**ОБЛАСТНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ПРОФЕССИОНАЛЬНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ**

**«ТОМСКИЙ ЭКОНОМИКО-ПРОМЫШЛЕННЫЙ КОЛЛЕДЖ»**

**Проектирование и разработка интерфейсной части для информационной системы "Popcorn Film" на платформе Windows Presentation Foundation**

***ОТЧЁТ ПО УЧЕБНОЙ ПРАКТИКЕ***

*СПЕЦИАЛЬНОСТЬ 09.02.07 ИНФОРМАЦИОННЫЕ СИСТЕМЫ И ПРОГРАММИРОВАНИЕ*

Выполнил:

Студент группы 1912с

\_\_\_\_\_\_\_\_\_\_\_ / Тимофеев А.В.

«\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_ 2024 г.

Проверил:

Руководитель УП

\_\_\_\_\_\_\_\_\_\_\_ / Уляхин В.А.

«\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_ 2024 г.

Томск 2024

Оглавление

[ВВЕДЕНИЕ 3](#_Toc166288086)

[Информация о проекте и команде разработчиков 3](#_Toc166288087)

[Назначение и структура документа 3](#_Toc166288088)

[Глоссарий 4](#_Toc166288089)

[ОБЩЕЕ ОПИСАНИЕ 5](#_Toc166288090)

[Общий взгляд на предлагаемое решение 5](#_Toc166288091)

[**ТЕХНИЧЕСКОЕ ЗАДАНИЕ** 6](#_Toc166288092)

[**1.** **Основные требования** 6](#_Toc166288093)

[**2.** **Технические детали** 7](#_Toc166288094)

[**3.** **Дополнительные требования** 7](#_Toc166288095)

[**4.** **Заключение** 7](#_Toc166288096)

[ТЕХНИЧЕСКАЯ РЕАЛИЗАЦИЯ 8](#_Toc166288097)

[Концептуальная Модель 8](#_Toc166288098)

[Разработка приложения 15](#_Toc166288099)

[ТЕСТИРОВАНИЕ 22](#_Toc166288100)

[Авторизация пользователя 22](#_Toc166288101)

# ВВЕДЕНИЕ

## Информация о проекте и команде разработчиков

|  |  |
| --- | --- |
| Название проекта | «Popcorn Film» |
| Цель проекта | Разработать сайт для кинотеатра с возможностью получения информации о фильмах, сеансах и бронировании билетов. А также разработать WPF приложение с возможностью работы с данными, которые используются на сайте. API позволит управлять данными кинотеатра через веб-интерфейс, включая регистрацию, аутентификацию пользователей и управление профилями и т.д. |
| Задачи проекта | 1. Разработать техническое задания API сервиса. 2. Разработать техническое задания веб-сайта. 3. Разработать техническое задания десктоп приложения. 4. Разработать API сервис на фреймворке Laravel 10. 5. Разработать веб-сайт на фреймворке Vue 3. 6. Разработать десктоп приложение на C# WPF. 7. Разработать автоматизированные тесты и провести тестирование API сервиса. 8. Разработать автоматизированные тесты и провести тестирование веб-сайта. 9. Разработать автоматизированные тесты и провести тестирование десктоп приложения. |
| Название команды | «MRT» |
| Состав команды и роли | Мотов Алибала Эльманович – Аnalyst, Backend developer, QA.  Ридт Денис Александрович– Аnalyst, Frontend developer, QA.  Тимофеев Александр Викторович – Аnalyst, C# developer, QA. |

## Назначение и структура документа

Настоящий документ рассчитан на использование в проектной команде в качестве ключевого документа, определяющего требования по разработке приложения WPF для проекта «Popcorn Film».

## Глоссарий

|  |  |
| --- | --- |
| **Понятие** | **Определение понятия** |
| WPF | WPF (Windows Presentation Foundation) – Технология, используемая для создания графических пользовательских интерфейсов в приложениях под операционную систему Windows. |
| API | API (интерфейс программирования приложений) – это набор определенных правил и соглашений, который используется для взаимодействия между различными программными компонентами. API определяет, как различные части программного обеспечения могут взаимодействовать друг с другом, обычно путем обмена данными и выполнения определенных операций. |
| Draw.io | Это бесплатный онлайн инструмент для создания диаграмм и схем. Он предоставляет широкий набор инструментов и элементов, которые позволяют пользователям создавать различные типы диаграмм, такие как организационные диаграммы, блок-схемы, ER-диаграммы, сетевые диаграммы, диаграммы потока процессов и многое другое. |
| CRUD операции | CRUD операции (Create, Read, Update, Delete) – Основные операции, поддерживаемые многими API для работы с данными: создание, чтение, обновление и удаление. |

# ОБЩЕЕ ОПИСАНИЕ

## Общий взгляд на предлагаемое решение

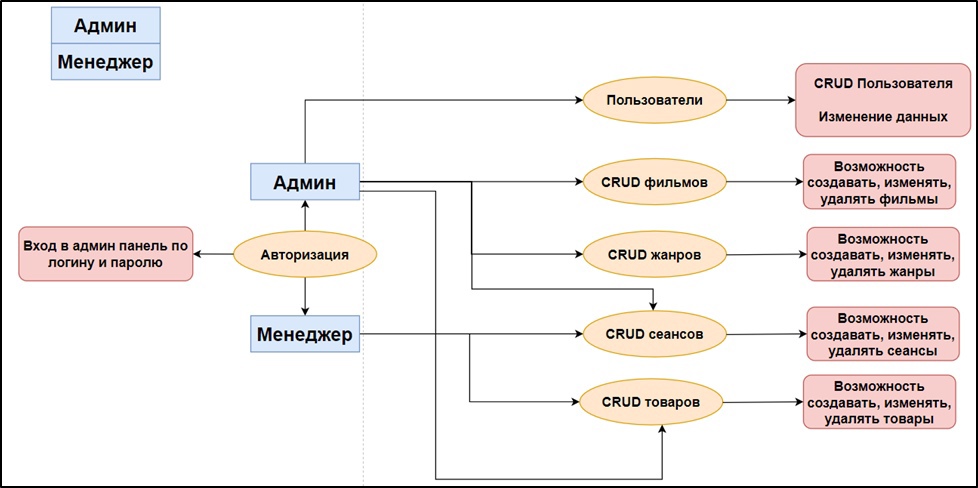


Рисунок 1 Структура приложения

## Ролевая модель

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Раздел | Фичи | Админ | Менеджер | Пользователь | Гость |
| Афиша | Просмотр списка фильмов | Да | Да | Да | Да |
| Просмотр информации о фильмах | Да | Да | Да | Да |
| Добавление фильма в афишу | Да | Нет | Нет | Нет |
| Редактирование фильма в афишу | Да | Нет | Нет | Нет |
| Удаление фильма в афишу | Да | Нет | Нет | Нет |
| Пользователь | Регистрация | Нет | Нет | Нет | Да |
| Авторизация | Да | Да | Да | Нет |
| Просмотр всех пользователей | Да | Нет | Нет | Нет |
| Просмотр конкретного пользователя | Да | Да | Да (себя) | Нет |
| Редактирование информации о пользователе | Да | Да (себя) | Да (себя) | Нет |
| Удаление пользователя | Да | Да (себя) | Да (себя) | Нет |
| Просмотр ролей | Да | Нет | Нет | Нет |
| Жанр | Добавление жанра | Да | Нет | Нет | Нет |
| Изменение жанра | Да | Нет | Нет | Нет |
| Просмотр жанра | Да | Да | Да | Да |
| Удаление жанра | Да | Нет | Нет | Нет |
| Роли | Изменение роли | Да | Нет | Нет | Нет |
| Просмотр ролей | Да | Нет | Нет | Нет |
| Товар | Добавление товара | Да | Да | Нет | Нет |
| Просмотр товаров | Да | Да | Да | Да |
| Обновление информации о товаре | Да | Да | Нет | Нет |
| Удаление товара | Да | Да | Нет | Нет |
| Корзина | Добавление в корзину | Да | Да | Да | Нет |
| Просмотр корзины | Да | Да | Да (свою) | Нет |
| Удаление корзины | Да | Да | Да (свою) | Нет |
| Сеанс | Добавление сеанса | Да | Да | Нет | Нет |
| Просмотр сеансов | Да | Да | Да | Да |
| Обновление сеанса | Да | Да | Нет | Нет |
| Удаление сеанса | Да | Да | Нет | Нет |
| Билет | Добавление билета | Да | Да | Да | Нет |

**ТЕХНИЧЕСКОЕ ЗАДАНИЕ**

Целью данного проекта является разработка приложения для взаимодействия с данными сайта “Popcorn Film”. Приложение будет предоставлять удобный интерфейс для управления фильмами, жанрами, сеансами, пользователями и продуктами.

1. **Основные требования**
   1. Авторизация

* При открытии приложения отображается страница авторизации.
* Поля на странице авторизации: логин, пароль.
* Кнопка "Войти" для подтверждения данных.
  1. Пользовательский интерфейс
* После успешной авторизации отображается информация о сотруднике: Фамилия, Имя, Роль.
* Предоставленные возможности зависят от роли:
* Менеджер: управление сеансами, продуктами.
* Администратор: управление пользователями, фильмами, жанрами, сеансами, продуктами.
  1. Функционал для Менеджера
* Управление сеансами
  + Просмотр сеансов,
  + Редактирование сеансов,
  + Добавление сеансов,
  + Удаление сеансов,
* Управление Продуктами
  + Просмотр продуктов,
  + Редактирование продуктов,
  + Добавление продуктов,
  + Удаление продуктов
  1. Функционал для Администратора
* Управление фильмами
  + Просмотр фильмов,
  + Редактирование фильмов,
  + Добавление фильмов,
  + Удаление фильмов,
* Управление жанрами
  + Просмотр жанров,
  + Редактирование жанров,
  + Добавление жанров,
  + Удаление жанров,
* Управление пользователями
  + Просмотр пользователей,
  + Редактирование пользователей,
  + Добавление пользователей,
  + Удаление пользователей.
* Управление сеансами
  + Просмотр сеансов,
  + Редактирование сеансов,
  + Добавление сеансов,
  + Удаление сеансов,
* Управление Продуктами
  + Просмотр продуктов,
  + Редактирование продуктов,
  + Добавление продуктов,
  + Удаление продуктов

1. **Технические детали**
   1. Платформа

* Приложение разрабатывается на WPF (Windows Presentation Foundation).

1. **Дополнительные требования**
   1. Интерфейс

* Интуитивно понятный и привлекательный пользовательский интерфейс.
* Поддержка русского языка.
  1. Фотографии фильмов
* Возможность добавления и просмотра фотографий фильмов.
  1. Фотографии продуктов
* Возможность добавления и просмотра фотографий продуктов.

1. **Заключение**

Данное техническое задание определяет основные требования к приложению для взаимодействия сотрудников “Popcorn Film”. Разработка приложения должна быть выполнена с учетом указанных требований и технических деталей.

# ТЕХНИЧЕСКАЯ РЕАЛИЗАЦИЯ

## Макет приложения

Разработка приложения началась с создания структуры приложения (Рис.1), после его создания было решено перейти к создания макета в draw.io.

Создание макета началось с страницы авторизации (Рис. 2).

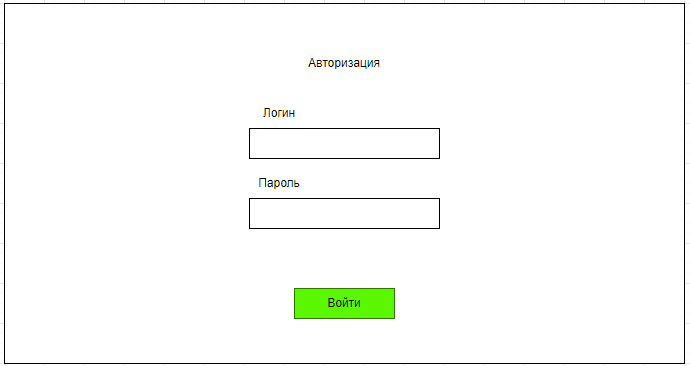


Рисунок 2 Концепт страницы авторизации

После авторизации были созданы макеты страниц с профилем администратора (Рис.3) и менеджера (Рис. 4). Так же создан макет страницы для редактирования текущего пользователя (Рис.5).

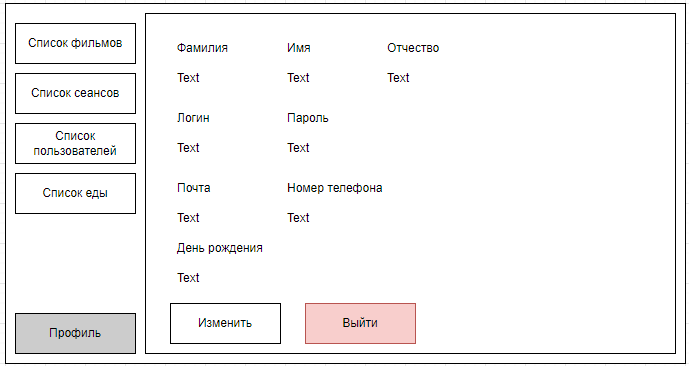


Рисунок 3 Макет страницы Профиль (Администратор)

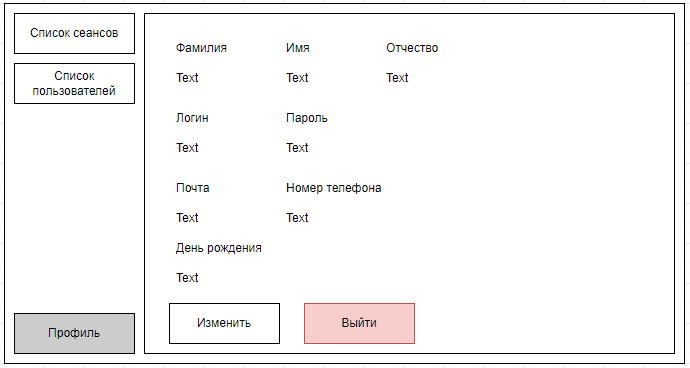


Рисунок 4 Макет страницы Профиль (Менеджер)



Рисунок 5 Макет страницы Редактирование пользователя

После создания профиля был сделан макет страниц для просмотра фильмов (Рис. 6), добавления (Рис. 7) и редактирования фильма (Рис. 8) у администратора.

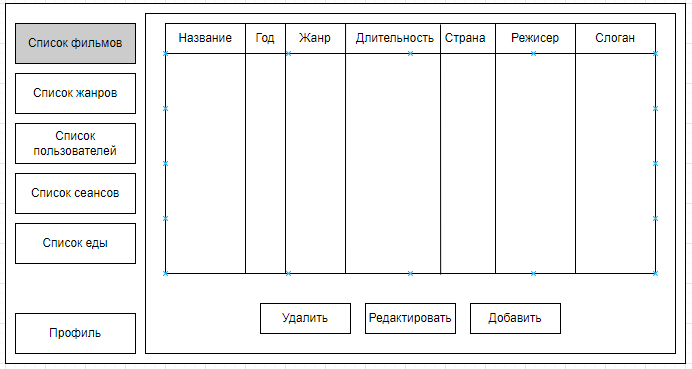


Рисунок 6 Макет страницы Список фильмов



Рисунок 7 Макет страницы Добавление фильма



Рисунок 8 Макет страницы Редактирование фильма

Для фильмов был создан макет страниц для просмотра (Рис. 9), создания (Рис. 10) и редактирования (Рис. 11) жанров у администратора.

