**ОБЛАСТНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ**

**ПРОФЕССИОНАЛЬНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ**

**«ТОМСКИЙ ЭКОНОМИКО-ПРОМЫШЛЕННЫЙ КОЛЛЕДЖ»**

**Проектирование и разработка интерфейсной части для информационной системы "Городской сад аттракционов" на платформе Windows Presentation Foundation**

***ОТЧЁТ ПО УЧЕБНОЙ ПРАКТИКЕ***

*СПЕЦИАЛЬНОСТЬ 09.02.07 ИНФОРМАЦИОННЫЕ СИСТЕМЫ И ПРОГРАММИРОВАНИЕ*

Выполнил:

Студент группы 1912с

\_\_\_\_\_\_ / Ойноткинов А.М.

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

Проверил:

Преподаватель

\_\_\_\_\_\_\_\_\_ /Летягина И.Е.

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

Оценка «\_\_\_\_\_\_\_\_\_\_\_\_\_\_»

Томск 2024

# ОГЛАВЛЕНИЕ

[ОГЛАВЛЕНИЕ 2](#_Toc166266387)

[ВВЕДЕНИЕ 4](#_Toc166266388)

[ОПИСАНИЕ ПРЕДМЕТНОЙ ОБЛАСТИ 4](#_Toc166266389)

[АКТУАЛЬНОСТЬ 4](#_Toc166266390)

[ЦЕЛЬ И ЗАДАЧИ 4](#_Toc166266391)

[Глоссарий 5](#_Toc166266392)

[ПРОЕКТИРОВАНИЕ И КОНЦЕПТ ГОРОДСКОЙ САД АТТРАКЦИОНОВ 6](#_Toc166266393)

[Структура приложения 6](#_Toc166266394)

[ТЕХНИЧЕСКОЕ ЗАДАНИЕ: ГОРОДСКОЙ САД АТТРАКЦИОНА 7](#_Toc166266395)

[1. Основные требования 7](#_Toc166266396)

[1.1. Авторизация 7](#_Toc166266397)

[1.2. Пользовательский интерфейс 7](#_Toc166266398)

[1.3. Функциональности для Менеджера 7](#_Toc166266399)

[1.4. Функциональности для Администратора 7](#_Toc166266400)

[2. Технические детали 7](#_Toc166266401)

[3.1. Платформа 7](#_Toc166266402)

[3.2. Безопасность 7](#_Toc166266403)

[3.3. Хранение данных 7](#_Toc166266404)

[3. Дополнительные требования 7](#_Toc166266405)

[3.1. Интерфейс 7](#_Toc166266406)

[3.2. Фотографии сувениров 8](#_Toc166266407)

[ТЕХНИЧЕСКАЯ РЕАЛИЗАЦИЯ 8](#_Toc166266408)

[Описание приложения 8](#_Toc166266409)

[Окно авторизации 8](#_Toc166266410)

[Главная страница 9](#_Toc166266411)

[Список заказов 11](#_Toc166266412)

[Просмотр дохода 12](#_Toc166266413)

[Просмотр популярного тарифа 14](#_Toc166266414)

[Просмотра списка сотрудников 14](#_Toc166266415)

[Добавление сотрудника 15](#_Toc166266416)

[Просмотр списка сувениров 16](#_Toc166266417)

[Изменение сувенира 17](#_Toc166266418)

[Добавления сувенира 18](#_Toc166266419)

[ТЕСТИРОВАНИЕ 19](#_Toc166266420)

[ЗАКЛЮЧЕНИЕ 48](#_Toc166266421)

# ВВЕДЕНИЕ

## ОПИСАНИЕ ПРЕДМЕТНОЙ ОБЛАСТИ

Предметной областью является предоставляет всесторонний инструментарий для управления и анализа основных аспектов работы аттракционного парка, обеспечивая комфортное и эффективное взаимодействие персонала с информацией.Начало формы

## АКТУАЛЬНОСТЬ

Важно отметить, что в современном мире автоматизация и цифровизация играют ключевую роль в управлении бизнесом. Приложение предоставляет сотрудникам инструменты для эффективного управления заказами сувениров, доходом и персоналом, что повышает производительность и оптимизирует рабочие процессы.

## ЦЕЛЬ И ЗАДАЧИ

Цель:

Разработать приложение "Городской сад аттракционов" в рамках WPF, для обеспечения повышения эффективности управления и максимизация прибыли аттракционного парка.

