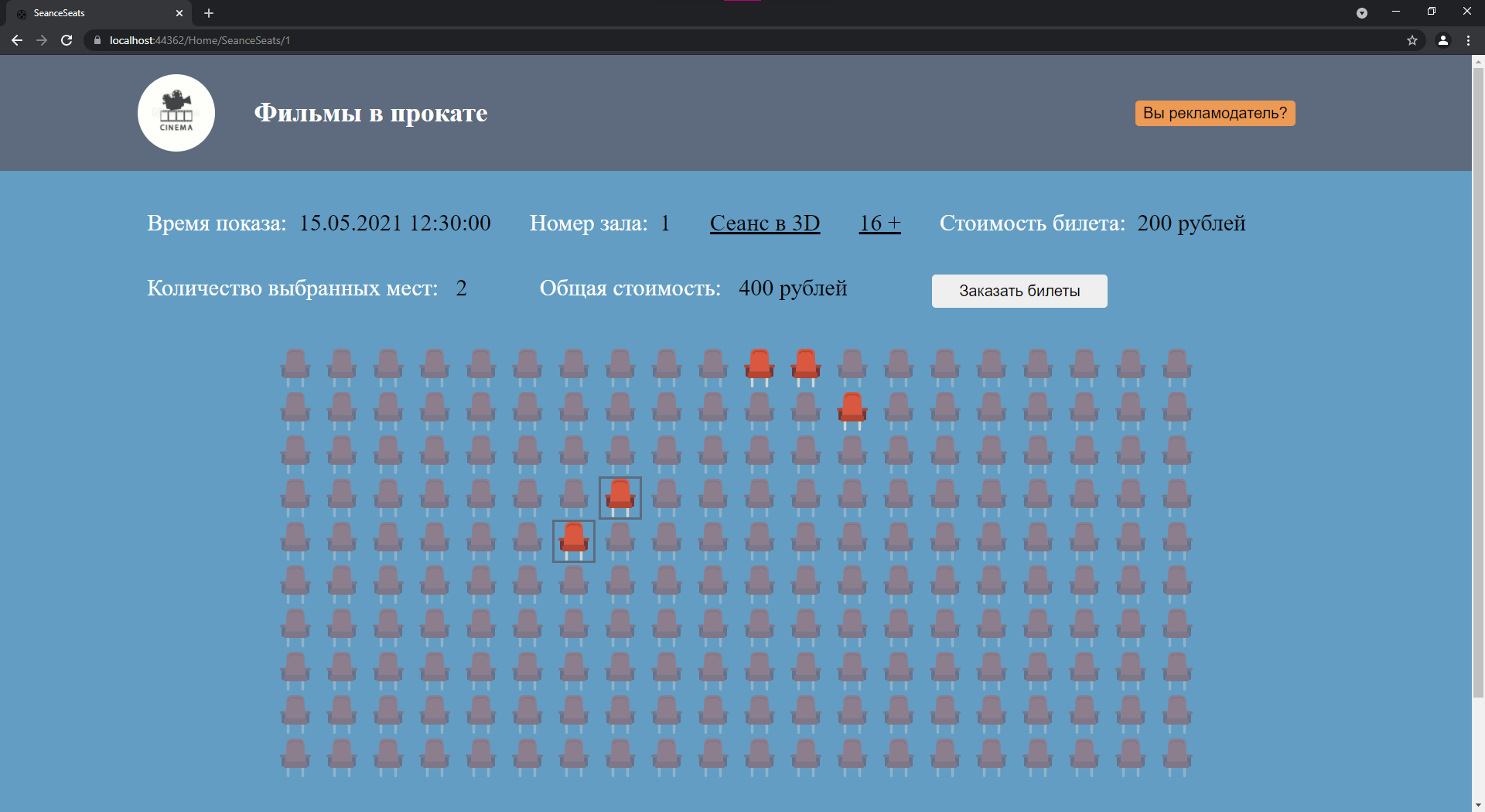


Рисунок 6.2.3 – Страница сеанса фильма

1. **Разработка программной документации**

**Назначение программы**

Программа представляет собой веб-приложение кинотеатра, позволяющее получать информацию о фильмах, сеансах, а также заказывать (бронировать) билеты на выбранные сеансы.