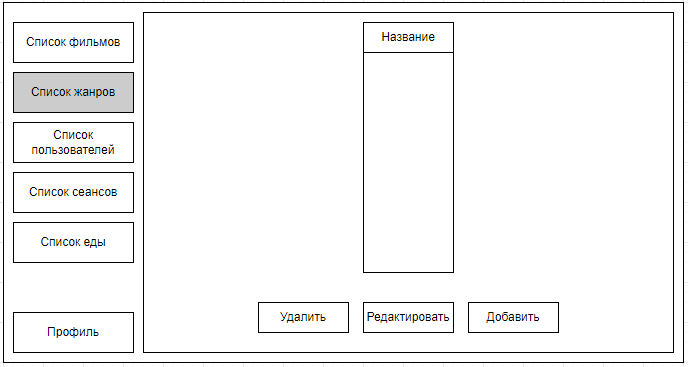


Рисунок 9 Макет страницы Список жанров

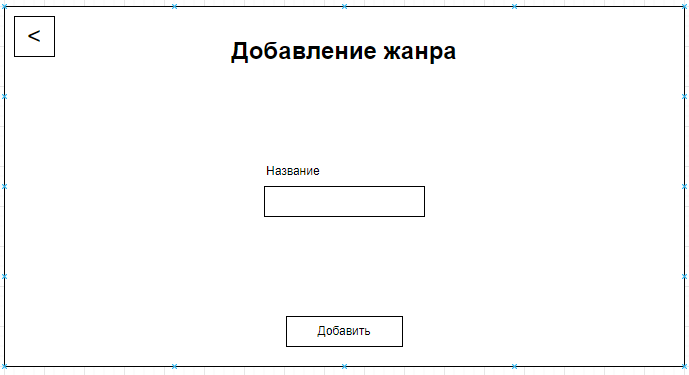


Рисунок 10 Макет страницы Добавление жанра

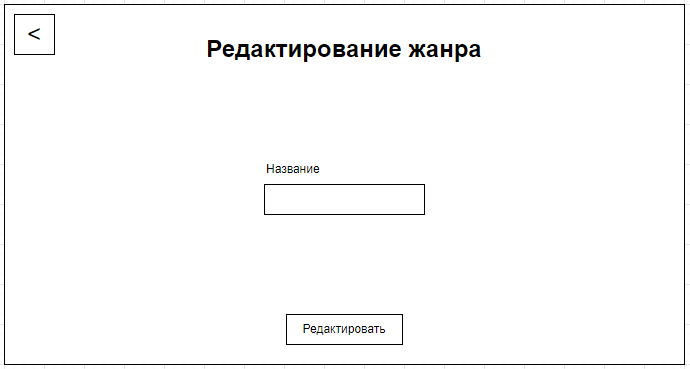


Рисунок 11 Макет страницы Редактирование жанра

Для работы с пользователем был создан следующий макет страниц: Список пользователей (Рис. 12), добавление пользователя (Рис. 13), редактирование пользователя (Рис. 14) у администратора.

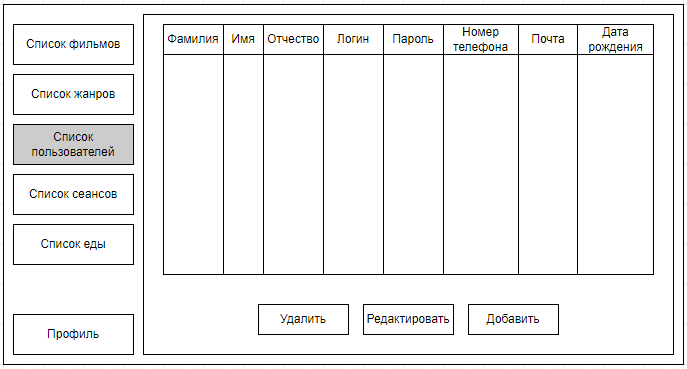


Рисунок 12 Макет страницы Список пользователей

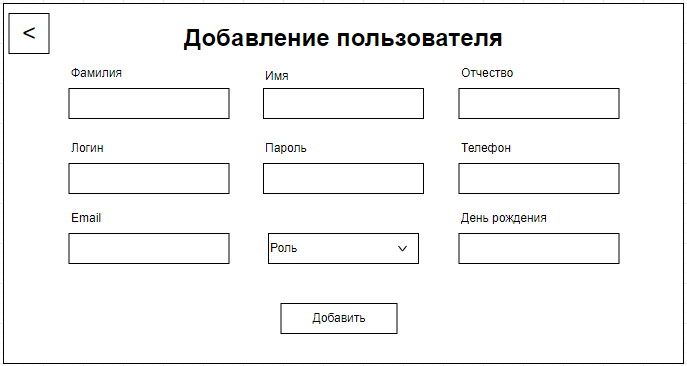


Рисунок 13 Макет страницы Добавление пользователя

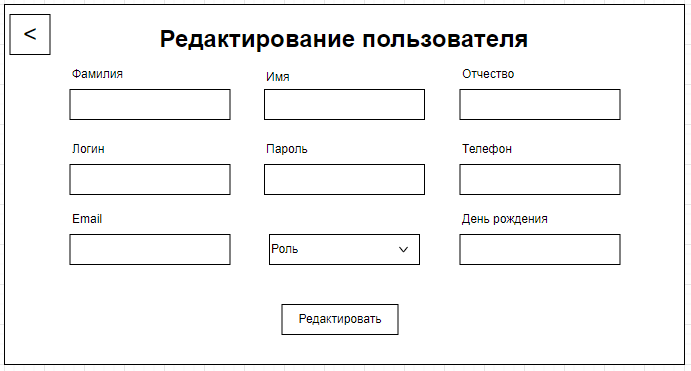


Рисунок 14 Макет страницы Редактирование пользователя

Для работы с сеансами был сделан следующий макет страниц: Список сеансов (Рис. 15), добавление сеанса (Рис. 16), редактирование сеанса (Рис. 17) у администратора и менеджера.

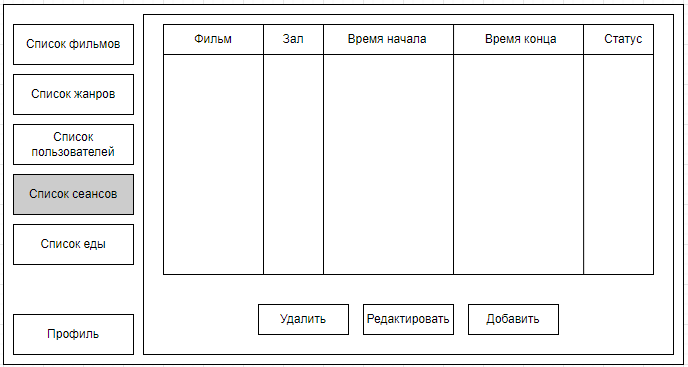


Рисунок 15 Макет страницы Список сеансов

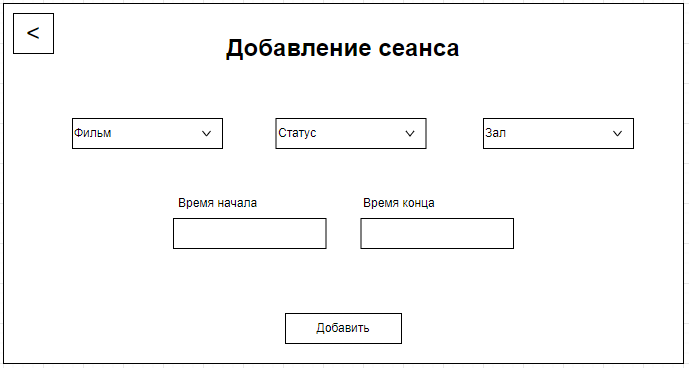


Рисунок 16 Макет страницы Добавление Сеансов

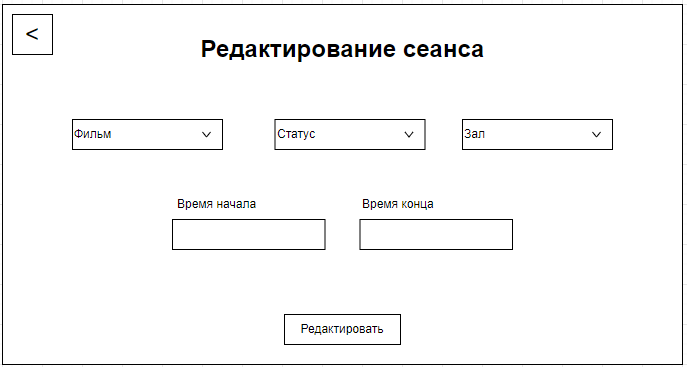


Рисунок 17 Макет страницы Редактирование сеанса

Для работы с едой был сделан следующий макет страниц: Список еды (Рис. 18), добавление еды (Рис. 19), редактирование еды (Рис. 20) у администратора и менеджера.

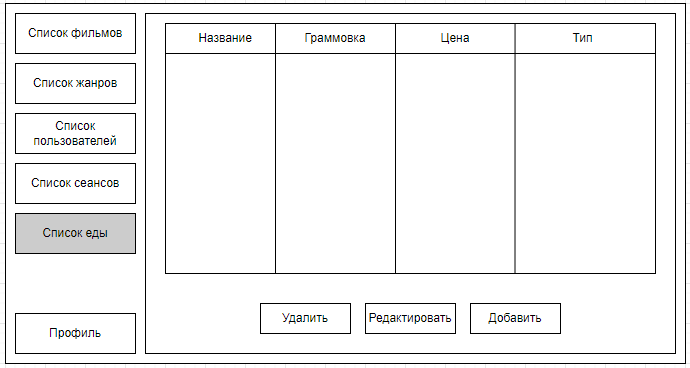


Рисунок 18 Макет страницы Список еды

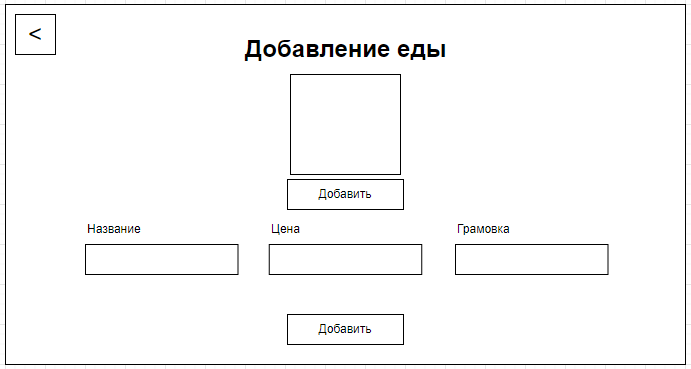


Рисунок 19 Макет страницы Добавление еды=

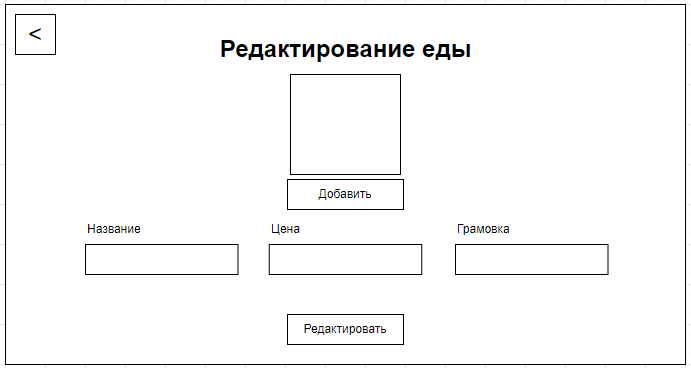


Рисунок 20 Макет страницы Редактирование еды

## Разработка приложения

После создания концептуальной модели было разработано приложение по готовому концепту.

Разработка приложения была начата с страниц авторизации (Рис. 21), профиля для менеджера (Рис. 22) и администратора (Рис. 23), также редактирование профиля (Рис. 24).

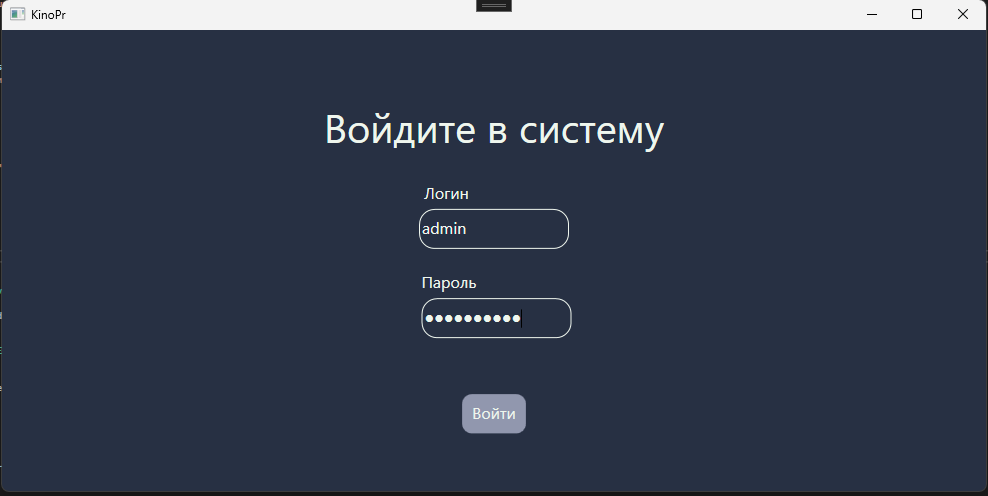


Рисунок 21 Готовая страница Авторизации

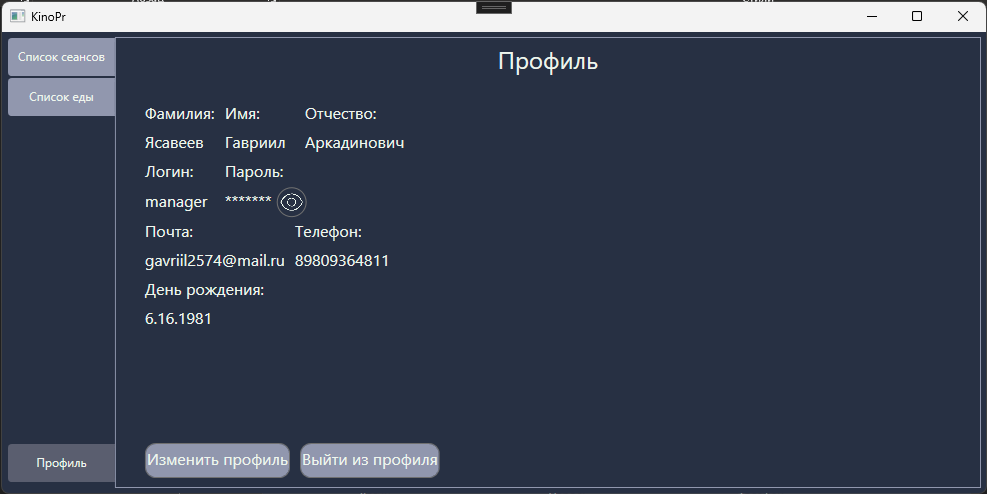


Рисунок 22 Готовая страница Профиля (Менеджер)

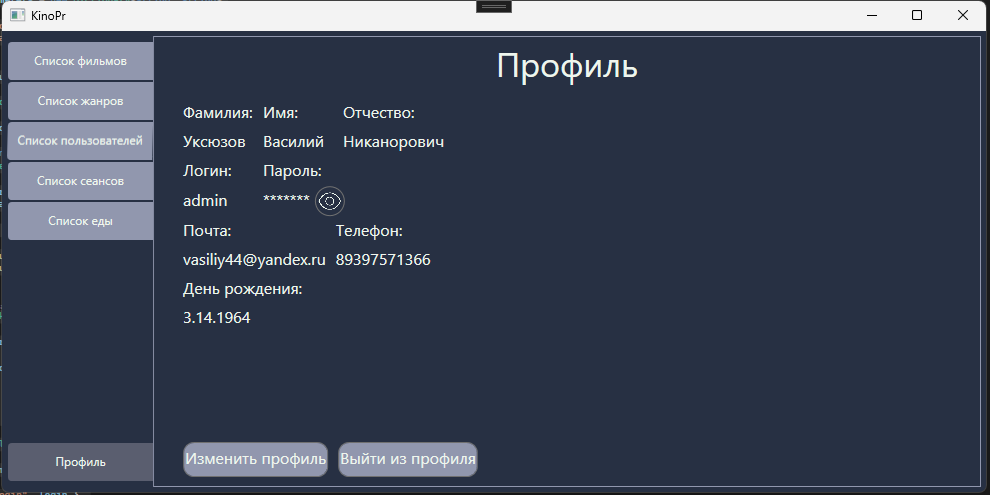


Рисунок 23 Готовая страница Профиль (Администратор)

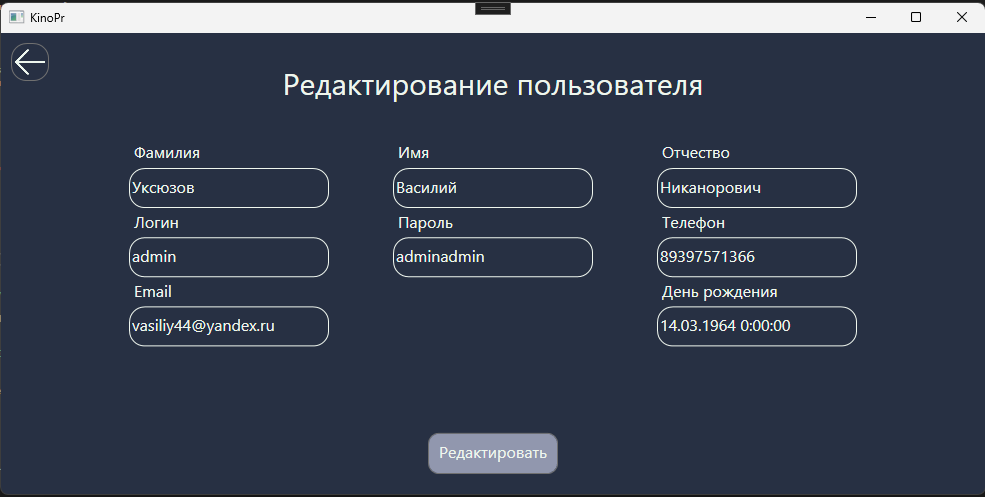


Рисунок 24 Готовая страница Редактирование текущего пользователя

После страниц с профилем были сделаны страницы с основным функционалом приложения, а именно: список фильмов (Рис. 25-27), список жанров (Рис. 28-30), список пользователей (Рис. 31-33), список сеансов (Рис. 34-36), список еды (Рис. 37-39).