Задачи:

* + **Спроектировать и разработать интерфейс приложения.**
  + **Спроектировать и разработать функционал приложения.**
  + **Провести тесты разработанного приложения.**

## Глоссарий

|  |  |
| --- | --- |
| **Понятие** | **Определение понятия** |
| 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. |
| Postman | Это популярный инструмент для разработки, тестирования и документирования API. Он предоставляет разработчикам удобный способ отправлять HTTP-запросы к веб-сервисам и анализировать ответы. |
| 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. |
| WPF | WPF (Windows Presentation Foundation) — это фреймворк, разработанный Microsoft для создания графических пользовательских интерфейсов в приложениях под операционные системы Windows. Он предоставляет разработчикам широкие возможности для создания интерактивных, стильных и функциональных пользовательских интерфейсов, используя различные элементы управления, анимации, шаблоны и стили. WPF основан на векторной графике, что обеспечивает масштабируемость интерфейсов и высокое качество отображения на экранах с различным разрешением. Он также интегрируется с языком разметки XAML (eXtensible Application Markup Language), что позволяет разделять дизайн интерфейса и логику приложения, упрощая разработку и поддержку кода. WPF часто используется для создания различных приложений, включая десктопные приложения, игры, инструменты аналитики данных и другие. |

# ПРОЕКТИРОВАНИЕ И КОНЦЕПТ ГОРОДСКОЙ САД АТТРАКЦИОНОВ

## Структура приложения

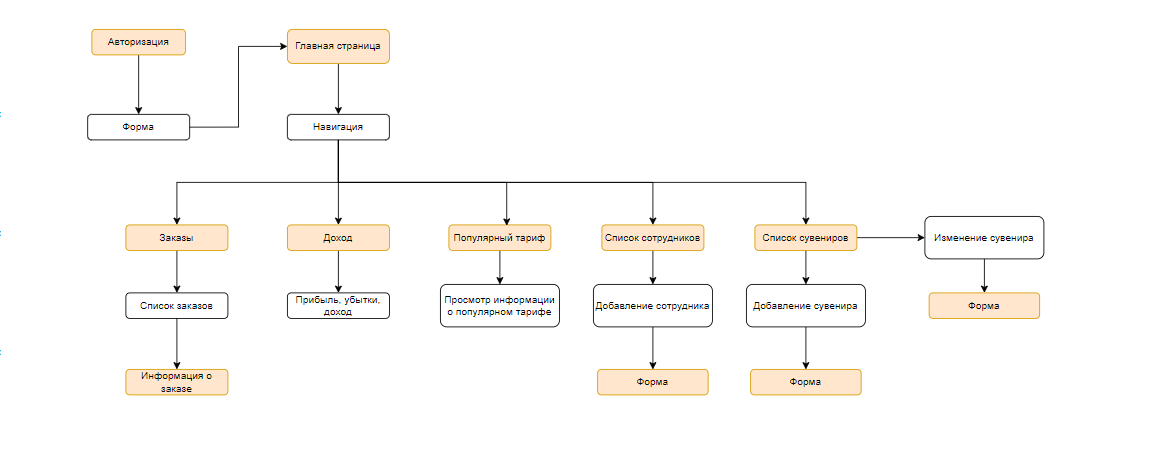


Рисунок 1 Структура приложения

# ТЕХНИЧЕСКОЕ ЗАДАНИЕ: ГОРОДСКОЙ САД АТТРАКЦИОНА

Целью данного проекта является разработка приложения для взаимодействия с работниками Городского сада Аттракциона. Приложение будет предоставлять удобный интерфейс для авторизации пользователей, управления заказами, сотрудниками и сувенирами.

## Основные требования

### Авторизация

* При открытии приложения отображается страница авторизации.
* Поля на странице авторизации: логин, пароль.
* Кнопка "Войти" для подтверждения данных.

### Пользовательский интерфейс

* После успешной авторизации отображается информация о сотруднике: Фамилия, Имя, Роль.
* Предоставленные возможности зависят от роли:
* Менеджер: просмотр заказов, список сувениров.
* Администратор: просмотр заказов, просмотр дохода, просмотр самого популярного тарифа, список сотрудников, список сувениров.

### Функциональности для Менеджера

* Просмотр заказов.
* Просмотр списка сувениров.