**Программные и аппаратные средства**

Для корректной работы веб-сайта в качестве аппаратного средства нужен ПК, обладающий следующими минимальными характеристиками:

* ОС: любая ОС;
* ОЗУ: не менее 1 Гб;
* Наличие интернет-соединения.

В качестве программного средства необходим любой современный браузер.

**Выполнение программы**

Для данной курсовой работы сайт не был помещен на хостинг в связи с проблемами установления работы базы данных. Поэтому запуск сайта может быть осуществлен только через среду разработки Microsoft Visual Studio.

Вся работа производится в браузере.

Для завершения работы можно либо закрыть браузер, либо остановить отладку в Microsoft Visual Studio.

**Заключение**

По итогу данного курсового проекта было произведено проектирование системы кинотеатра. Была составлена база данных, содержащая информацию об основных объектах кинотеатра, таких как фильмы, сеансы, сотрудники и др. А также разработано веб-приложение, которое, в ограниченном формате, позволяет производить действия над базой данных, а именно выборку данных и модификацию таблиц(ы). Кроме того, было произведено тестирование созданных объектов базы данных.

Данная работа позволила получить новые и улучшить уже имеющиеся навыки работы с базами данных, анализом исследуемой системы, проектированием и реализацией программной части проекта, проведением тестирования и составления программной документации.

**Список использованной литературы**

1. Гринченко Н.Н. Конспект лекций по дисциплине «Базы данных и клиент-серверные приложения», 2021.
2. Райордан Р. Основы реляционных баз данных. - Русская Редакция, 2001. - 390 с.
3. ГОСТ 19.503-79 ЕСПД. Руководство системного программиста. Требования к содержанию и оформлению.

ПРИЛОЖЕНИЕ А: исходный текст клиентской программы.

Код Controller:

using System;

using System.Collections.Generic;

using System.Linq;

using System.Web.Mvc;

using DAL.Department;

using Newtonsoft.Json;

namespace CinemaClientApplication.Controllers

{

public class HomeController : Controller

{

private readonly SQLManagerDAO requestDAO;

public HomeController()

{

requestDAO = new SQLManagerDAO();

}

public ActionResult Index()

{

return View(requestDAO.GetAllFilms().ToList());

}

public ActionResult FilmInfo(int id)

{

Models.SeancesFilmModel model = new Models.SeancesFilmModel

{

Seances = requestDAO.GetSeancesForFilms(id).ToList(),

Film = requestDAO.GetFilmById(id)

};

return View(model);

}

public ActionResult SeanceSeats(int id)

{

Models.SeanceHollModel model = new Models.SeanceHollModel

{

Tickets = requestDAO.GetTicketsForSeance(id).ToList(),

Holl = requestDAO.GetCinemaHollInfoBySeance(id),

Seance = requestDAO.GetSeanceInfo(id)

};

return View(model);

}

[HttpPost]

public ActionResult CreateTickets(string tickets)

{

var jsonRow = JsonConvert.DeserializeObject<string[]>(tickets);

List<Entities.Ticket> ticketsInfo = new List<Entities.Ticket>();

foreach (var str in jsonRow)

{

string[] row = str.Split('-');

ticketsInfo.Add(new Entities.Ticket(Entities.Ticket.LastId + 1, "standart", 0, Convert.ToInt32(row[0]), Convert.ToInt32(row[1]), Convert.ToInt32(row[2]), Convert.ToInt32(row[3])));

}

requestDAO.CreateTickets(ticketsInfo);

return RedirectToAction("Index");