Рисунок 25 Готовая страница Список фильмов

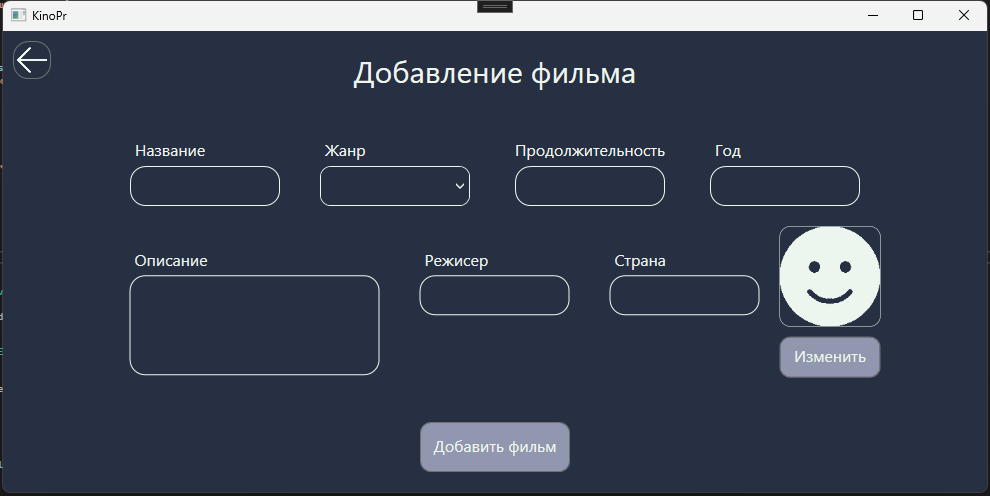


Рисунок 26 Готовая страница Добавление фильма

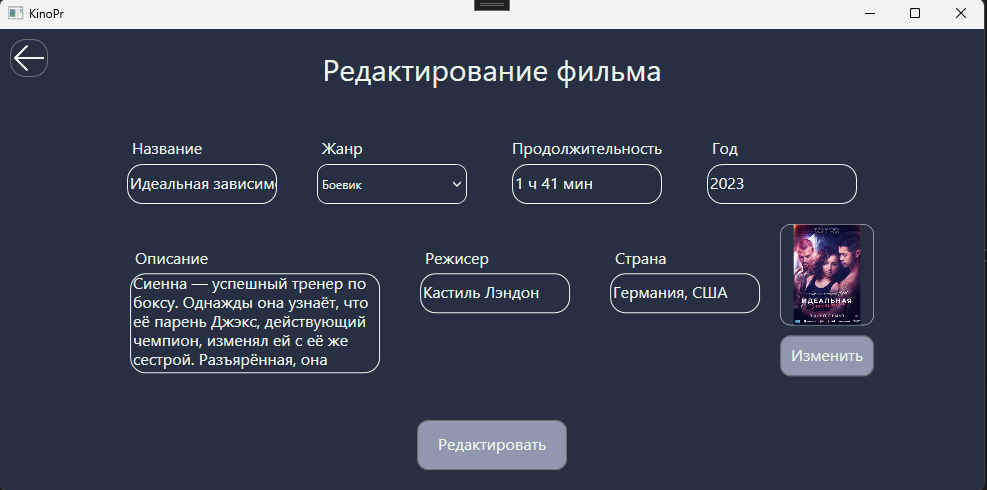


Рисунок 27 Готовая страница Редактирование фильма

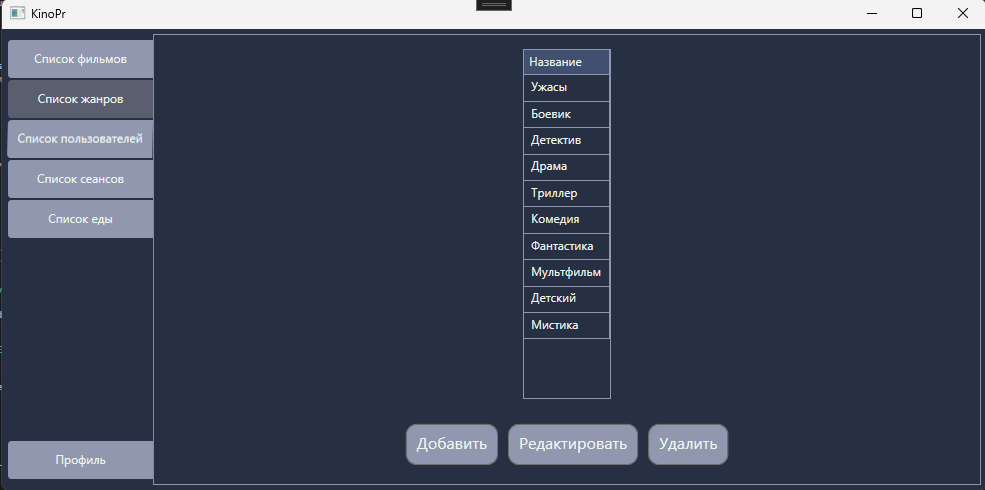


Рисунок 28 Готовая страница Список фильмов

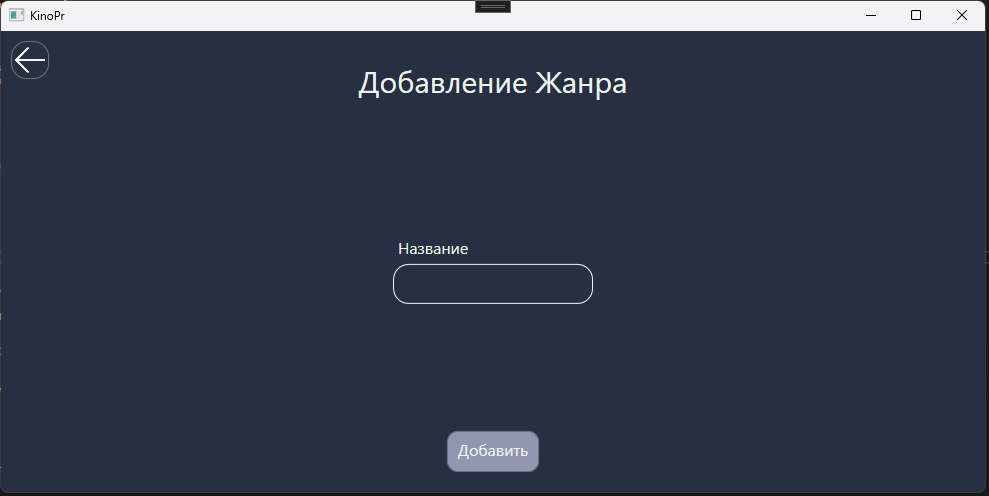


Рисунок 29 Готовая страница Добавление жанра

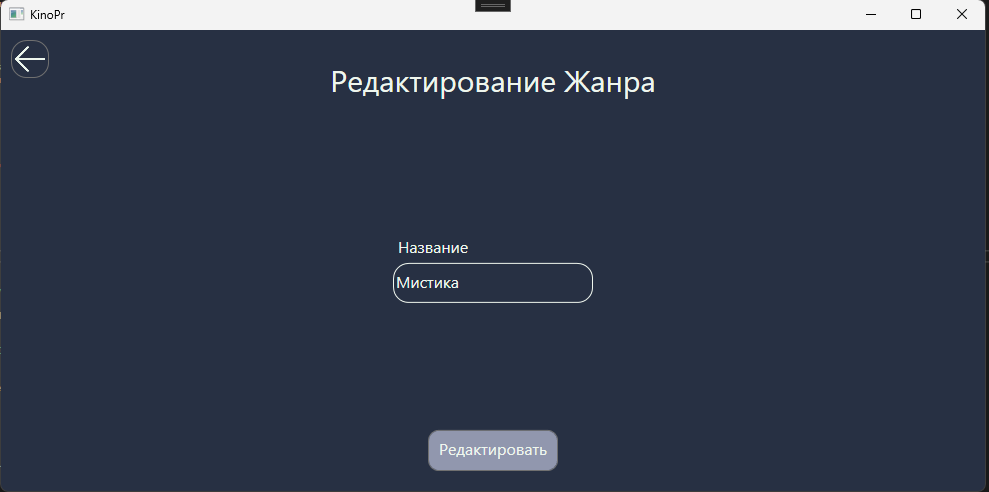


Рисунок 30 Готовая страница Редактирование жанра

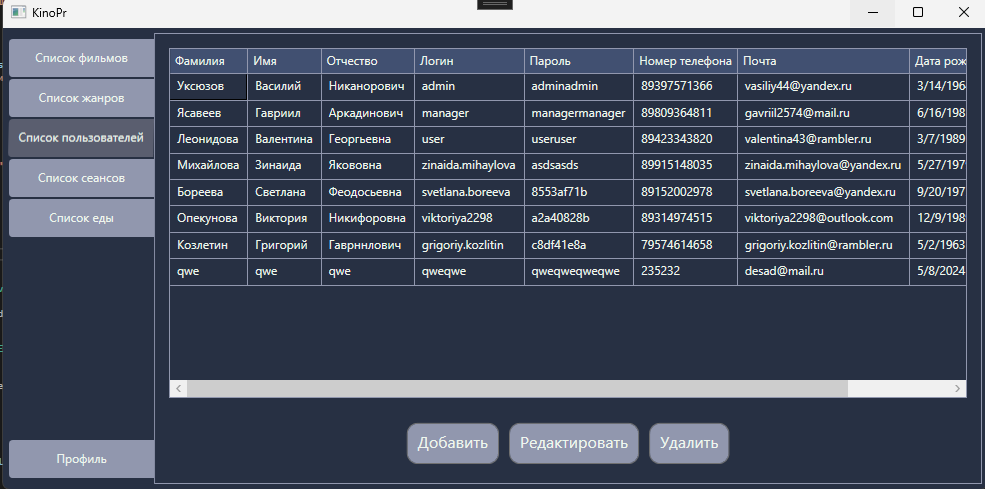


Рисунок 31 Готовая страница Список пользователей

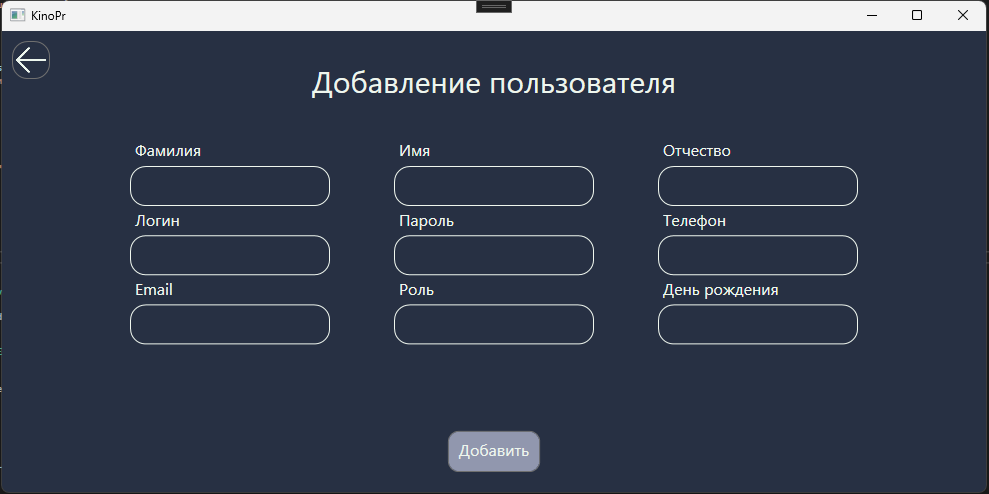


Рисунок 32 Готовая страница Добавление пользователя

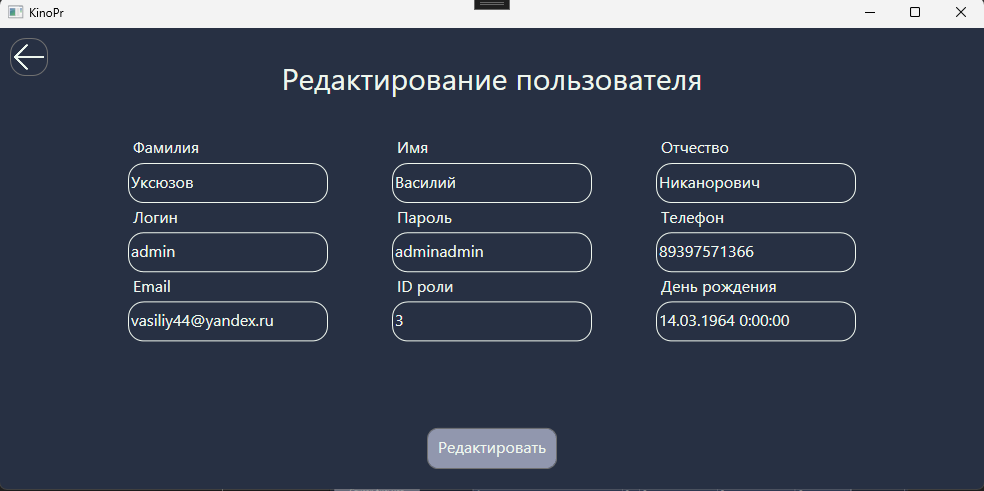


Рисунок 33 Готовая страница Редактирование пользователя



Рисунок 34 Готовая страница Список сеансов

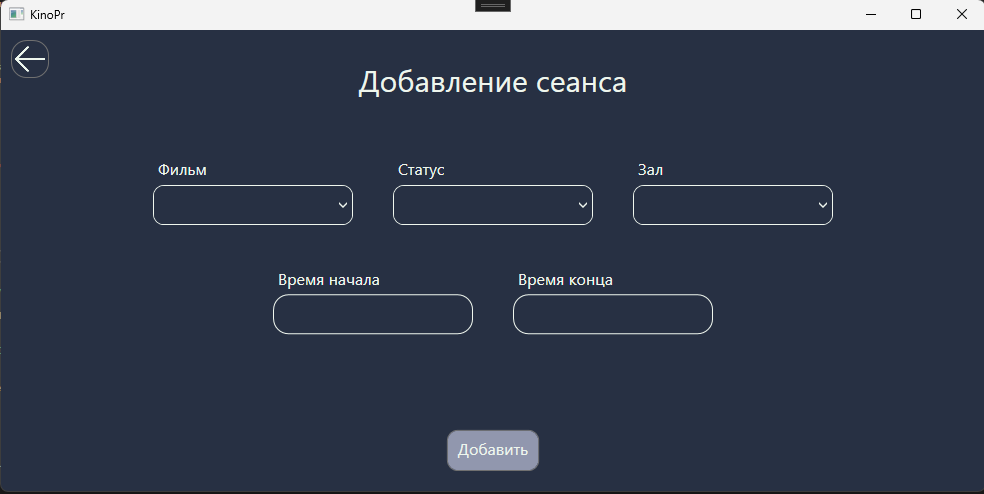


Рисунок 35 Готовая страница Добавление сеанса

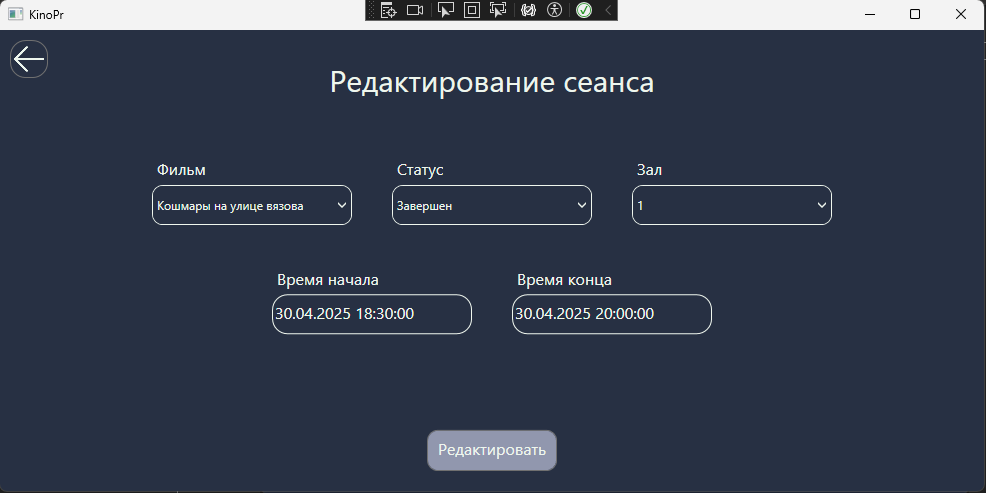


Рисунок 36 Готовая страница Редактирование сеанса

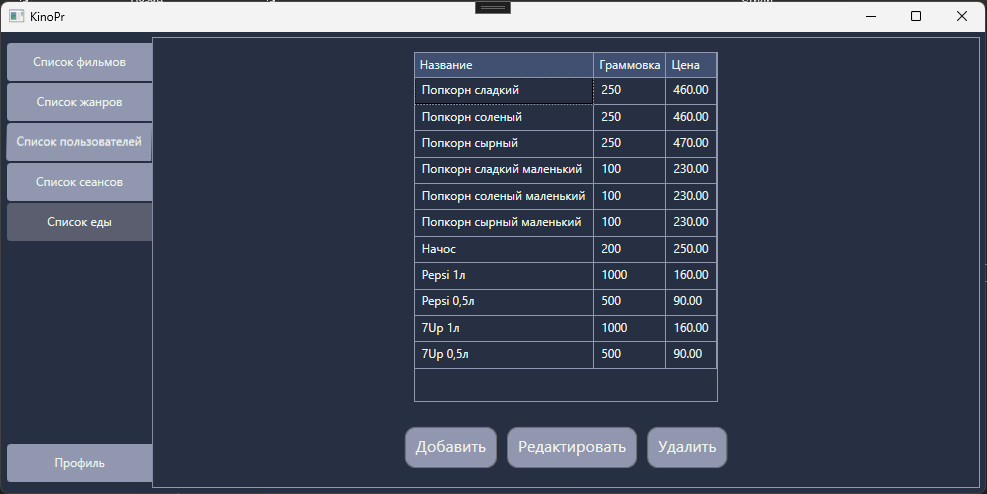


Рисунок 37 Готовая страница Список еды

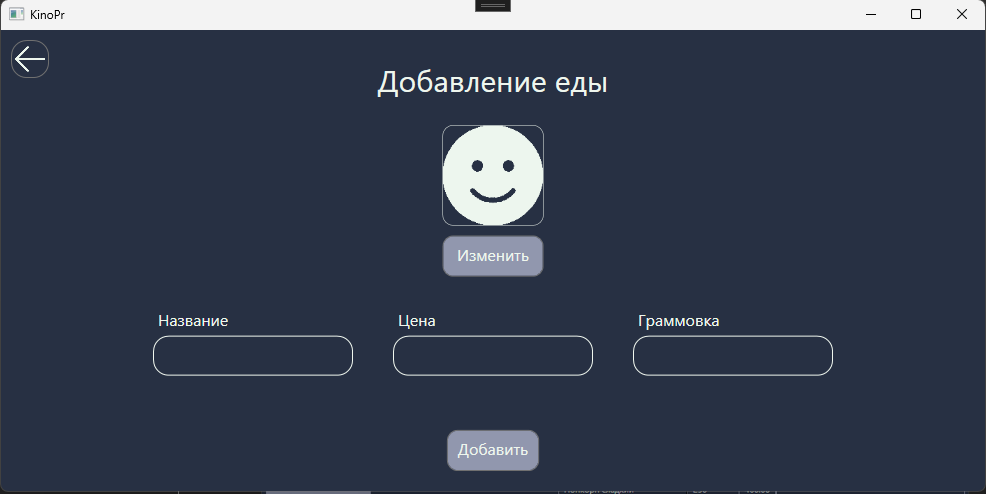


Рисунок 38 Готовая страница Добавление еды

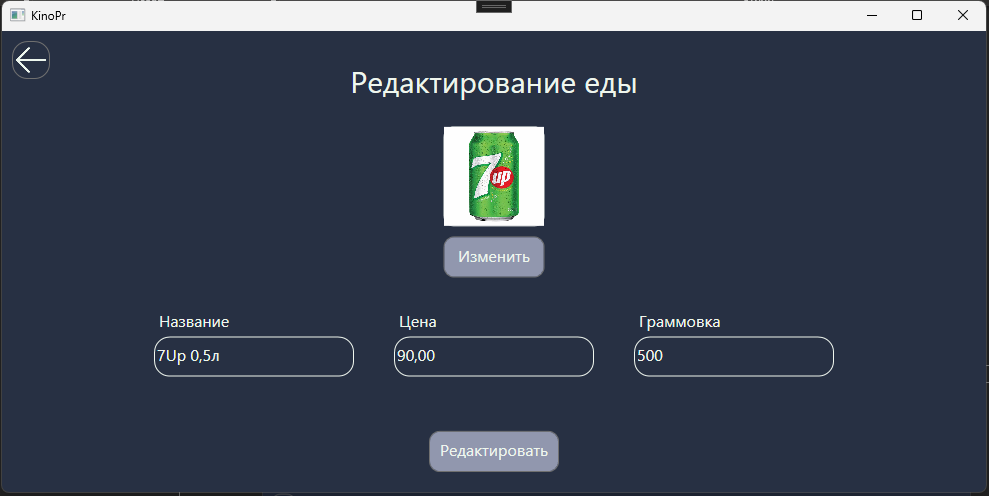


Рисунок 39 Готовая страница Редактирование еды

# ТЕСТИРОВАНИЕ

После разработки приложения были проведены следующие тесты (Рис. 40).

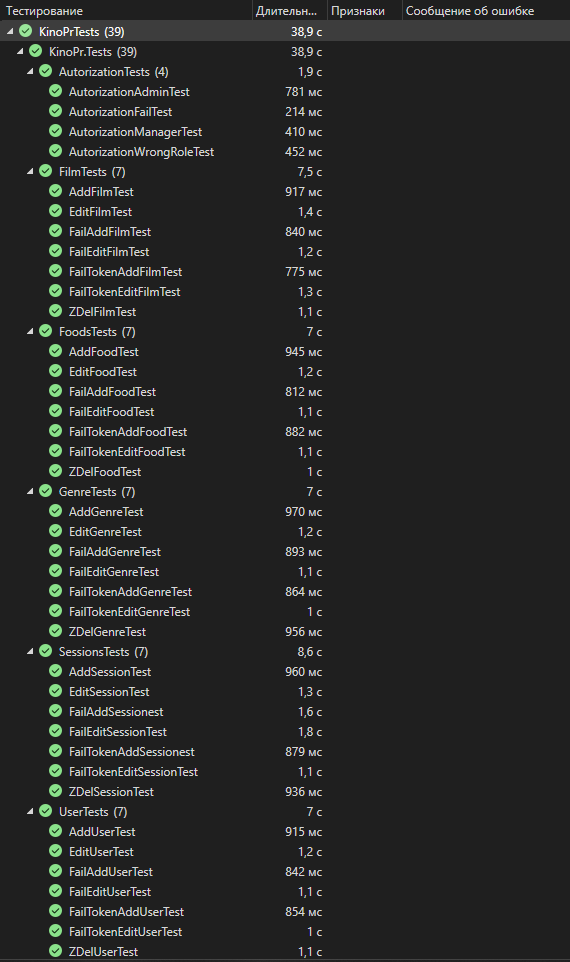


Рисунок 40 Тесты приложения

## Тестирование авторизации

* + 1. Авторизация админа

Листинг 1

[TestMethod()]

public async Task AutorizationAdminTest()

{

string BaseUrl = "http://motov-ae.tepk-it.ru/api/login";

string login = "admin";

string password = "adminadmin";

int actual = 0;

int expected = 200;

int roleac = 0;

int roleex = 3;

using (HttpClient client = new HttpClient())

{

var parameters = new Dictionary<string, string>

{

{ "login", login },

{ "password", password }

};

string queryString = string.Join("&", parameters.Select(x => $"{x.Key}={x.Value}"));

HttpResponseMessage response = await client.PostAsync($"{BaseUrl}?{queryString}", null);

if (response.IsSuccessStatusCode)

{

string responseContent = await response.Content.ReadAsStringAsync();

JObject responseData = JObject.Parse(responseContent);

int roleId = (int)responseData["data"]["role\_id"];

string token = (string)responseData["data"]["api\_token"];

Data.token = token;

roleac = roleId;

}

actual = (int)response.StatusCode;

}

Assert.AreEqual(expected, actual);

Assert.AreEqual(roleex, roleac);

}

* + 1. Авторизация менеджера

Листинг 2

[TestMethod()]

public async Task AutorizationManagerTest()

{

string BaseUrl = "http://motov-ae.tepk-it.ru/api/login";

string login = "manager";

string password = "managermanager";

int actual = 0;

int expected = 200;

int roleac = 0;

int roleex = 2;

using (HttpClient client = new HttpClient())

{

var parameters = new Dictionary<string, string>

{

{ "login", login },

{ "password", password }

};

string queryString = string.Join("&", parameters.Select(x => $"{x.Key}={x.Value}"));

HttpResponseMessage response = await client.PostAsync($"{BaseUrl}?{queryString}", null);

if (response.IsSuccessStatusCode)

{

string responseContent = await response.Content.ReadAsStringAsync();

JObject responseData = JObject.Parse(responseContent);

int roleId = (int)responseData["data"]["role\_id"];

roleac = roleId;

}

actual = (int)response.StatusCode;

}

Assert.AreEqual(expected, actual);

Assert.AreEqual(roleex, roleac);

}

* + 1. Авторизация с неправильными данными

Листинг 3

[TestMethod()]

public async Task AutorizationWrongRoleTest()

{

string BaseUrl = "http://motov-ae.tepk-it.ru/api/login";

string login = "user";

string password = "useruser";

int actual = 0;

int expected = 200;

int roleac = 0;

int roleex = 1;

using (HttpClient client = new HttpClient())

{

var parameters = new Dictionary<string, string>

{

{ "login", login },

{ "password", password }

};

string queryString = string.Join("&", parameters.Select(x => $"{x.Key}={x.Value}"));

HttpResponseMessage response = await client.PostAsync($"{BaseUrl}?{queryString}", null);

if (response.IsSuccessStatusCode)

{

string responseContent = await response.Content.ReadAsStringAsync();

JObject responseData = JObject.Parse(responseContent);

int roleId = (int)responseData["data"]["role\_id"];

roleac = roleId;

}

actual = (int)response.StatusCode;

}

Assert.AreEqual(expected, actual);

Assert.AreEqual(roleex, roleac);

}

* + 1. Авторизация под пользователя с другой ролью

Листинг 4

[TestMethod()]

public async Task AutorizationFailTest()

{

string BaseUrl = "http://motov-ae.tepk-it.ru/api/login";

string login = "da";

string password = "da";

int actual = 0;

int expected = 401;

int roleac = 0;

int roleex = 0;

using (HttpClient client = new HttpClient())

{

var parameters = new Dictionary<string, string>

{

{ "login", login },

{ "password", password }

};

string queryString = string.Join("&", parameters.Select(x => $"{x.Key}={x.Value}"));

HttpResponseMessage response = await client.PostAsync($"{BaseUrl}?{queryString}", null);