### Функциональности для Администратора

* Просмотр заказов с детализацией: номер заказа, название товара, имя заказчика, цена, количество, номер телефона заказчика, итоговая цена заказа.
* Просмотр дохода за выбранный период.
* Просмотр самого популярного тарифа.
* Управление списком сотрудников: просмотр, добавление нового сотрудника.
* Управление списком сувениров: просмотр, добавление, изменение, удаление.

## Технические детали

### 3.1. Платформа

Приложение разрабатывается на WPF (Windows Presentation Foundation).

### 3.2. Безопасность

Пароли пользователей хранятся в зашифрованном виде.

### 3.3. Хранение данных

Использование локальной базы данных для хранения информации о сотрудниках, заказах и сувенирах.

## Дополнительные требования

### 3.1. Интерфейс

* Интуитивно понятный и привлекательный пользовательский интерфейс.
* Поддержка русского языка.

### 3.2. Фотографии сувениров

Возможность добавления и просмотра фотографий сувениров.

# ТЕХНИЧЕСКАЯ РЕАЛИЗАЦИЯ

## Описание приложения

## Окно авторизации

Разработка приложения началась с создания структуры приложения (Рис.1), после его создания было решено перейти к создания концепта дизайна в draw.io. После перехода было решено начать с создания концепта окна авторизации. (Рис.2). А после создания его в приложении. (Рис.3)

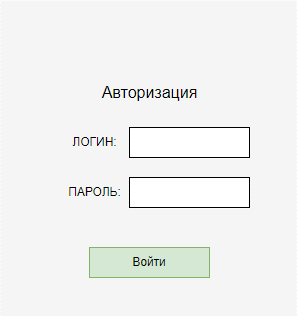


Рисунок 2 Концепт окна авторизация

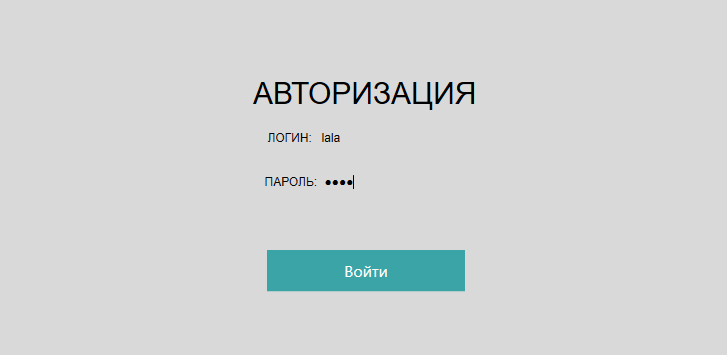


Рисунок 3 Страница авторизации в приложении

## Главная страница

После создания страницы авторизации было решено перейти к созданию концепта Главной страницы (Рис.4). В ней было решено создать приветствие пользователя, а также вывода его роли, в зависимости от роли пользователя ему будут доступны определенные возможности, для администратора это просмотр всех заказов, просмотр дохода, просмотр самого популярного тарифа, и просмотра списка со всеми сотрудниками. (Рис.4). В то время если авторизовался менеджер, ему будет доступно только просмотр заказов и список сувениров. (Рис.6). После создания концепта было решено перейти к созданию его в приложении. (Рис.5). После создания страницы в приложении было решено ее реализовать как она будет выгладить для менеджера (Рис.7).

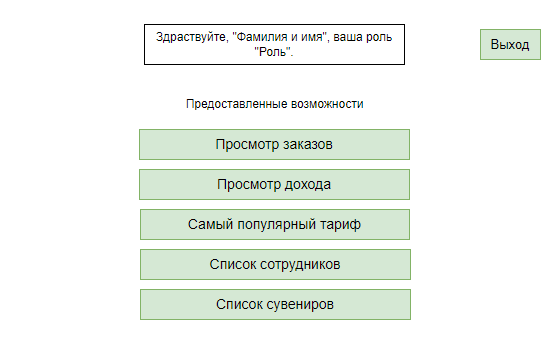


Рисунок 4 Концепт главной страницы (Для администратора)