}

}

}

Код класса «Билет» (остальные классы отношений подобны приведенному):

namespace Entities

{

public class Ticket

{

public static int LastId { private set; get; }

public int TicketID { private set; get; }

public string TicketType { private set; get; }

public int CashboxID { private set; get; }

public int SeanceId { private set; get; }

public int RowNumber { private set; get; }

public int SeatNumber { private set; get; }

public decimal Cost { private set; get; }

public Ticket(int id, string type, int cashId, int seanceId, int row, int seat, int cost)

{

TicketID = LastId = id;

TicketType = type;

CashboxID = cashId;

SeanceId = seanceId;

RowNumber = row;

SeatNumber = seat;

Cost = cost;

}

}

}

Код класса «SQLManager»:

using System;

using System.Collections.Generic;

using System.Data.SqlClient;

using Entities;

namespace DAL.Department

{

public class SQLManagerDAO

{

private readonly SqlConnection DBConnection;

public SQLManagerDAO()

{

DBConnection = DBAccess.GetDBConnection();

}

public IEnumerable<Film> GetAllFilms()

{

if (DBConnection.State != System.Data.ConnectionState.Open)

DBConnection.Open();

try

{

SqlCommand command = new SqlCommand("GetAllFilms", DBConnection)

{

CommandType = System.Data.CommandType.StoredProcedure

};

SqlDataReader dataReader = command.ExecuteReader();

while (dataReader.Read())

{

int id = dataReader.GetByte(0);

string name = dataReader.GetString(dataReader.GetOrdinal("FilmName"));

DateTime year = dataReader.GetDateTime(dataReader.GetOrdinal("ReleaseYear"));

string director = dataReader.GetString(dataReader.GetOrdinal("Director"));

int duration = dataReader.GetInt16(dataReader.GetOrdinal("Duration"));

string genre = dataReader.GetString(dataReader.GetOrdinal("Genre"));

float rating = (float)dataReader.GetDouble(dataReader.GetOrdinal("Rating"));

string image = dataReader.GetString(dataReader.GetOrdinal("FilmImage"));

yield return new Film(id, name, year, director, duration, genre, rating, image); ;

}

} finally

{

DBConnection.Close();

}

}

public IEnumerable<Ticket> GetTicketsForSeance(int seance)

{

if (DBConnection.State != System.Data.ConnectionState.Open)

DBConnection.Open();

try

{

SqlCommand command = new SqlCommand("GetTicketsForSeance", DBConnection)

{

CommandType = System.Data.CommandType.StoredProcedure

};

command.Parameters.AddWithValue("@seance", seance);

SqlDataReader dataReader = command.ExecuteReader();

while (dataReader.Read())

{

int id = dataReader.GetInt16(dataReader.GetOrdinal("TicketID"));

string type = dataReader.GetString(dataReader.GetOrdinal("TicketType"));

int cashId = dataReader.GetByte(dataReader.GetOrdinal("CashboxID"));

int seanceId = dataReader.GetByte(dataReader.GetOrdinal("SeanceId"));

int row = dataReader.GetByte(dataReader.GetOrdinal("RowNumber"));

int seat = dataReader.GetByte(dataReader.GetOrdinal("SeatNumber"));

int cost = dataReader.GetSqlMoney(dataReader.GetOrdinal("Cost")).ToInt32();

yield return new Ticket(id, type, cashId, seanceId, row, seat, cost);

}

}

finally

{

DBConnection.Close();

}

}

public Seance GetSeanceInfo(int seance)

{

if (DBConnection.State != System.Data.ConnectionState.Open)

DBConnection.Open();

try

{

SqlCommand command = new SqlCommand(@"GetSeanceInfo", DBConnection)

{

CommandType = System.Data.CommandType.StoredProcedure

};

command.Parameters.AddWithValue("@seance", seance);

SqlDataReader dataReader = command.ExecuteReader();

while (dataReader.Read())