if (response.IsSuccessStatusCode)

{

string responseContent = await response.Content.ReadAsStringAsync();

JObject responseData = JObject.Parse(responseContent);

int roleId = (int)responseData["data"]["role\_id"];

roleac = roleId;

}

actual = (int)response.StatusCode;

}

Assert.AreEqual(expected, actual);

Assert.AreEqual(roleex, roleac);

}

## Тестирование CRUD операций для фильмов

1. Создание фильма

Листинг 5

[TestMethod()]

public async Task AddFilmTest()

{

string BaseUrl = "http://motov-ae.tepk-it.ru/api/login";

string login = "admin";

string password = "adminadmin";

int actual = 0;

int expected = 201;

string Name = "джокер";

int GenreId = 1;

string Duration = "1 ч 30 мин";

int Year = 2005;

string Description = "Джокер плахой";

string Director = "Тимофеев Александр";

string Country = "США";

using (HttpClient client = new HttpClient())

{

var parameters = new Dictionary<string, string>

{

{ "login", login },

{ "password", password }

};

string queryString = string.Join("&", parameters.Select(x => $"{x.Key}={x.Value}"));

HttpResponseMessage response = await client.PostAsync($"{BaseUrl}?{queryString}", null);

if (response.IsSuccessStatusCode)

{

string responseContent = await response.Content.ReadAsStringAsync();

JObject responseData = JObject.Parse(responseContent);

string token = (string)responseData["data"]["api\_token"];

Data.token = token;

}

}

Movie newMovie = new Movie

{

Name = Name,

GenreId = GenreId,

Duration = Duration,

Year = Year,

Description = Description,

Director = Director,

Country = Country

};

using (HttpClient client = new HttpClient())

{

client.DefaultRequestHeaders.Authorization = new AuthenticationHeaderValue("Bearer", Data.token);

MultipartFormDataContent multiContent = new MultipartFormDataContent();

multiContent.Add(new StringContent(newMovie.Name), "name");

multiContent.Add(new StringContent(newMovie.Duration), "duration");

multiContent.Add(new StringContent(newMovie.Description), "description");

multiContent.Add(new StringContent(newMovie.Year.ToString()), "year");

multiContent.Add(new StringContent(newMovie.Country), "country");

multiContent.Add(new StringContent(newMovie.Director), "director");

multiContent.Add(new StringContent(newMovie.GenreId.ToString()), "genre\_id");

HttpResponseMessage response = await client.PostAsync("http://motov-ae.tepk-it.ru/api/film", multiContent);

actual = (int)response.StatusCode;

}

Assert.AreEqual(expected, actual);

}

1. Изменение фильма

Листинг 6

[TestMethod()]

public async Task EditFilmTest()

{

string BaseUrl = "http://motov-ae.tepk-it.ru/api/login";

string login = "admin";

string password = "adminadmin";

int actual = 0;

int expected = 200;

int filmid = 0;

using (HttpClient client = new HttpClient())

{

var parameters = new Dictionary<string, string>

{

{ "login", login },

{ "password", password }

};

string queryString = string.Join("&", parameters.Select(x => $"{x.Key}={x.Value}"));

HttpResponseMessage response = await client.PostAsync($"{BaseUrl}?{queryString}", null);

if (response.IsSuccessStatusCode)

{

string responseContent = await response.Content.ReadAsStringAsync();

JObject responseData = JObject.Parse(responseContent);

string token = (string)responseData["data"]["api\_token"];

Data.token = token;

}

}

using (HttpClient client = new HttpClient())

{

client.DefaultRequestHeaders.Authorization = new AuthenticationHeaderValue("Bearer", Data.token);

HttpResponseMessage response = await client.GetAsync("http://motov-ae.tepk-it.ru/api/film");

if (response.IsSuccessStatusCode)

{

string responseBody = await response.Content.ReadAsStringAsync();

List<Movie> movies = JsonConvert.DeserializeObject<List<Movie>>(responseBody);

Movie nfilm = movies.FirstOrDefault(f => f.Name == "джокер");

filmid = nfilm.Id;

}

}

Movie updatedMovie = new Movie

{

Name = "Джокер",

GenreId = 2,

Duration = "1 ч 31мин",

Year = 2000,

Description = "Джокер хороший",

Director = "Я",

Country = "Россия"

};

using (HttpClient client = new HttpClient())

{

client.DefaultRequestHeaders.Authorization = new AuthenticationHeaderValue("Bearer", Data.token);

MultipartFormDataContent multiContent = new MultipartFormDataContent();

multiContent.Add(new StringContent(updatedMovie.Name), "name");

multiContent.Add(new StringContent(updatedMovie.Duration), "duration");

multiContent.Add(new StringContent(updatedMovie.Description), "description");

multiContent.Add(new StringContent(updatedMovie.Year.ToString()), "year");

multiContent.Add(new StringContent(updatedMovie.Country), "country");

multiContent.Add(new StringContent(updatedMovie.Director), "director");

multiContent.Add(new StringContent(updatedMovie.GenreId.ToString()), "genre\_id");

HttpResponseMessage response = await client.PostAsync($"http://motov-ae.tepk-it.ru/api/film/{filmid}", multiContent);

actual = (int)response.StatusCode;

}

Assert.AreEqual(expected, actual);

}

1. Создание фильма с неверными данными

Листинг 7

[TestMethod()]

public async Task FailAddFilmTest()