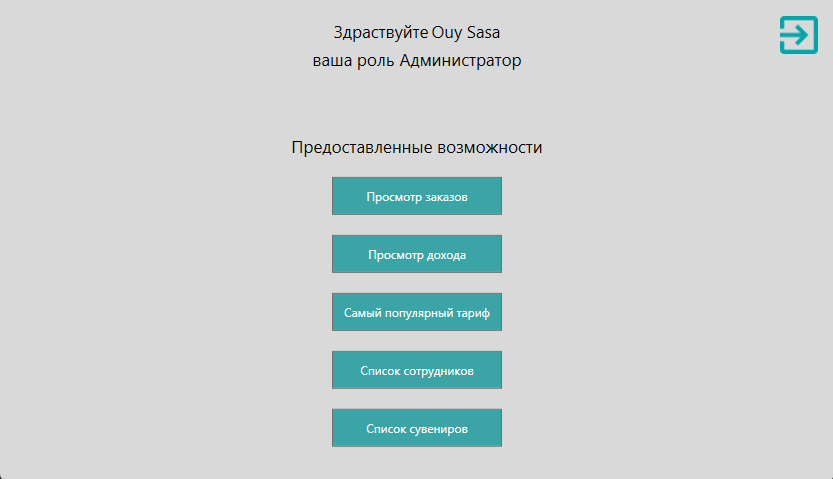


Рисунок 5 Главная страница в приложении (Для администратора)

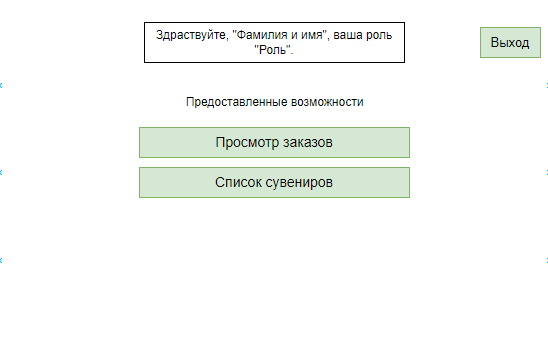


Рисунок 6 Концепт главной страницы (Для менеджера)

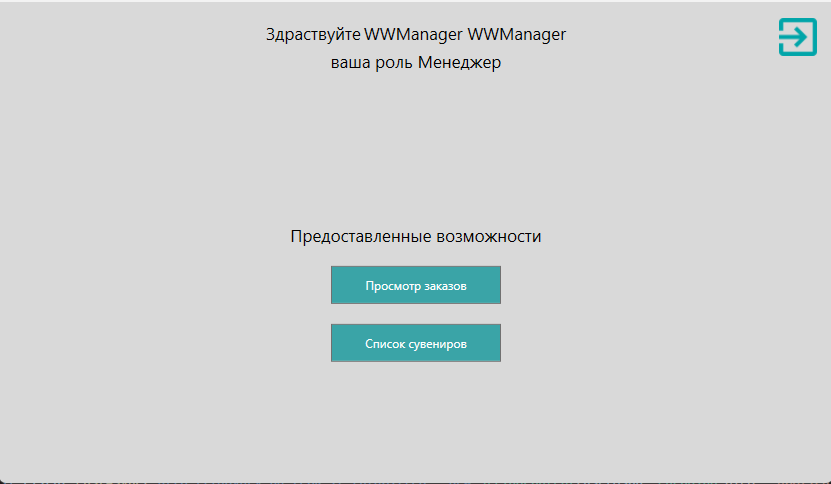


Рисунок 7 Главная страница в приложении (Для менеджера)

## Список заказов

После создания главной было решено перейти к созданию концепта списка заказов в нем было решено выводить номер заказа, название товара, имя заказчика, цена товара, количество заказанного товара, и итоговая цена (Рис.8). После создания концепта было решено реализовать его в приложении. (Рис.9).