{

int id = dataReader.GetByte(dataReader.GetOrdinal("SeanceId"));

int fimlId = dataReader.GetByte(dataReader.GetOrdinal("FilmID"));

int hollId = dataReader.GetByte(dataReader.GetOrdinal("HollID"));

DateTime time = dataReader.GetDateTime(dataReader.GetOrdinal("ShowTime"));

int rating = dataReader.GetByte(dataReader.GetOrdinal("AgeRating"));

string type = dataReader.GetString(dataReader.GetOrdinal("SeanceType"));

int cost = dataReader.GetSqlMoney(dataReader.GetOrdinal("TicketCost")).ToInt32();

return new Seance(id, fimlId, hollId, time, rating, type, cost);

}

}

finally {

DBConnection.Close();

}

return null;

}

public Film GetFilmById(int film)

{

if (DBConnection.State != System.Data.ConnectionState.Open)

DBConnection.Open();

try

{

SqlCommand command = new SqlCommand(@"GetFilmById", DBConnection);

command.CommandType = System.Data.CommandType.StoredProcedure;

command.Parameters.AddWithValue("@film", film);

SqlDataReader dataReader = command.ExecuteReader();

while (dataReader.Read())

{

int id = dataReader.GetByte(0);

string name = dataReader.GetString(dataReader.GetOrdinal("FilmName"));

DateTime year = dataReader.GetDateTime(dataReader.GetOrdinal("ReleaseYear"));

string director = dataReader.GetString(dataReader.GetOrdinal("Director"));

int duration = dataReader.GetInt16(dataReader.GetOrdinal("Duration"));

string genre = dataReader.GetString(dataReader.GetOrdinal("Genre"));

float rating = (float)dataReader.GetDouble(dataReader.GetOrdinal("Rating"));

string image = dataReader.GetString(dataReader.GetOrdinal("FilmImage"));

return new Film(id, name, year, director, duration, genre, rating, image);

}

}

finally

{

DBConnection.Close();

}

return null;

}

public IEnumerable<Seance> GetSeancesForFilms(int film)

{

if (DBConnection.State != System.Data.ConnectionState.Open)

DBConnection.Open();

try

{

SqlCommand command = new SqlCommand(@"SELECT\* FROM Seances WHERE FilmID = @FilmID", DBConnection);

command.Parameters.AddWithValue("@FilmID", film);

SqlDataReader dataReader = command.ExecuteReader();

while (dataReader.Read())

{

int id = dataReader.GetByte(dataReader.GetOrdinal("SeanceId"));

int fimlId = dataReader.GetByte(dataReader.GetOrdinal("FilmID"));

int hollId = dataReader.GetByte(dataReader.GetOrdinal("HollID"));

DateTime time = dataReader.GetDateTime(dataReader.GetOrdinal("ShowTime"));

int rating = dataReader.GetByte(dataReader.GetOrdinal("AgeRating"));

string type = dataReader.GetString(dataReader.GetOrdinal("SeanceType"));

int cost = dataReader.GetSqlMoney(dataReader.GetOrdinal("TicketCost")).ToInt32();

yield return new Seance(id, fimlId, hollId, time, rating, type, cost);

}

}

finally

{

DBConnection.Close();

}

}

public CinemaHoll GetCinemaHollInfoBySeance(int seance)

{

if (DBConnection.State != System.Data.ConnectionState.Open)

DBConnection.Open();

try

{

SqlCommand command = new SqlCommand(@"GetHollInfoBySeance", DBConnection)

{

CommandType = System.Data.CommandType.StoredProcedure

};

command.Parameters.AddWithValue("@seance", seance);

SqlDataReader dataReader = command.ExecuteReader();

while (dataReader.Read())