{

string BaseUrl = "http://motov-ae.tepk-it.ru/api/login";

string login = "admin";

string password = "adminadmin";

int actual = 0;

int expected = 422;

string Name = "";

int GenreId = 0;

string Duration = "";

int Year = 0;

string Description = "";

string Director = "";

string Country = "";

using (HttpClient client = new HttpClient())

{

var parameters = new Dictionary<string, string>

{

{ "login", login },

{ "password", password }

};

string queryString = string.Join("&", parameters.Select(x => $"{x.Key}={x.Value}"));

HttpResponseMessage response = await client.PostAsync($"{BaseUrl}?{queryString}", null);

if (response.IsSuccessStatusCode)

{

string responseContent = await response.Content.ReadAsStringAsync();

JObject responseData = JObject.Parse(responseContent);

string token = (string)responseData["data"]["api\_token"];

Data.token = token;

}

}

Movie newMovie = new Movie

{

Name = Name,

GenreId = GenreId,

Duration = Duration,

Year = Year,

Description = Description,

Director = Director,

Country = Country

};

using (HttpClient client = new HttpClient())

{

client.DefaultRequestHeaders.Authorization = new AuthenticationHeaderValue("Bearer", Data.token);

MultipartFormDataContent multiContent = new MultipartFormDataContent();

multiContent.Add(new StringContent(newMovie.Name), "name");

multiContent.Add(new StringContent(newMovie.Duration), "duration");

multiContent.Add(new StringContent(newMovie.Description), "description");

multiContent.Add(new StringContent(newMovie.Year.ToString()), "year");

multiContent.Add(new StringContent(newMovie.Country), "country");

multiContent.Add(new StringContent(newMovie.Director), "director");

multiContent.Add(new StringContent(newMovie.GenreId.ToString()), "genre\_id");

HttpResponseMessage response = await client.PostAsync("http://motov-ae.tepk-it.ru/api/film", multiContent);

actual = (int)response.StatusCode;

}

Assert.AreEqual(expected, actual);

}

1. Изменение фильма с неверными данными

Листинг 8

[TestMethod()]

public async Task FailEditFilmTest()

{

string BaseUrl = "http://motov-ae.tepk-it.ru/api/login";

string login = "admin";

string password = "adminadmin";

int actual = 0;

int expected = 422;

int filmid = 0;

using (HttpClient client = new HttpClient())

{

var parameters = new Dictionary<string, string>

{

{ "login", login },

{ "password", password }

};

string queryString = string.Join("&", parameters.Select(x => $"{x.Key}={x.Value}"));

HttpResponseMessage response = await client.PostAsync($"{BaseUrl}?{queryString}", null);

if (response.IsSuccessStatusCode)

{

string responseContent = await response.Content.ReadAsStringAsync();

JObject responseData = JObject.Parse(responseContent);

string token = (string)responseData["data"]["api\_token"];

Data.token = token;

}

}

using (HttpClient client = new HttpClient())

{

client.DefaultRequestHeaders.Authorization = new AuthenticationHeaderValue("Bearer", Data.token);

HttpResponseMessage response = await client.GetAsync("http://motov-ae.tepk-it.ru/api/film");

if (response.IsSuccessStatusCode)

{

string responseBody = await response.Content.ReadAsStringAsync();

List<Movie> movies = JsonConvert.DeserializeObject<List<Movie>>(responseBody);

Movie nfilm = movies.FirstOrDefault(f => f.Name == "Джокер");

filmid = nfilm.Id;

}

}

Movie updatedMovie = new Movie

{

Name = "",

GenreId = 0,

Duration = "",

Year = 0,

Description = "",

Director = "",

Country = ""

};

using (HttpClient client = new HttpClient())

{

client.DefaultRequestHeaders.Authorization = new AuthenticationHeaderValue("Bearer", Data.token);

MultipartFormDataContent multiContent = new MultipartFormDataContent();

multiContent.Add(new StringContent(updatedMovie.Name), "name");

multiContent.Add(new StringContent(updatedMovie.Duration), "duration");

multiContent.Add(new StringContent(updatedMovie.Description), "description");

multiContent.Add(new StringContent(updatedMovie.Year.ToString()), "year");

multiContent.Add(new StringContent(updatedMovie.Country), "country");

multiContent.Add(new StringContent(updatedMovie.Director), "director");

multiContent.Add(new StringContent(updatedMovie.GenreId.ToString()), "genre\_id");

HttpResponseMessage response = await client.PostAsync($"http://motov-ae.tepk-it.ru/api/film/{filmid}", multiContent);

actual = (int)response.StatusCode;

}

Assert.AreEqual(expected, actual);

}

1. Создание фильма не авторизированным пользователем

Листинг 9

[TestMethod()]

public async Task FailTokenAddFilmTest()

{

int actual = 0;

int expected = 401;

string Name = "джокер";

int GenreId = 1;

string Duration = "1 ч 30 мин";

int Year = 2005;

string Description = "Джокер плахой";

string Director = "Тимофеев Александр";

string Country = "США";

Movie newMovie = new Movie

{

Name = Name,

GenreId = GenreId,

Duration = Duration,

Year = Year,

Description = Description,

Director = Director,

Country = Country

};

using (HttpClient client = new HttpClient())

{

MultipartFormDataContent multiContent = new MultipartFormDataContent();

multiContent.Add(new StringContent(newMovie.Name), "name");

multiContent.Add(new StringContent(newMovie.Duration), "duration");

multiContent.Add(new StringContent(newMovie.Description), "description");

multiContent.Add(new StringContent(newMovie.Year.ToString()), "year");

multiContent.Add(new StringContent(newMovie.Country), "country");

multiContent.Add(new StringContent(newMovie.Director), "director");

multiContent.Add(new StringContent(newMovie.GenreId.ToString()), "genre\_id");

HttpResponseMessage response = await client.PostAsync("http://motov-ae.tepk-it.ru/api/film", multiContent);

actual = (int)response.StatusCode;

}

Assert.AreEqual(expected, actual);

}

1. Изменение фильма не авторизированным пользователем

Листинг 10

[TestMethod()]

public async Task FailTokenEditFilmTest()

{

int actual = 0;

int expected = 401;

int filmid = 0;

using (HttpClient client = new HttpClient())

{

client.DefaultRequestHeaders.Authorization = new AuthenticationHeaderValue("Bearer", Data.token);

HttpResponseMessage response = await client.GetAsync("http://motov-ae.tepk-it.ru/api/film");

if (response.IsSuccessStatusCode)

{

string responseBody = await response.Content.ReadAsStringAsync();

List<Movie> movies = JsonConvert.DeserializeObject<List<Movie>>(responseBody);

Movie nfilm = movies.FirstOrDefault(f => f.Name == "Джокер");

filmid = nfilm.Id;

}

}

Movie updatedMovie = new Movie

{

Name = "Джокер",

GenreId = 2,

Duration = "1 ч 31мин",

Year = 2000,

Description = "Джокер хороший",

Director = "Я",

Country = "Россия"

};

using (HttpClient client = new HttpClient())

{

MultipartFormDataContent multiContent = new MultipartFormDataContent();

multiContent.Add(new StringContent(updatedMovie.Name), "name");

multiContent.Add(new StringContent(updatedMovie.Duration), "duration");

multiContent.Add(new StringContent(updatedMovie.Description), "description");

multiContent.Add(new StringContent(updatedMovie.Year.ToString()), "year");

multiContent.Add(new StringContent(updatedMovie.Country), "country");

multiContent.Add(new StringContent(updatedMovie.Director), "director");

multiContent.Add(new StringContent(updatedMovie.GenreId.ToString()), "genre\_id");

HttpResponseMessage response = await client.PostAsync($"http://motov-ae.tepk-it.ru/api/film/{filmid}", multiContent);

actual = (int)response.StatusCode;

}

Assert.AreEqual(expected, actual);

}

1. Удаление фильма

Листинг 11

[TestMethod()]

public async Task ZDelFilmTest()

{

string BaseUrl = "http://motov-ae.tepk-it.ru/api/login";

string login = "admin";

string password = "adminadmin";

int actual = 0;

int expected = 200;

int filmid = 0;

using (HttpClient client = new HttpClient())

{

var parameters = new Dictionary<string, string>

{

{ "login", login },

{ "password", password }

};

string queryString = string.Join("&", parameters.Select(x => $"{x.Key}={x.Value}"));

HttpResponseMessage response = await client.PostAsync($"{BaseUrl}?{queryString}", null);

if (response.IsSuccessStatusCode)

{

string responseContent = await response.Content.ReadAsStringAsync();

JObject responseData = JObject.Parse(responseContent);

string token = (string)responseData["data"]["api\_token"];

Data.token = token;

}

}

using (HttpClient client = new HttpClient())

{

client.DefaultRequestHeaders.Authorization = new AuthenticationHeaderValue("Bearer", Data.token);

HttpResponseMessage response = await client.GetAsync("http://motov-ae.tepk-it.ru/api/film");

if (response.IsSuccessStatusCode)

{

string responseBody = await response.Content.ReadAsStringAsync();

List<Movie> movies = JsonConvert.DeserializeObject<List<Movie>>(responseBody);

Movie nfilm = movies.FirstOrDefault(f => f.Name == "Джокер");

filmid = nfilm.Id;

}

}

using (HttpClient client = new HttpClient())

{

client.DefaultRequestHeaders.Authorization = new AuthenticationHeaderValue("Bearer", Data.token);

HttpResponseMessage response = await client.DeleteAsync($"http://motov-ae.tepk-it.ru/api/film/{filmid}");

actual = (int)response.StatusCode;

}

Assert.AreEqual(expected, actual);

}

## Тестирование CRUD операций для еды

1. Создание еды

Листинг 12

[TestMethod()]

public async Task AddFoodTest()

{

string BaseUrl = "http://motov-ae.tepk-it.ru/api/login";

string login = "manager";

string password = "managermanager";

int actual = 0;

int expected = 201;

string name = "Еда";

decimal price = Convert.ToDecimal( "166,45");

int mass = 150;

using (HttpClient client = new HttpClient())

{

var parameters = new Dictionary<string, string>

{

{ "login", login },

{ "password", password }

};

string queryString = string.Join("&", parameters.Select(x => $"{x.Key}={x.Value}"));

HttpResponseMessage response = await client.PostAsync($"{BaseUrl}?{queryString}", null);

if (response.IsSuccessStatusCode)

{

string responseContent = await response.Content.ReadAsStringAsync();

JObject responseData = JObject.Parse(responseContent);

string token = (string)responseData["data"]["api\_token"];

Data.token = token;

}

}

Product newProduct = new Product

{

Name = name,

Price = price,

Mass = mass

};

using (HttpClient client = new HttpClient())

{

client.DefaultRequestHeaders.Authorization = new AuthenticationHeaderValue("Bearer", Data.token);

MultipartFormDataContent multiContent = new MultipartFormDataContent();

multiContent.Add(new StringContent(newProduct.Name), "name");

multiContent.Add(new StringContent(newProduct.Price.ToString(CultureInfo.InvariantCulture)), "price");

multiContent.Add(new StringContent(newProduct.Mass.ToString()), "mass");

HttpResponseMessage response = await client.PostAsync("http://motov-ae.tepk-it.ru/api/product", multiContent);

actual = (int)response.StatusCode;

}

Assert.AreEqual(expected, actual);

}

1. Изменение еды

Листинг 13

[TestMethod()]

public async Task EditFoodTest()

{

string BaseUrl = "http://motov-ae.tepk-it.ru/api/login";

string login = "manager";

string password = "managermanager";

int actual = 0;

int expected = 200;

int productid = 0;

string name = "да";

decimal price = Convert.ToDecimal("166,45");

int mass = 150;

using (HttpClient client = new HttpClient())

{

var parameters = new Dictionary<string, string>

{

{ "login", login },

{ "password", password }

};

string queryString = string.Join("&", parameters.Select(x => $"{x.Key}={x.Value}"));

HttpResponseMessage response = await client.PostAsync($"{BaseUrl}?{queryString}", null);

if (response.IsSuccessStatusCode)

{

string responseContent = await response.Content.ReadAsStringAsync();

JObject responseData = JObject.Parse(responseContent);

string token = (string)responseData["data"]["api\_token"];

Data.token = token;

}

}

using (HttpClient client = new HttpClient())

{

client.DefaultRequestHeaders.Authorization = new AuthenticationHeaderValue("Bearer", Data.token);

HttpResponseMessage response = await client.GetAsync("http://motov-ae.tepk-it.ru/api/product");

if (response.IsSuccessStatusCode)

{

string responseBody = await response.Content.ReadAsStringAsync();

List<Product> products = JsonConvert.DeserializeObject<List<Product>>(responseBody);

Product nproducts = products.FirstOrDefault(f => f.Name == "Еда");

productid = nproducts.Id;

}

}

Product updatedProduct = new Product

{

Name = name,

Price = price,

Mass = mass

};

using (HttpClient client = new HttpClient())

{

client.DefaultRequestHeaders.Authorization = new AuthenticationHeaderValue("Bearer", Data.token);

MultipartFormDataContent multiContent = new MultipartFormDataContent();

multiContent.Add(new StringContent(updatedProduct.Name), "name");

multiContent.Add(new StringContent(updatedProduct.Price.ToString(CultureInfo.InvariantCulture)), "price");

multiContent.Add(new StringContent(updatedProduct.Mass.ToString()), "mass");

HttpResponseMessage response = await client.PostAsync($"http://motov-ae.tepk-it.ru/api/product/{productid}", multiContent);

actual =(int)response.StatusCode;

}

Assert.AreEqual(expected, actual);

}

1. Создание еды с неверными данными

Листинг 14

[TestMethod()]

public async Task FailAddFoodTest()

{

string BaseUrl = "http://motov-ae.tepk-it.ru/api/login";

string login = "manager";

string password = "managermanager";

int actual = 0;

int expected = 422;

string name = "";

decimal price = Convert.ToDecimal( "0,6");

int mass = 0;

using (HttpClient client = new HttpClient())

{

var parameters = new Dictionary<string, string>

{

{ "login", login },

{ "password", password }

};

string queryString = string.Join("&", parameters.Select(x => $"{x.Key}={x.Value}"));

HttpResponseMessage response = await client.PostAsync($"{BaseUrl}?{queryString}", null);

if (response.IsSuccessStatusCode)

{

string responseContent = await response.Content.ReadAsStringAsync();

JObject responseData = JObject.Parse(responseContent);

string token = (string)responseData["data"]["api\_token"];

Data.token = token;

}

}

Product newProduct = new Product

{

Name = name,

Price = price,

Mass = mass

};

using (HttpClient client = new HttpClient())

{

client.DefaultRequestHeaders.Authorization = new AuthenticationHeaderValue("Bearer", Data.token);

MultipartFormDataContent multiContent = new MultipartFormDataContent();

multiContent.Add(new StringContent(newProduct.Name), "name");

multiContent.Add(new StringContent(newProduct.Price.ToString(CultureInfo.InvariantCulture)), "price");

multiContent.Add(new StringContent(newProduct.Mass.ToString()), "mass");

HttpResponseMessage response = await client.PostAsync("http://motov-ae.tepk-it.ru/api/product", multiContent);

actual = (int)response.StatusCode;

}

Assert.AreEqual(expected, actual);

}

1. Изменение еды с неверными данными

Листинг 15

[TestMethod()]

public async Task FailEditFoodTest()

{

string BaseUrl = "http://motov-ae.tepk-it.ru/api/login";

string login = "manager";

string password = "managermanager";

int actual = 0;

int expected = 422;

int productid = 0;

string name = "";

decimal price = Convert.ToDecimal("15,2");

int mass = 10;

using (HttpClient client = new HttpClient())

{

var parameters = new Dictionary<string, string>

{

{ "login", login },

{ "password", password }

};

string queryString = string.Join("&", parameters.Select(x => $"{x.Key}={x.Value}"));

HttpResponseMessage response = await client.PostAsync($"{BaseUrl}?{queryString}", null);

if (response.IsSuccessStatusCode)

{

string responseContent = await response.Content.ReadAsStringAsync();

JObject responseData = JObject.Parse(responseContent);

string token = (string)responseData["data"]["api\_token"];

Data.token = token;

}

}

using (HttpClient client = new HttpClient())

{

client.DefaultRequestHeaders.Authorization = new AuthenticationHeaderValue("Bearer", Data.token);

HttpResponseMessage response = await client.GetAsync("http://motov-ae.tepk-it.ru/api/product");

if (response.IsSuccessStatusCode)

{

string responseBody = await response.Content.ReadAsStringAsync();

List<Product> products = JsonConvert.DeserializeObject<List<Product>>(responseBody);