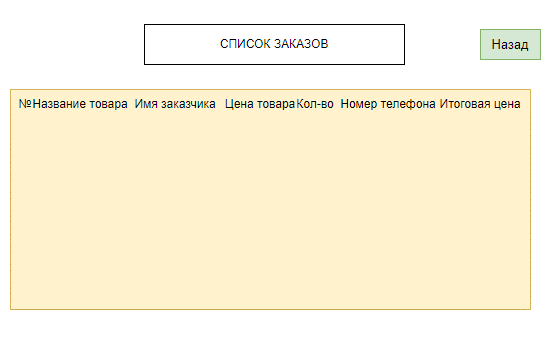


Рисунок 8 Концепт списка заказов

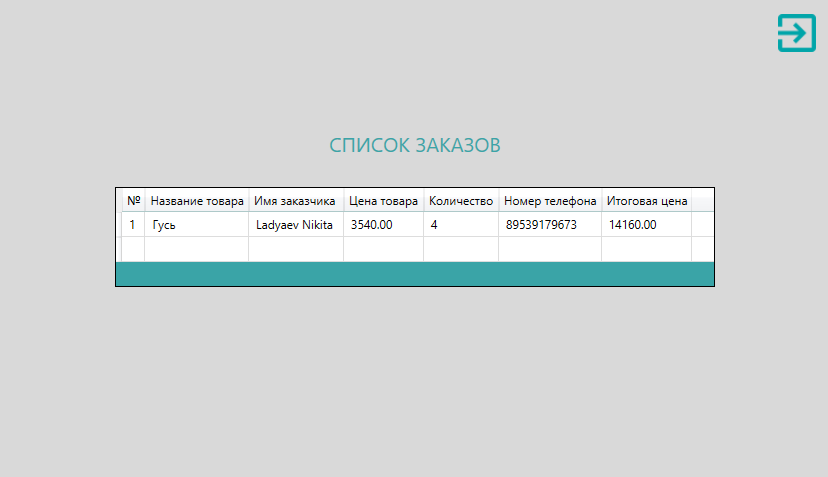


Рисунок 9 Главная страница в приложении (Для администратора)

## Просмотр дохода

После создания списка заказов было решено перейти к созданию, концепт просмотра дохода в нем было решено выводить поля, состоящее из 2 дат также кнопку, при нажатии на которую будет выводиться доход за эти 2 даты (Рис.10). После создания концепта было решено реализовать его в приложении. (Рис.11).

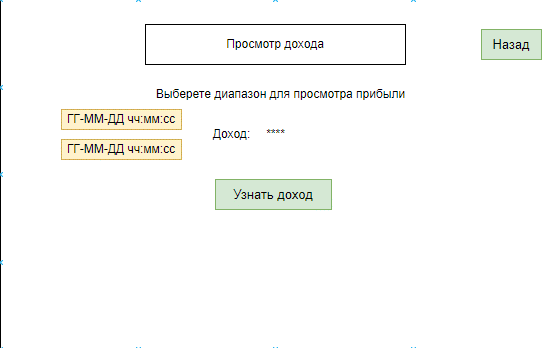


Рисунок 10 Концепт просмотра дохода

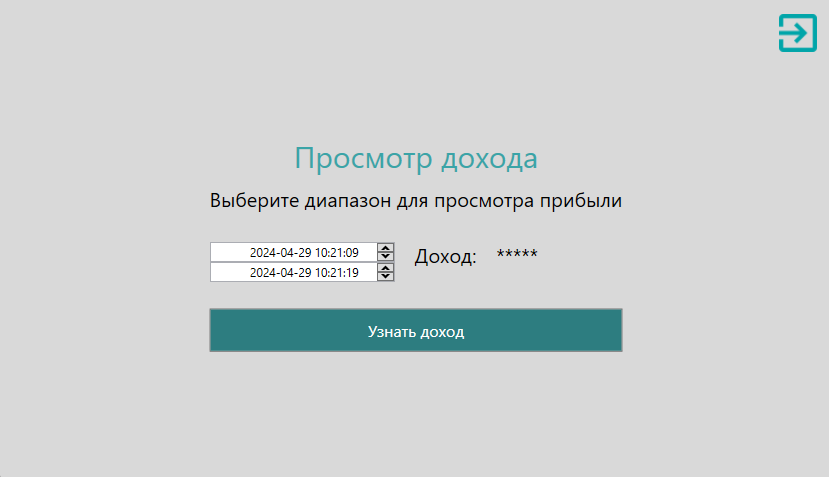


Рисунок 11 Главная страница в приложении (Для администратора)

## Просмотр популярного тарифа

После просмотра дохода было решено перейти к созданию концепта просмотра популярного тарифа в нем было решено выводить поля, с названием популярного тарифа, описанием тарифа, и количеством покупок за все время этого тарифа (Рис.12). После создания концепта было решено реализовать его в приложении. (Рис.13).