{

int id = dataReader.GetByte(dataReader.GetOrdinal("HollID"));

bool td = dataReader.GetBoolean(dataReader.GetOrdinal("TdEnable"));

int rows = dataReader.GetByte(dataReader.GetOrdinal("RowNumber"));

int seats = dataReader.GetByte(dataReader.GetOrdinal("SeatNumber"));

return new CinemaHoll(id, td, rows, seats);

}

}

finally

{

DBConnection.Close();

}

return null;

}

public void CreateTickets(List<Ticket> tickets)

{

if (DBConnection.State != System.Data.ConnectionState.Open)

DBConnection.Open();

try

{

foreach (var ticket in tickets)

{

SqlCommand command = new SqlCommand(@"CreateTicket", DBConnection)

{

CommandType = System.Data.CommandType.StoredProcedure

};

command.Parameters.AddWithValue("@SeanceId", ticket.SeanceId);

command.Parameters.AddWithValue("@TypeName", ticket.TicketType);

command.Parameters.AddWithValue("@CashboxID", ticket.CashboxID);

command.Parameters.AddWithValue("@RowNumber", ticket.RowNumber);

command.Parameters.AddWithValue("@SeatNumber", ticket.SeatNumber);

command.Parameters.AddWithValue("@Cost", ticket.Cost);

if (command.ExecuteNonQuery() < 1)

{

throw new Exception("Error with ticket");

}

}

}

finally

{

DBConnection.Close();

Код стартовой страницы:

@{

ViewBag.Title = "КиноТеатр";

}

<div class="films-area">

@foreach (var item in Model)

{

<div class="film-item">

<div class="film-logo">

<img src="~/Content/img/films.img/@item.FilmImage" alt="Alternate Text" />

</div>

<div class="film-label">

<span>@item.FilmName</span>

<div class="film-short-info">

<div class="film-director">

<span>Режиссер:</span> <span>@item.Director</span>

</div>

<div class="film-director">

<span>Жанр: </span> <span>@item.Genre</span>

</div>

</div>

</div>

<div class="film-button">

@Html.ActionLink("Перейти к сеансам", "FilmInfo", "Home", new { id = @item.FilmID }, new { @class = "button" })

</div>

</div>

}

</div>

Код страницы с информацией о фильме:

@{

ViewBag.Title = "FilmInfo";

}

<div class="film-area">

<div class="image">

<img src="~/Content/img/films.img/@Model.Film.FilmImage" alt="Alternate Text" />

</div>

<div class="info">

<div class="label">

<span>@Model.Film.FilmName</span>

</div>

<div class="info-table">

<div><span>Год выхода: &nbsp;</span><span> @Model.Film.ReleaseYear.ToString("dd/MM/yyyy")</span></div>

<div><span>Режиссер: &nbsp;</span><span> @Model.Film.Director</span></div>

<div><span>Длительность фильма: &nbsp;</span><span> @Model.Film.Duration</span></div>

<div><span>Жанр: &nbsp;</span><span> @Model.Film.Genre</span></div>

<div><span>Рейтинг (IMDb): &nbsp;</span><span> @Model.Film.Rating</span></div>

</div>

</div>

</div>

<div class="seances-area">

@if (Model.Seances.Count == 0)

{

<span class="no-seances">На данные фильм сеансов нет!</span>

} else

{

<div class="seance-general-label">

<span>Расписание сеансов</span>

</div>

}

@foreach (var seance in Model.Seances)

{

<div class="seance-item">

<div class="seance-time">

<span>@seance.ShowTime</span>

</div>

<div class="seance-button">

@Html.ActionLink("Перейти к сеансу", "SeanceSeats", "Home", new { id = @seance.SeanceId }, new { @class = "button" })

</div>

</div>

}

</div>

Код страницы выбранного сеанса:

@using Newtonsoft.Json;

@{

ViewBag.Title = "SeanceSeats";

}

<script>

let lockedSeats = [];

let selectedTickets = [];

let ticketCost = @Model.Seance.TicketCost;

let seanceId = @Model.Seance.SeanceId;

let seatPlace;

let ticketsOrdered;

function LockSeats(row, seat) {

let str = row + '-' + seat;

seat = document.querySelector(`img[alt = "${str}"]`);