Product nproducts = products.FirstOrDefault(f => f.Name == "да");

productid = nproducts.Id;

}

}

Product updatedProduct = new Product

{

Name = name,

Price = price,

Mass = mass

};

using (HttpClient client = new HttpClient())

{

client.DefaultRequestHeaders.Authorization = new AuthenticationHeaderValue("Bearer", Data.token);

MultipartFormDataContent multiContent = new MultipartFormDataContent();

multiContent.Add(new StringContent(updatedProduct.Name), "name");

multiContent.Add(new StringContent(updatedProduct.Price.ToString(CultureInfo.InvariantCulture)), "price");

multiContent.Add(new StringContent(updatedProduct.Mass.ToString()), "mass");

HttpResponseMessage response = await client.PostAsync($"http://motov-ae.tepk-it.ru/api/product/{productid}", multiContent);

actual =(int)response.StatusCode;

}

Assert.AreEqual(expected, actual);

}

1. Создание еды не авторизированным пользователем

Листинг 16

[TestMethod()]

public async Task FailTokenAddFoodTest()

{

int actual = 0;

int expected = 401;

string name = "Еда";

decimal price = Convert.ToDecimal( "166,45");

int mass = 150;

Product newProduct = new Product

{

Name = name,

Price = price,

Mass = mass

};

using (HttpClient client = new HttpClient())

{

MultipartFormDataContent multiContent = new MultipartFormDataContent();

multiContent.Add(new StringContent(newProduct.Name), "name");

multiContent.Add(new StringContent(newProduct.Price.ToString(CultureInfo.InvariantCulture)), "price");

multiContent.Add(new StringContent(newProduct.Mass.ToString()), "mass");

HttpResponseMessage response = await client.PostAsync("http://motov-ae.tepk-it.ru/api/product", multiContent);

actual = (int)response.StatusCode;

}

Assert.AreEqual(expected, actual);

}

1. Изменение еды не авторизированным пользователем

Листинг 17

[TestMethod()]

public async Task FailTokenEditFoodTest()

{

int actual = 0;

int expected = 401;

int productid = 0;

string name = "Eда";

decimal price = Convert.ToDecimal("166,45");

int mass = 150;

using (HttpClient client = new HttpClient())

{

client.DefaultRequestHeaders.Authorization = new AuthenticationHeaderValue("Bearer", Data.token);

HttpResponseMessage response = await client.GetAsync("http://motov-ae.tepk-it.ru/api/product");

if (response.IsSuccessStatusCode)

{

string responseBody = await response.Content.ReadAsStringAsync();

List<Product> products = JsonConvert.DeserializeObject<List<Product>>(responseBody);

Product nproducts = products.FirstOrDefault(f => f.Name == "да");

productid = nproducts.Id;

}

}

Product updatedProduct = new Product

{

Name = name,

Price = price,

Mass = mass

};

using (HttpClient client = new HttpClient())

{

MultipartFormDataContent multiContent = new MultipartFormDataContent();

multiContent.Add(new StringContent(updatedProduct.Name), "name");

multiContent.Add(new StringContent(updatedProduct.Price.ToString(CultureInfo.InvariantCulture)), "price");

multiContent.Add(new StringContent(updatedProduct.Mass.ToString()), "mass");

HttpResponseMessage response = await client.PostAsync($"http://motov-ae.tepk-it.ru/api/product/{productid}", multiContent);

actual =(int)response.StatusCode;

}

Assert.AreEqual(expected, actual);

}

1. Удаление еды

Листинг 18

[TestMethod()]

public async Task ZDelFoodTest()

{

string BaseUrl = "http://motov-ae.tepk-it.ru/api/login";

string login = "manager";

string password = "managermanager";

int actual = 0;

int expected = 200;

int productid = 0;

using (HttpClient client = new HttpClient())

{

var parameters = new Dictionary<string, string>

{

{ "login", login },

{ "password", password }

};

string queryString = string.Join("&", parameters.Select(x => $"{x.Key}={x.Value}"));

HttpResponseMessage response = await client.PostAsync($"{BaseUrl}?{queryString}", null);

if (response.IsSuccessStatusCode)

{

string responseContent = await response.Content.ReadAsStringAsync();

JObject responseData = JObject.Parse(responseContent);

string token = (string)responseData["data"]["api\_token"];

Data.token = token;

}

}

using (HttpClient client = new HttpClient())

{

client.DefaultRequestHeaders.Authorization = new AuthenticationHeaderValue("Bearer", Data.token);

HttpResponseMessage response = await client.GetAsync("http://motov-ae.tepk-it.ru/api/product");

if (response.IsSuccessStatusCode)

{

string responseBody = await response.Content.ReadAsStringAsync();

List<Product> products = JsonConvert.DeserializeObject<List<Product>>(responseBody);

Product nproducts = products.FirstOrDefault(f => f.Name == "да");

productid = nproducts.Id;

}

}

using (HttpClient client = new HttpClient())

{

client.DefaultRequestHeaders.Authorization = new AuthenticationHeaderValue("Bearer", Data.token);

HttpResponseMessage response = await client.DeleteAsync($"http://motov-ae.tepk-it.ru/api/product/{productid}");

actual = (int)response.StatusCode;

}

Assert.AreEqual(expected, actual);

}

## Тестирование CRUD операций для жанров

1. Создание жанров

Листинг 19

[TestMethod()]

public async Task AddGenreTest()

{

string BaseUrl = "http://motov-ae.tepk-it.ru/api/login";

string login = "admin";

string password = "adminadmin";

int actual = 0;

int expected = 201;

string genreName = "da";

using (HttpClient client = new HttpClient())

{

var parameters = new Dictionary<string, string>

{

{ "login", login },

{ "password", password }

};

string queryString = string.Join("&", parameters.Select(x => $"{x.Key}={x.Value}"));

HttpResponseMessage response = await client.PostAsync($"{BaseUrl}?{queryString}", null);

if (response.IsSuccessStatusCode)

{

string responseContent = await response.Content.ReadAsStringAsync();

JObject responseData = JObject.Parse(responseContent);

string token = (string)responseData["data"]["api\_token"];

Data.token = token;

}

}

Genre newGenre = new Genre

{

Name = genreName

};

using (HttpClient client = new HttpClient())

{

client.DefaultRequestHeaders.Authorization = new AuthenticationHeaderValue("Bearer", Data.token);

MultipartFormDataContent multiContent = new MultipartFormDataContent();

multiContent.Add(new StringContent(newGenre.Name), "name");

HttpResponseMessage response = await client.PostAsync("http://motov-ae.tepk-it.ru/api/genre", multiContent);

actual = (int)response.StatusCode;

}

Assert.AreEqual(expected, actual);

}

1. Изменение жанров

Листинг 20

[TestMethod()]

public async Task EditGenreTest()

{

string BaseUrl = "http://motov-ae.tepk-it.ru/api/login";

string login = "admin";

string password = "adminadmin";

int actual = 0;

int expected = 200;

int genreid = 0;

using (HttpClient client = new HttpClient())

{

var parameters = new Dictionary<string, string>

{

{ "login", login },

{ "password", password }

};

string queryString = string.Join("&", parameters.Select(x => $"{x.Key}={x.Value}"));

HttpResponseMessage response = await client.PostAsync($"{BaseUrl}?{queryString}", null);

if (response.IsSuccessStatusCode)

{

string responseContent = await response.Content.ReadAsStringAsync();

JObject responseData = JObject.Parse(responseContent);

string token = (string)responseData["data"]["api\_token"];

Data.token = token;

}

}

using (HttpClient client = new HttpClient())

{

client.DefaultRequestHeaders.Authorization = new AuthenticationHeaderValue("Bearer", Data.token);

HttpResponseMessage response = await client.GetAsync("http://motov-ae.tepk-it.ru/api/genre");

if (response.IsSuccessStatusCode)

{

string responseBody = await response.Content.ReadAsStringAsync();

List<Genre> genres = JsonConvert.DeserializeObject<List<Genre>>(responseBody);

Genre daGenre = genres.FirstOrDefault(g => g.Name == "da");

genreid = daGenre.Id;

}

}

Genre updatedGenre = new Genre

{

Id = genreid,

Name = "Da"

};

using (HttpClient client = new HttpClient())

{

client.DefaultRequestHeaders.Authorization = new AuthenticationHeaderValue("Bearer", Data.token);

MultipartFormDataContent multiContent = new MultipartFormDataContent();

multiContent.Add(new StringContent(updatedGenre.Name), "name");

HttpResponseMessage response = await client.PostAsync($"http://motov-ae.tepk-it.ru/api/genre/{genreid}", multiContent);

actual = (int)response.StatusCode;

}

Assert.AreEqual(expected, actual);

}

1. Создание жанров с неверными данными

Листинг 21

[TestMethod()]

public async Task FailAddGenreTest()

{

string BaseUrl = "http://motov-ae.tepk-it.ru/api/login";

string login = "admin";

string password = "adminadmin";

int actual = 0;

int expected = 422;

string genreName = "";

using (HttpClient client = new HttpClient())

{

var parameters = new Dictionary<string, string>

{

{ "login", login },

{ "password", password }

};

string queryString = string.Join("&", parameters.Select(x => $"{x.Key}={x.Value}"));

HttpResponseMessage response = await client.PostAsync($"{BaseUrl}?{queryString}", null);

if (response.IsSuccessStatusCode)

{

string responseContent = await response.Content.ReadAsStringAsync();

JObject responseData = JObject.Parse(responseContent);

string token = (string)responseData["data"]["api\_token"];

Data.token = token;

}

}

Genre newGenre = new Genre

{

Name = genreName

};

using (HttpClient client = new HttpClient())

{

client.DefaultRequestHeaders.Authorization = new AuthenticationHeaderValue("Bearer", Data.token);

MultipartFormDataContent multiContent = new MultipartFormDataContent();

multiContent.Add(new StringContent(newGenre.Name), "name");

HttpResponseMessage response = await client.PostAsync("http://motov-ae.tepk-it.ru/api/genre", multiContent);

actual =(int)response.StatusCode;

}

Assert.AreEqual(expected, actual);

}

1. Изменение жанров с неверными данными

Листинг 22

[TestMethod()]

public async Task FailEditGenreTest()

{

string BaseUrl = "http://motov-ae.tepk-it.ru/api/login";

string login = "admin";

string password = "adminadmin";

int actual = 0;

int expected = 422;

int genreid = 0;

using (HttpClient client = new HttpClient())

{

var parameters = new Dictionary<string, string>

{

{ "login", login },

{ "password", password }

};

string queryString = string.Join("&", parameters.Select(x => $"{x.Key}={x.Value}"));

HttpResponseMessage response = await client.PostAsync($"{BaseUrl}?{queryString}", null);

if (response.IsSuccessStatusCode)

{

string responseContent = await response.Content.ReadAsStringAsync();

JObject responseData = JObject.Parse(responseContent);

string token = (string)responseData["data"]["api\_token"];

Data.token = token;

}

}

using (HttpClient client = new HttpClient())

{

client.DefaultRequestHeaders.Authorization = new AuthenticationHeaderValue("Bearer", Data.token);

HttpResponseMessage response = await client.GetAsync("http://motov-ae.tepk-it.ru/api/genre");

if (response.IsSuccessStatusCode)

{

string responseBody = await response.Content.ReadAsStringAsync();

List<Genre> genres = JsonConvert.DeserializeObject<List<Genre>>(responseBody);

Genre daGenre = genres.FirstOrDefault(g => g.Name == "Da");

genreid = daGenre.Id;

}

}

Genre updatedGenre = new Genre

{

Id = genreid,

Name = ""

};

using (HttpClient client = new HttpClient())

{

client.DefaultRequestHeaders.Authorization = new AuthenticationHeaderValue("Bearer", Data.token);

MultipartFormDataContent multiContent = new MultipartFormDataContent();

multiContent.Add(new StringContent(updatedGenre.Name), "name");

HttpResponseMessage response = await client.PostAsync($"http://motov-ae.tepk-it.ru/api/genre/{genreid}", multiContent);

actual = (int)response.StatusCode;

}

Assert.AreEqual(expected, actual);

}

1. Создание жанров не авторизированным пользователем

Листинг 23

[TestMethod()]

public async Task FailTokenAddGenreTest()

{

int actual = 0;

int expected = 401;

string genreName = "da";

Genre newGenre = new Genre

{

Name = genreName

};

using (HttpClient client = new HttpClient())

{

MultipartFormDataContent multiContent = new MultipartFormDataContent();

multiContent.Add(new StringContent(newGenre.Name), "name");

HttpResponseMessage response = await client.PostAsync("http://motov-ae.tepk-it.ru/api/genre", multiContent);

actual =(int)response.StatusCode;

}

Assert.AreEqual(expected, actual);

}

1. Изменение жанров не авторизированным пользователем

Листинг 24

[TestMethod()]

public async Task FailTokenEditGenreTest()

{

int actual = 0;

int expected = 401;

int genreid = 0;

using (HttpClient client = new HttpClient())

{

HttpResponseMessage response = await client.GetAsync("http://motov-ae.tepk-it.ru/api/genre");

if (response.IsSuccessStatusCode)

{

string responseBody = await response.Content.ReadAsStringAsync();

List<Genre> genres = JsonConvert.DeserializeObject<List<Genre>>(responseBody);

Genre daGenre = genres.FirstOrDefault(g => g.Name == "Da");

genreid = daGenre.Id;

}

}

Genre updatedGenre = new Genre

{

Id = genreid,

Name = "Da"

};

using (HttpClient client = new HttpClient())

{

MultipartFormDataContent multiContent = new MultipartFormDataContent();

multiContent.Add(new StringContent(updatedGenre.Name), "name");

HttpResponseMessage response = await client.PostAsync($"http://motov-ae.tepk-it.ru/api/genre/{genreid}", multiContent);

actual = (int)response.StatusCode;

}

Assert.AreEqual(expected, actual);

}

1. Удаление жанров

Листинг 25

[TestMethod()]

public async Task ZDelGenreTest()

{

string BaseUrl = "http://motov-ae.tepk-it.ru/api/login";

string login = "admin";

string password = "adminadmin";

int actual = 0;

int expected = 200;

int genreid = 0;

using (HttpClient client = new HttpClient())

{

var parameters = new Dictionary<string, string>

{

{ "login", login },

{ "password", password }

};

string queryString = string.Join("&", parameters.Select(x => $"{x.Key}={x.Value}"));

HttpResponseMessage response = await client.PostAsync($"{BaseUrl}?{queryString}", null);

if (response.IsSuccessStatusCode)

{

string responseContent = await response.Content.ReadAsStringAsync();

JObject responseData = JObject.Parse(responseContent);

string token = (string)responseData["data"]["api\_token"];

Data.token = token;

}

}

using (HttpClient client = new HttpClient())

{

client.DefaultRequestHeaders.Authorization = new AuthenticationHeaderValue("Bearer", Data.token);

HttpResponseMessage response = await client.GetAsync("http://motov-ae.tepk-it.ru/api/genre");

if (response.IsSuccessStatusCode)

{

string responseBody = await response.Content.ReadAsStringAsync();

List<Genre> genres = JsonConvert.DeserializeObject<List<Genre>>(responseBody);

Genre daGenre = genres.FirstOrDefault(g => g.Name == "Da");

genreid = daGenre.Id;

}

}

using (HttpClient client = new HttpClient())

{

client.DefaultRequestHeaders.Authorization = new AuthenticationHeaderValue("Bearer", Data.token);

HttpResponseMessage response = await client.DeleteAsync($"http://motov-ae.tepk-it.ru/api/genre/{genreid}");

actual = (int)response.StatusCode;

}

Assert.AreEqual(expected, actual);

}

## Тестирование CRUD операций для сеансов

1. Создание сеансов

Листинг 26

1. Изменение сеансов

Листинг 27

[TestMethod()]

public async Task EditSessionTest()

{

string BaseUrl = "http://motov-ae.tepk-it.ru/api/login";

string login = "manager";

string password = "managermanager";

int actual = 0;

int expected = 200;

int sessionid = 0;

DateTime time\_start = DateTime.Parse("2005-5-5 16:0:0");

DateTime time\_end = DateTime.Parse("2005-5-5 16:0:0");

int session\_status\_id = 4;

int FilmId = 2;

int hall = 1;

using (HttpClient client = new HttpClient())

{

var parameters = new Dictionary<string, string>

{

{ "login", login },

{ "password", password }

};

string queryString = string.Join("&", parameters.Select(x => $"{x.Key}={x.Value}"));

HttpResponseMessage response = await client.PostAsync($"{BaseUrl}?{queryString}", null);

if (response.IsSuccessStatusCode)

{

string responseContent = await response.Content.ReadAsStringAsync();