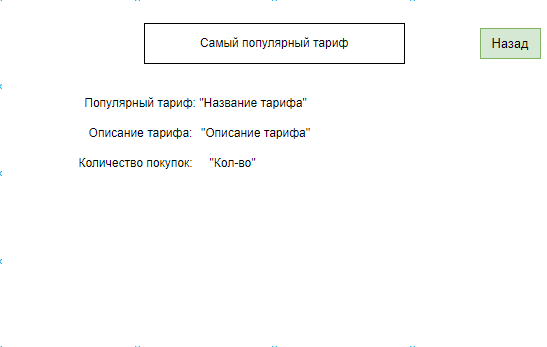


Рисунок 12 Концепт просмотра популярного тарифа

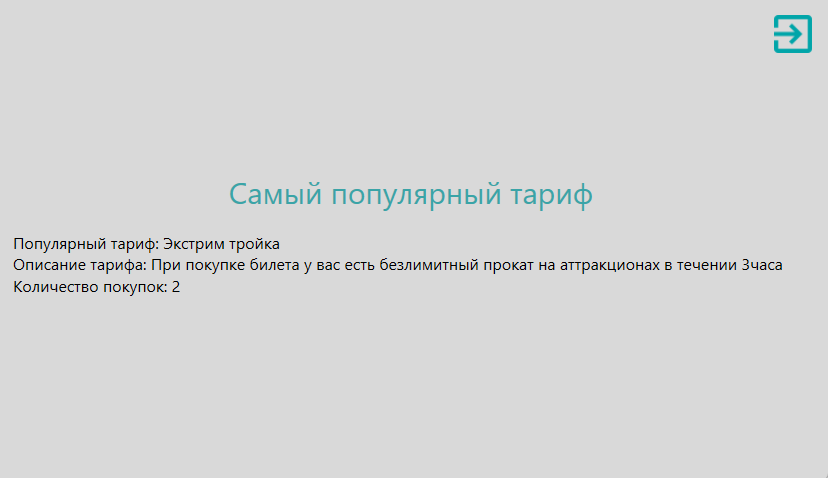


Рисунок 13 Главная страница в приложении (Для администратора)

## Просмотра списка сотрудников

После просмотра популярного тарифа было решено перейти к созданию концепта просмотра списка сотрудников в нем было решено выводить поля, с ФИО сотрудника и его ролью (Рис.14). После создания концепта было решено реализовать его в приложении. (Рис.15).

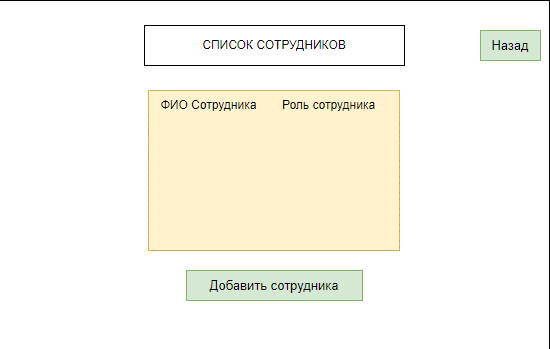


Рисунок 14 Концепт просмотра списка сотрудников



Рисунок 15 Страница списка сотрудников (Для администратора)

## Добавление сотрудника

После создания страницы просмотра списка сотрудников было решено перейти к реализации кнопки для добавления сотрудника (Рис.16). После создания концепта было решено реализовать его в приложении. (Рис.17).