$(seat).css({ 'opacity': 1, 'cursor': 'default' });

lockedSeats.push(str);

}

</script>

<div class="seance-info">

<div><span>Время показа: &nbsp;</span><span>@Model.Seance.ShowTime</span></div>

<div><span>Номер зала: &nbsp;</span><span>@Model.Holl.HollID</span></div>

@if (Model.Holl.TdEnable)

{<div><span style="text-decoration: underline">Сеанс в 3D</span></div>}

<div><span style="text-decoration: underline">@Model.Seance.AgeRating +</span></div>

<div><span>Стоимость билета: &nbsp;</span><span>@Model.Seance.TicketCost рублей</span></div>

</div>

<div class="seats-handler">

<div><span>Количество выбранных мест: &nbsp;</span> <span class="seats-count"></span></div>

<div><span>Общая стоимость: &nbsp;</span> <span class="ticket-cost"></span></div>

<div><button disabled>Заказать билеты</button></div>

</div>

<div class="seats-area">

@for (int i = 0; i < Model.Holl.RowNumber; i++)

{

<div class="seat-row">

@for (int j = 0; j < Model.Holl.SeatNumber; j++)

{

<div class="seat">

<img src="~/Content/img/seat.svg" alt="@(i+1)-@(j+1)" />

</div>

}

</div>

}

</div>

<div class="script-area">

@foreach (var ticket in Model.Tickets)

{

int row = ticket.RowNumber;

int seat = ticket.SeatNumber;

<script>

LockSeats(@(row), @(seat));

</script>

}

</div>

Код файла «Script.js»:

$(document).ready(function () {

$('.seance-button').hover(function () {

$(this).siblings('.seance-time').css('text-decoration', 'underline');

}, function () {

$(this).siblings('.seance-time').css('text-decoration', 'none');

});

$('.seat').on("click", function () {

let seatPlace = $(this).children('img').attr('alt');

let row = seatPlace.split('-')[0];

let seat = seatPlace.split('-')[1];

if (lockedSeats.indexOf(row + '-' + seat) != -1) {

return;

}

if (selectedTickets.indexOf(row + '-' + seat) != -1) {

selectedTickets.splice(selectedTickets.indexOf(row + '-' + seat), 1);

$(this).children('img').css({ 'outline': 'none', 'opacity': .4 });

} else {

$(this).children('img').css({ 'outline': '3px solid #5c6b7f', 'opacity': 1 });

selectedTickets.push(row + '-' + seat);

}

drawTicketInfo();

});

function drawTicketInfo() {

if (selectedTickets.length > 0) {

$('.seats-count').text(selectedTickets.length);

$('.ticket-cost').text(selectedTickets.length \* ticketCost + ' рублей');

$('.seats-handler > div > button').prop('disabled', false);

$('.seats-handler > div > button').css({ 'opacity': 1, 'cursor': 'pointer' });

}

else {

$('.seats-count').text("");

$('.ticket-cost').text("");

$('.seats-handler > div > button').prop('disabled', true);

$('.seats-handler > div > button').css({ 'opacity': .5, 'cursor': 'default' });

}

}

$(".seats-handler button").on("click", ticketOrdered = function () {

let dataRow = [];

for (var i = 0; i < selectedTickets.length; i++) {

dataRow.push(seanceId + '-' + selectedTickets[i] + '-' + ticketCost);

}

let data = JSON.stringify(dataRow);

$.ajax({

url: "../CreateTickets",

type: "POST",

data: { tickets: data },

success: function () {

ShowOrdered();

//window.location = "../Index";

},

error: function () {

console.log(data);

}

});

});

function ShowOrdered() {

$('.wrapper').css('z-index', -1);

$('.ordered-active').css('visibility', 'visible');

}

});