JObject responseData = JObject.Parse(responseContent);

string token = (string)responseData["data"]["api\_token"];

Data.token = token;

}

}

using (HttpClient client = new HttpClient())

{

client.DefaultRequestHeaders.Authorization = new AuthenticationHeaderValue("Bearer", Data.token);

HttpResponseMessage response = await client.GetAsync("http://motov-ae.tepk-it.ru/api/session");

if (response.IsSuccessStatusCode)

{

string responseBody = await response.Content.ReadAsStringAsync();

List<Session> sessions = JsonConvert.DeserializeObject<List<Session>>(responseBody);

Session nsession = sessions.FirstOrDefault(f => f.time\_start == DateTime.Parse("2005-5-5 16:0:0"));

sessionid = nsession.id;

}

}

Session updatedSession = new Session

{

time\_start = time\_start,

time\_end = time\_start,

session\_status\_id = session\_status\_id,

hall = hall,

FilmId = FilmId,

};

using (HttpClient client = new HttpClient())

{

client.DefaultRequestHeaders.Authorization = new AuthenticationHeaderValue("Bearer", Data.token);

MultipartFormDataContent multiContent = new MultipartFormDataContent();

multiContent.Add(new StringContent(updatedSession.time\_start.ToString("yyyy-M-d H:m:s")), "time\_start");

multiContent.Add(new StringContent(updatedSession.time\_end.ToString("yyyy-M-d H:m:s")), "time\_end");

multiContent.Add(new StringContent(updatedSession.session\_status\_id.ToString()), "session\_status\_id");

multiContent.Add(new StringContent(updatedSession.FilmId.ToString()), "film\_id");

multiContent.Add(new StringContent(updatedSession.hall.ToString()), "hall\_id");

HttpResponseMessage response = await client.PostAsync($"http://motov-ae.tepk-it.ru/api/session/{sessionid}", multiContent);

actual = (int)response.StatusCode;

}

Assert.AreEqual(expected, actual);

}

1. Создание сеансов с неверными данными

Листинг 28

[TestMethod()]

public async Task FailAddSessionest()

{

string BaseUrl = "http://motov-ae.tepk-it.ru/api/login";

string login = "manager";

string password = "managermanager";

int actual = 0;

int expected = 500;

DateTime time\_start = DateTime.Parse("2005-5-5 16:0:0");

DateTime time\_end = DateTime.Parse("2005-5-5 16:0:0");

int session\_status\_id = 10;

int FilmId = 10;

int hall = 10;

using (HttpClient client = new HttpClient())

{

var parameters = new Dictionary<string, string>

{

{ "login", login },

{ "password", password }

};

string queryString = string.Join("&", parameters.Select(x => $"{x.Key}={x.Value}"));

HttpResponseMessage response = await client.PostAsync($"{BaseUrl}?{queryString}", null);

if (response.IsSuccessStatusCode)

{

string responseContent = await response.Content.ReadAsStringAsync();

JObject responseData = JObject.Parse(responseContent);

string token = (string)responseData["data"]["api\_token"];

Data.token = token;

}

}

Session newSession = new Session

{

time\_start = time\_start,

time\_end = time\_end,

session\_status\_id = session\_status\_id,

FilmId = FilmId,

hall = hall

};

using (HttpClient client = new HttpClient())

{

client.DefaultRequestHeaders.Authorization = new AuthenticationHeaderValue("Bearer", Data.token);

MultipartFormDataContent multiContent = new MultipartFormDataContent();

multiContent.Add(new StringContent(newSession.time\_start.ToString("yyyy-M-d H:m:s")), "time\_start");

multiContent.Add(new StringContent(newSession.time\_end.ToString("yyyy-MM-dd HH:mm:ss")), "time\_end");

multiContent.Add(new StringContent(newSession.session\_status\_id.ToString()), "session\_status\_id");

multiContent.Add(new StringContent(newSession.FilmId.ToString()), "film\_id");

multiContent.Add(new StringContent(newSession.hall.ToString()), "hall\_id");

HttpResponseMessage response = await client.PostAsync("http://motov-ae.tepk-it.ru/api/session", multiContent);

actual = (int)response.StatusCode;

}

Assert.AreEqual(expected, actual);

}

1. Изменение сеансов с неверными данными

Листинг 29

[TestMethod()]

public async Task FailEditSessionTest()

{

string BaseUrl = "http://motov-ae.tepk-it.ru/api/login";

string login = "manager";

string password = "managermanager";

int actual = 0;

int expected = 500;

int sessionid = 0;

DateTime time\_start = DateTime.Parse("2005-5-5 16:0:0");

DateTime time\_end = DateTime.Parse("2005-5-5 16:0:0");

int session\_status\_id = 10;

int FilmId = 10;

int hall = 10;

using (HttpClient client = new HttpClient())

{

var parameters = new Dictionary<string, string>

{

{ "login", login },

{ "password", password }

};

string queryString = string.Join("&", parameters.Select(x => $"{x.Key}={x.Value}"));

HttpResponseMessage response = await client.PostAsync($"{BaseUrl}?{queryString}", null);

if (response.IsSuccessStatusCode)

{

string responseContent = await response.Content.ReadAsStringAsync();

JObject responseData = JObject.Parse(responseContent);

string token = (string)responseData["data"]["api\_token"];

Data.token = token;

}

}

using (HttpClient client = new HttpClient())

{

client.DefaultRequestHeaders.Authorization = new AuthenticationHeaderValue("Bearer", Data.token);

HttpResponseMessage response = await client.GetAsync("http://motov-ae.tepk-it.ru/api/session");

if (response.IsSuccessStatusCode)

{

string responseBody = await response.Content.ReadAsStringAsync();

List<Session> sessions = JsonConvert.DeserializeObject<List<Session>>(responseBody);

Session nsession = sessions.FirstOrDefault(f => f.time\_start == DateTime.Parse("2005-5-5 16:0:0"));

sessionid = nsession.id;

}

}

Session updatedSession = new Session

{

time\_start = time\_start,

time\_end = time\_start,

session\_status\_id = session\_status\_id,

hall = hall,

FilmId = FilmId,

};

using (HttpClient client = new HttpClient())

{

client.DefaultRequestHeaders.Authorization = new AuthenticationHeaderValue("Bearer", Data.token);

MultipartFormDataContent multiContent = new MultipartFormDataContent();

multiContent.Add(new StringContent(updatedSession.time\_start.ToString("yyyy-M-d H:m:s")), "time\_start");

multiContent.Add(new StringContent(updatedSession.time\_end.ToString("yyyy-M-d H:m:s")), "time\_end");

multiContent.Add(new StringContent(updatedSession.session\_status\_id.ToString()), "session\_status\_id");

multiContent.Add(new StringContent(updatedSession.FilmId.ToString()), "film\_id");

multiContent.Add(new StringContent(updatedSession.hall.ToString()), "hall\_id");

HttpResponseMessage response = await client.PostAsync($"http://motov-ae.tepk-it.ru/api/session/{sessionid}", multiContent);

actual = (int)response.StatusCode;

}

Assert.AreEqual(expected, actual);

}

1. Создание сеансов не авторизированным пользователем

Листинг 30

[TestMethod()]

public async Task FailTokenAddSessionest()

{

int actual = 0;

int expected = 401;

DateTime time\_start = DateTime.Parse("2005-5-5 16:0:0");

DateTime time\_end = DateTime.Parse("2005-5-5 16:0:0");

int session\_status\_id = 1;

int FilmId = 1;

int hall = 1;

Session newSession = new Session

{

time\_start = time\_start,

time\_end = time\_end,

session\_status\_id = session\_status\_id,

FilmId = FilmId,

hall = hall

};

using (HttpClient client = new HttpClient())

{

MultipartFormDataContent multiContent = new MultipartFormDataContent();

multiContent.Add(new StringContent(newSession.time\_start.ToString("yyyy-M-d H:m:s")), "time\_start");

multiContent.Add(new StringContent(newSession.time\_end.ToString("yyyy-MM-dd HH:mm:ss")), "time\_end");

multiContent.Add(new StringContent(newSession.session\_status\_id.ToString()), "session\_status\_id");

multiContent.Add(new StringContent(newSession.FilmId.ToString()), "film\_id");

multiContent.Add(new StringContent(newSession.hall.ToString()), "hall\_id");

HttpResponseMessage response = await client.PostAsync("http://motov-ae.tepk-it.ru/api/session", multiContent);

actual = (int)response.StatusCode;

}

Assert.AreEqual(expected, actual);

}

1. Изменение сеансов не авторизированным пользователем

Листинг 31

[TestMethod()]

public async Task FailTokenEditSessionTest()

{

int actual = 0;

int expected = 401;

int sessionid = 0;

DateTime time\_start = DateTime.Parse("2005-5-5 16:0:0");

DateTime time\_end = DateTime.Parse("2005-5-5 16:0:0");

int session\_status\_id = 4;

int FilmId = 2;

int hall = 1;

using (HttpClient client = new HttpClient())

{

client.DefaultRequestHeaders.Authorization = new AuthenticationHeaderValue("Bearer", Data.token);

HttpResponseMessage response = await client.GetAsync("http://motov-ae.tepk-it.ru/api/session");

if (response.IsSuccessStatusCode)

{

string responseBody = await response.Content.ReadAsStringAsync();

List<Session> sessions = JsonConvert.DeserializeObject<List<Session>>(responseBody);

Session nsession = sessions.FirstOrDefault(f => f.time\_start == DateTime.Parse("2005-5-5 16:0:0"));

sessionid = nsession.id;

}

}

Session updatedSession = new Session

{

time\_start = time\_start,

time\_end = time\_start,

session\_status\_id = session\_status\_id,

hall = hall,

FilmId = FilmId,

};

using (HttpClient client = new HttpClient())

{

MultipartFormDataContent multiContent = new MultipartFormDataContent();

multiContent.Add(new StringContent(updatedSession.time\_start.ToString("yyyy-M-d H:m:s")), "time\_start");

multiContent.Add(new StringContent(updatedSession.time\_end.ToString("yyyy-M-d H:m:s")), "time\_end");

multiContent.Add(new StringContent(updatedSession.session\_status\_id.ToString()), "session\_status\_id");

multiContent.Add(new StringContent(updatedSession.FilmId.ToString()), "film\_id");

multiContent.Add(new StringContent(updatedSession.hall.ToString()), "hall\_id");

HttpResponseMessage response = await client.PostAsync($"http://motov-ae.tepk-it.ru/api/session/{sessionid}", multiContent);

actual = (int)response.StatusCode;

}

Assert.AreEqual(expected, actual);

}

1. Удаление сеансов

Листинг 32

[TestMethod()]

public async Task FailTokenEditSessionTest()

{

int actual = 0;

int expected = 401;

int sessionid = 0;

DateTime time\_start = DateTime.Parse("2005-5-5 16:0:0");

DateTime time\_end = DateTime.Parse("2005-5-5 16:0:0");

int session\_status\_id = 4;

int FilmId = 2;

int hall = 1;

using (HttpClient client = new HttpClient())

{

client.DefaultRequestHeaders.Authorization = new AuthenticationHeaderValue("Bearer", Data.token);

HttpResponseMessage response = await client.GetAsync("http://motov-ae.tepk-it.ru/api/session");

if (response.IsSuccessStatusCode)

{

string responseBody = await response.Content.ReadAsStringAsync();

List<Session> sessions = JsonConvert.DeserializeObject<List<Session>>(responseBody);

Session nsession = sessions.FirstOrDefault(f => f.time\_start == DateTime.Parse("2005-5-5 16:0:0"));

sessionid = nsession.id;

}

}

Session updatedSession = new Session

{

time\_start = time\_start,

time\_end = time\_start,

session\_status\_id = session\_status\_id,

hall = hall,

FilmId = FilmId,

};

using (HttpClient client = new HttpClient())

{

MultipartFormDataContent multiContent = new MultipartFormDataContent();

multiContent.Add(new StringContent(updatedSession.time\_start.ToString("yyyy-M-d H:m:s")), "time\_start");

multiContent.Add(new StringContent(updatedSession.time\_end.ToString("yyyy-M-d H:m:s")), "time\_end");

multiContent.Add(new StringContent(updatedSession.session\_status\_id.ToString()), "session\_status\_id");

multiContent.Add(new StringContent(updatedSession.FilmId.ToString()), "film\_id");

multiContent.Add(new StringContent(updatedSession.hall.ToString()), "hall\_id");

HttpResponseMessage response = await client.PostAsync($"http://motov-ae.tepk-it.ru/api/session/{sessionid}", multiContent);

actual = (int)response.StatusCode;

}

Assert.AreEqual(expected, actual);

}

## Тестирование CRUD операций для пользователей

1. Создание пользователей

Листинг 33

[TestMethod()]

public async Task AddUserTest()

{

string BaseUrl = "http://motov-ae.tepk-it.ru/api/login";

string login = "admin";

string password = "adminadmin";

int actual = 0;

int expected = 200;

string Surname = "Михайличенко";

string Name = "Елена";

string Patronumic = "Феликсовна";

string PhoneNumber = "89196975141";

DateTime Birh = DateTime.Parse("1981-4-4");

string Login = "elena7735";

string Password = "e75bbf55f";

string Email = "elena7735@hotmail.com";

int Role = 1;

using (HttpClient client = new HttpClient())

{

var parameters = new Dictionary<string, string>

{

{ "login", login },

{ "password", password }

};

string queryString = string.Join("&", parameters.Select(x => $"{x.Key}={x.Value}"));

HttpResponseMessage response = await client.PostAsync($"{BaseUrl}?{queryString}", null);

if (response.IsSuccessStatusCode)

{

string responseContent = await response.Content.ReadAsStringAsync();

JObject responseData = JObject.Parse(responseContent);

string token = (string)responseData["data"]["api\_token"];

Data.token = token;

}

}

User newUser = new User

{

surname = Surname,

name = Name,

patronymic = Patronumic,

phone\_number = PhoneNumber,

birth = Birh,

login = Login,

password = Password,

email = Email,

role\_id = Role

};

using (HttpClient client = new HttpClient())

{

client.DefaultRequestHeaders.Authorization = new AuthenticationHeaderValue("Bearer", Data.token);

MultipartFormDataContent multiContent = new MultipartFormDataContent();

multiContent.Add(new StringContent(newUser.name), "name");

multiContent.Add(new StringContent(newUser.surname), "surname");

multiContent.Add(new StringContent(newUser.patronymic), "patronymic");

multiContent.Add(new StringContent(newUser.phone\_number), "phone\_number");

multiContent.Add(new StringContent(newUser.birth.ToString("yyyy-M-d H:m:s")), "birth");

multiContent.Add(new StringContent(newUser.login), "login");

multiContent.Add(new StringContent(newUser.password), "password");

multiContent.Add(new StringContent(newUser.email), "email");

multiContent.Add(new StringContent(newUser.role\_id.ToString()), "role\_id");

HttpResponseMessage response = await client.PostAsync("http://motov-ae.tepk-it.ru/api/user/create", multiContent);

actual = (int)response.StatusCode;

}

Assert.AreEqual(expected, actual);

}

1. Изменение пользователей

Листинг 34

[TestMethod()]

public async Task EditUserTest()

{

int actual = 0;

int expected = 200;

int userid = 0;

string Surname = "Швардыгула";

string Name = "Герман";

string Patronumic = "Георгиевич";

string PhoneNumber = "89097070101";

DateTime Birh = DateTime.Parse("1999-4-4");

string Login = "german90";

string Password = "ba658cd5f";

string Email = "german90@ya.ru";

int Role = 1;

using (HttpClient client = new HttpClient())

{

client.DefaultRequestHeaders.Authorization = new AuthenticationHeaderValue("Bearer", Data.token);

HttpResponseMessage response = await client.GetAsync("http://motov-ae.tepk-it.ru/api/users");

if (response.IsSuccessStatusCode)

{

string responseBody = await response.Content.ReadAsStringAsync();

List<User> users = JsonConvert.DeserializeObject<List<User>>(responseBody);

User nuser = users.FirstOrDefault(f => f.login == "elena7735");

userid = nuser.id;

}

}

User updatedUser = new User

{

surname = Surname,

name = Name,

patronymic = Patronumic,

phone\_number = PhoneNumber,

birth = Birh,

login = Login,

password = Password,

email = Email,

role\_id = Role

};

using (HttpClient client = new HttpClient())

{

client.DefaultRequestHeaders.Authorization = new AuthenticationHeaderValue("Bearer", Data.token);

MultipartFormDataContent multiContent = new MultipartFormDataContent();

multiContent.Add(new StringContent(updatedUser.name), "name");

multiContent.Add(new StringContent(updatedUser.surname), "surname");

multiContent.Add(new StringContent(updatedUser.patronymic), "patronymic");

multiContent.Add(new StringContent(updatedUser.phone\_number), "phone\_number");

multiContent.Add(new StringContent(updatedUser.birth.ToString("yyyy-M-d H:m:s")), "birth");

multiContent.Add(new StringContent(updatedUser.login), "login");

multiContent.Add(new StringContent(updatedUser.password), "password");

multiContent.Add(new StringContent(updatedUser.email), "email");

multiContent.Add(new StringContent(updatedUser.role\_id.ToString()), "role\_id");

HttpResponseMessage response = await client.PostAsync($"http://motov-ae.tepk-it.ru/api/user/update/{userid}", multiContent);

actual = (int)response.StatusCode;

}

Assert.AreEqual(expected, actual);

}

1. Создание пользователей с неверными данными

Листинг 35

[TestMethod()]

public async Task FailAddUserTest()

{

string BaseUrl = "http://motov-ae.tepk-it.ru/api/login";

string login = "admin";

string password = "adminadmin";

int actual = 0;

int expected = 422;

string Surname = "";

string Name = "";

string Patronumic = "";

string PhoneNumber = "";

DateTime Birh = DateTime.Parse("1981-4-4");

string Login = "";

string Password = "";

string Email = "";

int Role = 1;

using (HttpClient client = new HttpClient())

{

var parameters = new Dictionary<string, string>

{

{ "login", login },

{ "password", password }

};

string queryString = string.Join("&", parameters.Select(x => $"{x.Key}={x.Value}"));

HttpResponseMessage response = await client.PostAsync($"{BaseUrl}?{queryString}", null);

if (response.IsSuccessStatusCode)

{

string responseContent = await response.Content.ReadAsStringAsync();

JObject responseData = JObject.Parse(responseContent);

string token = (string)responseData["data"]["api\_token"];

Data.token = token;

}

}

User newUser = new User

{

surname = Surname,

name = Name,

patronymic = Patronumic,

phone\_number = PhoneNumber,

birth = Birh,

login = Login,

password = Password,

email = Email,

role\_id = Role

};

using (HttpClient client = new HttpClient())

{

client.DefaultRequestHeaders.Authorization = new AuthenticationHeaderValue("Bearer", Data.token);

MultipartFormDataContent multiContent = new MultipartFormDataContent();

multiContent.Add(new StringContent(newUser.name), "name");

multiContent.Add(new StringContent(newUser.surname), "surname");

multiContent.Add(new StringContent(newUser.patronymic), "patronymic");

multiContent.Add(new StringContent(newUser.phone\_number), "phone\_number");

multiContent.Add(new StringContent(newUser.birth.ToString("yyyy-M-d H:m:s")), "birth");

multiContent.Add(new StringContent(newUser.login), "login");

multiContent.Add(new StringContent(newUser.password), "password");

multiContent.Add(new StringContent(newUser.email), "email");

multiContent.Add(new StringContent(newUser.role\_id.ToString()), "role\_id");

HttpResponseMessage response = await client.PostAsync("http://motov-ae.tepk-it.ru/api/user/create", multiContent);

actual = (int)response.StatusCode;

}

Assert.AreEqual(expected, actual);

}

1. Изменение пользователей с неверными данными

Листинг 36

[TestMethod()]

public async Task FailEditUserTest()

{

string BaseUrl = "http://motov-ae.tepk-it.ru/api/login";

string login = "admin";

string password = "adminadmin";

int actual = 0;

int expected = 422;

int userid = 0;

string Surname = "da";

string Name = "da";

string Patronumic = "da";

string PhoneNumber = "da";

string Login = "da";

string Password = "da";

string Email = "da@da.ru";

int Role = 1;

using (HttpClient client = new HttpClient())

{

var parameters = new Dictionary<string, string>

{

{ "login", login },

{ "password", password }

};

string queryString = string.Join("&", parameters.Select(x => $"{x.Key}={x.Value}"));

HttpResponseMessage response = await client.PostAsync($"{BaseUrl}?{queryString}", null);

if (response.IsSuccessStatusCode)

{

string responseContent = await response.Content.ReadAsStringAsync();

JObject responseData = JObject.Parse(responseContent);

string token = (string)responseData["data"]["api\_token"];

Data.token = token;

}

}

using (HttpClient client = new HttpClient())

{

client.DefaultRequestHeaders.Authorization = new AuthenticationHeaderValue("Bearer", Data.token);

HttpResponseMessage response = await client.GetAsync("http://motov-ae.tepk-it.ru/api/users");

if (response.IsSuccessStatusCode)

{

string responseBody = await response.Content.ReadAsStringAsync();

List<User> users = JsonConvert.DeserializeObject<List<User>>(responseBody);

User nuser = users.FirstOrDefault(f => f.login == "german90");

userid = nuser.id;

}

}

User updatedUser = new User

{

surname = Surname,

name = Name,

patronymic = Patronumic,

phone\_number = PhoneNumber,

login = Login,

password = Password,

email = Email,

role\_id = Role

};

using (HttpClient client = new HttpClient())

{

client.DefaultRequestHeaders.Authorization = new AuthenticationHeaderValue("Bearer", Data.token);

MultipartFormDataContent multiContent = new MultipartFormDataContent();

multiContent.Add(new StringContent(updatedUser.name), "name");

multiContent.Add(new StringContent(updatedUser.surname), "surname");

multiContent.Add(new StringContent(updatedUser.patronymic), "patronymic");

multiContent.Add(new StringContent(updatedUser.phone\_number), "phone\_number");

multiContent.Add(new StringContent(updatedUser.login), "login");

multiContent.Add(new StringContent(updatedUser.password), "password");

multiContent.Add(new StringContent(updatedUser.email), "email");

multiContent.Add(new StringContent(updatedUser.role\_id.ToString()), "role\_id");

HttpResponseMessage response = await client.PostAsync($"http://motov-ae.tepk-it.ru/api/user/update/{userid}", multiContent);

actual = (int)response.StatusCode;

}

Assert.AreEqual(expected, actual);

}

1. Создание пользователей не авторизированным пользователем

Листинг 37

[TestMethod()]

public async Task FailTokenAddUserTest()

{

int actual = 0;

int expected = 401;

string Surname = "Михайличенко";

string Name = "Елена";

string Patronumic = "Феликсовна";

string PhoneNumber = "89196975141";

DateTime Birh = DateTime.Parse("1981-4-4");

string Login = "elena7735";

string Password = "e75bbf55f";

string Email = "elena7735@hotmail.com";

int Role = 1; User newUser = new User

{

surname = Surname,

name = Name,

patronymic = Patronumic,

phone\_number = PhoneNumber,

birth = Birh,

login = Login,

password = Password,

email = Email,

role\_id = Role

};

using (HttpClient client = new HttpClient())

{

MultipartFormDataContent multiContent = new MultipartFormDataContent();

multiContent.Add(new StringContent(newUser.name), "name");

multiContent.Add(new StringContent(newUser.surname), "surname");

multiContent.Add(new StringContent(newUser.patronymic), "patronymic");

multiContent.Add(new StringContent(newUser.phone\_number), "phone\_number");

multiContent.Add(new StringContent(newUser.birth.ToString("yyyy-M-d H:m:s")), "birth");

multiContent.Add(new StringContent(newUser.login), "login");

multiContent.Add(new StringContent(newUser.password), "password");

multiContent.Add(new StringContent(newUser.email), "email");

multiContent.Add(new StringContent(newUser.role\_id.ToString()), "role\_id");

HttpResponseMessage response = await client.PostAsync("http://motov-ae.tepk-it.ru/api/user/create", multiContent);

actual = (int)response.StatusCode;

}

Assert.AreEqual(expected, actual);

}

1. Изменение пользователей не авторизированным пользователем

Листинг 38

[TestMethod()]

public async Task FailTokenEditUserTest()

{

int actual = 0;

int expected = 401;

int userid = 0;

string Surname = "Швардыгула";

string Name = "Герман";

string Patronumic = "Георгиевич";

string PhoneNumber = "89097070101";

DateTime Birh = DateTime.Parse("1999-4-4");

string Login = "german90";

string Password = "ba658cd5f";

string Email = "german90@ya.ru";

int Role = 1;

using (HttpClient client = new HttpClient())

{

HttpResponseMessage response = await client.GetAsync("http://motov-ae.tepk-it.ru/api/users");

if (response.IsSuccessStatusCode)

{

string responseBody = await response.Content.ReadAsStringAsync();

List<User> users = JsonConvert.DeserializeObject<List<User>>(responseBody);

User nuser = users.FirstOrDefault(f => f.login == "german90");

userid = nuser.id;

}

}

User updatedUser = new User

{

surname = Surname,

name = Name,

patronymic = Patronumic,

phone\_number = PhoneNumber,

birth = Birh,

login = Login,

password = Password,

email = Email,

role\_id = Role

};

using (HttpClient client = new HttpClient())

{

MultipartFormDataContent multiContent = new MultipartFormDataContent();

multiContent.Add(new StringContent(updatedUser.name), "name");

multiContent.Add(new StringContent(updatedUser.surname), "surname");

multiContent.Add(new StringContent(updatedUser.patronymic), "patronymic");

multiContent.Add(new StringContent(updatedUser.phone\_number), "phone\_number");

multiContent.Add(new StringContent(updatedUser.birth.ToString("yyyy-M-d H:m:s")), "birth");

multiContent.Add(new StringContent(updatedUser.login), "login");

multiContent.Add(new StringContent(updatedUser.password), "password");

multiContent.Add(new StringContent(updatedUser.email), "email");

multiContent.Add(new StringContent(updatedUser.role\_id.ToString()), "role\_id");

HttpResponseMessage response = await client.PostAsync($"http://motov-ae.tepk-it.ru/api/user/update/{userid}", multiContent);

actual = (int)response.StatusCode;

}

Assert.AreEqual(expected, actual);

}

1. Удаление пользователей

Листинг 39

[TestMethod()]

public async Task ZDelUserTest()

{

string BaseUrl = "http://motov-ae.tepk-it.ru/api/login";

string login = "admin";

string password = "adminadmin";

int actual = 0;

int expected = 200;

int userid = 0;

using (HttpClient client = new HttpClient())

{

var parameters = new Dictionary<string, string>

{

{ "login", login },

{ "password", password }

};

string queryString = string.Join("&", parameters.Select(x => $"{x.Key}={x.Value}"));

HttpResponseMessage response = await client.PostAsync($"{BaseUrl}?{queryString}", null);

if (response.IsSuccessStatusCode)

{

string responseContent = await response.Content.ReadAsStringAsync();

JObject responseData = JObject.Parse(responseContent);

string token = (string)responseData["data"]["api\_token"];

Data.token = token;

}

}

using (HttpClient client = new HttpClient())

{

client.DefaultRequestHeaders.Authorization = new AuthenticationHeaderValue("Bearer", Data.token);

HttpResponseMessage response = await client.GetAsync("http://motov-ae.tepk-it.ru/api/users");

if (response.IsSuccessStatusCode)

{

string responseBody = await response.Content.ReadAsStringAsync();

List<User> users = JsonConvert.DeserializeObject<List<User>>(responseBody);

User nuser = users.FirstOrDefault(f => f.login == "german90");

userid = nuser.id;

}

}

using (HttpClient client = new HttpClient())

{

client.DefaultRequestHeaders.Authorization = new AuthenticationHeaderValue("Bearer", Data.token);

HttpResponseMessage response = await client.DeleteAsync($"http://motov-ae.tepk-it.ru/api/user/delete/{userid}");

actual = (int)response.StatusCode;

}

Assert.AreEqual(expected, actual);

}

## Таблица тестирования

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Номер теста** | **Назначение теста** | **Ожидаемый результат** | **Фактический результат** | **Статус теста** |
| TC  001 | Проверить, авторизацию под администратором | Код 200 | Код 200 | Пройден успешно (Passed) |
| TC  002 | Проверить, авторизацию под менеджером | Код 200 | Код 200 | Пройден успешно (Passed) |
| TC  003 | Проверить, авторизацию под другой ролью | Код 200 | Код 200 | Пройден успешно (Passed) |
| TC  004 | Проверить, авторизацию с неправильными данными | Код 401 | Код 401 | Пройден успешно (Passed) |
| TC  005 | Проверить, добавление фильма | Код 201 | Код 201 | Пройден успешно (Passed) |
| TC  006 | Проверить, изменение фильма | Код 200 | Код 200 | Пройден успешно (Passed) |
| TC  007 | Проверить, добавление фильма с неправильными данными | Код 422 | Код 422 | Пройден успешно (Passed) |
| TC  008 | Проверить, изменение фильма с неправильными данными | Код 422 | Код 422 | Пройден успешно (Passed) |
| TC  009 | Проверить, добавление фильма с неправильной ролью | Код 401 | Код 401 | Пройден успешно (Passed) |
| TC  010 | Проверить, изменение фильма с неправильной ролью | Код 401 | Код 401 | Пройден успешно (Passed) |
| TC  011 | Проверить, удаление фильма | Код 200 | Код 200 | Пройден успешно (Passed) |
| TC  012 | Проверить, добавление еды | Код 201 | Код 201 | Пройден успешно (Passed) |
| TC  013 | Проверить, изменение еды | Код 200 | Код 200 | Пройден успешно (Passed) |
| TC  014 | Проверить, добавление еды с неправильными данными | Код 422 | Код 422 | Пройден успешно (Passed) |
| TC  015 | Проверить, изменение еды с неправильными данными | Код 422 | Код 422 | Пройден успешно (Passed) |
| TC  016 | Проверить, добавление еды с неправильной ролью | Код 401 | Код 401 | Пройден успешно (Passed) |
| TC  017 | Проверить, изменение еды с неправильной ролью | Код 401 | Код 401 | Пройден успешно (Passed) |
| TC  018 | Проверить, удаление еды | Код 200 | Код 200 | Пройден успешно (Passed) |
| TC  019 | Проверить, добавление жанра | Код 201 | Код 201 | Пройден успешно (Passed) |
| TC  020 | Проверить, изменение жанра | Код 200 | Код 200 | Пройден успешно (Passed) |
| TC  021 | Проверить, добавление жанра с неправильными данными | Код 422 | Код 422 | Пройден успешно (Passed) |
| TC  022 | Проверить, изменение жанра с неправильными данными | Код 422 | Код 422 | Пройден успешно (Passed) |
| TC  023 | Проверить, добавление жанра с неправильной ролью | Код 401 | Код 401 | Пройден успешно (Passed) |
| TC  024 | Проверить, изменение жанра с неправильной ролью | Код 401 | Код 401 | Пройден успешно (Passed) |
| TC  025 | Проверить, удаление жанра | Код 200 | Код 200 | Пройден успешно (Passed) |
| TC  026 | Проверить, добавление сеанса | Код 201 | Код 201 | Пройден успешно (Passed) |
| TC  027 | Проверить, изменение сеанса | Код 200 | Код 200 | Пройден успешно (Passed) |
| TC  028 | Проверить, добавление сеанса с неправильными данными | Код 500 | Код 500 | Пройден успешно (Passed) |
| TC  029 | Проверить, изменение сеанса с неправильными данными | Код 500 | Код 500 | Пройден успешно (Passed) |
| TC  030 | Проверить, добавление сеанса с неправильной ролью | Код 401 | Код 401 | Пройден успешно (Passed) |
| TC  031 | Проверить, изменение сеанса с неправильной ролью | Код 401 | Код 401 | Пройден успешно (Passed) |
| TC  032 | Проверить, удаление сеанса | Код 200 | Код 200 | Пройден успешно (Passed) |
| TC  033 | Проверить, добавление фильма | Код 201 | Код 201 | Пройден успешно (Passed) |
| TC  034 | Проверить, изменение пользователя | Код 200 | Код 200 | Пройден успешно (Passed) |
| TC  035 | Проверить, добавление пользователя с неправильными данными | Код 422 | Код 422 | Пройден успешно (Passed) |
| TC  036 | Проверить, изменение пользователя с неправильными данными | Код 422 | Код 422 | Пройден успешно (Passed) |
| TC  037 | Проверить, добавление пользователя с неправильной ролью | Код 401 | Код 401 | Пройден успешно (Passed) |
| TC  038 | Проверить, изменение пользователя с неправильной ролью | Код 401 | Код 401 | Пройден успешно (Passed) |
| TC  039 | Проверить, удаление пользователя | Код 200 | Код 200 | Пройден успешно (Passed) |

## Вывод тестирования

По результатам тестирования можно сделать вывод, что приложение полностью соответствует требованиям, описанным в техническом задании. Отсутствие критических ошибок свидетельствует о готовности приложения к запуску и использованию конечными пользователями.