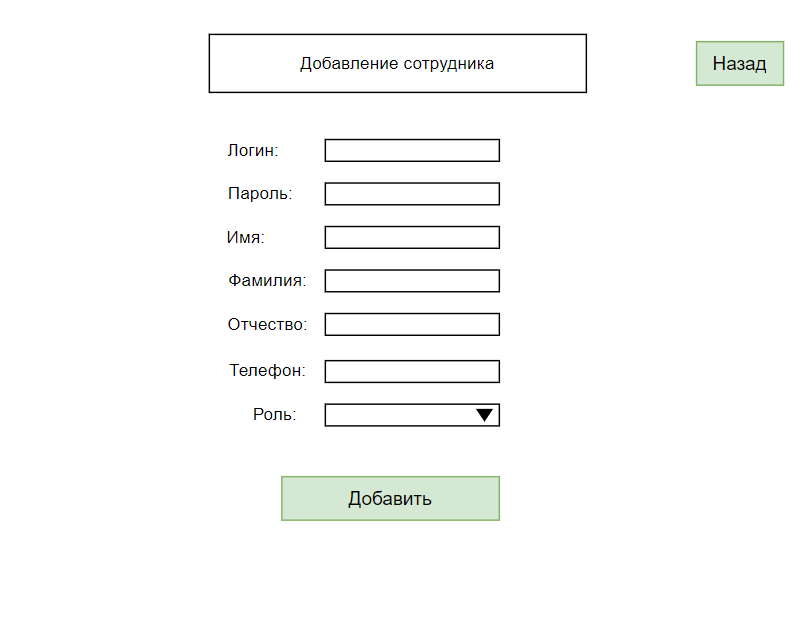


Рисунок 16 Концепт добавления сотрудника

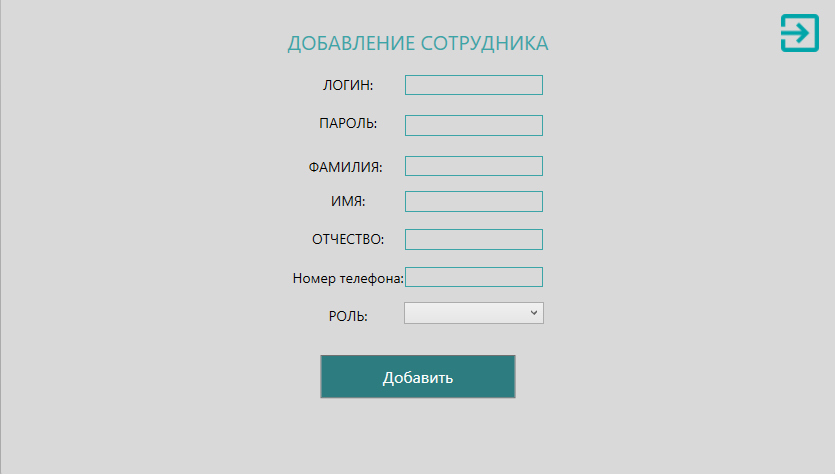


Рисунок 17 Главная страница в приложении (Для администратора)

## Просмотр списка сувениров

После создания страницы для добавления сотрудника было решено перейти созданию концепта просмотра списка сувениров (Рис.18). После создания концепта было решено реализовать его в приложении. (Рис.19).

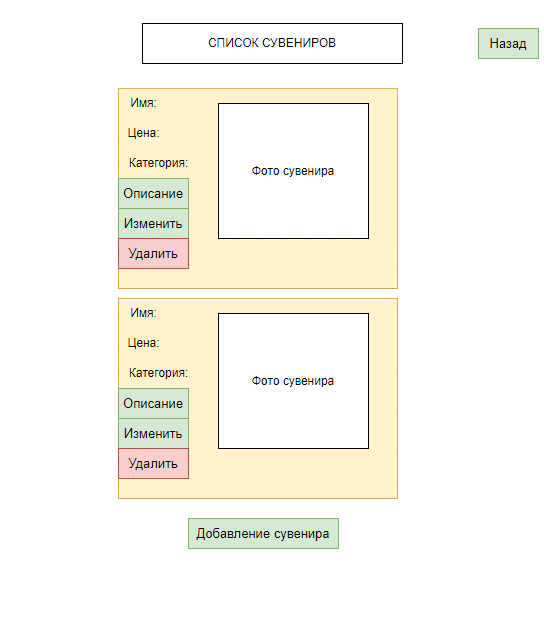


Рисунок 18 Концепт просмотра списка сувениров



Рисунок 19 Страница просмотра списка сувениров (Для администратора)

## Изменение сувенира

После создания страницы просмотра списка сувениров было решено перейти созданию концепта изменения сувенира (Рис.20). После создания концепта было решено реализовать его в приложении. (Рис.21).

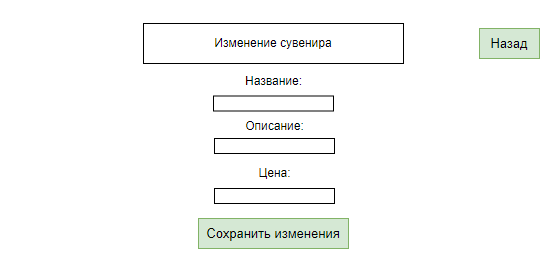


Рисунок 20 Концепт изменения сувенира

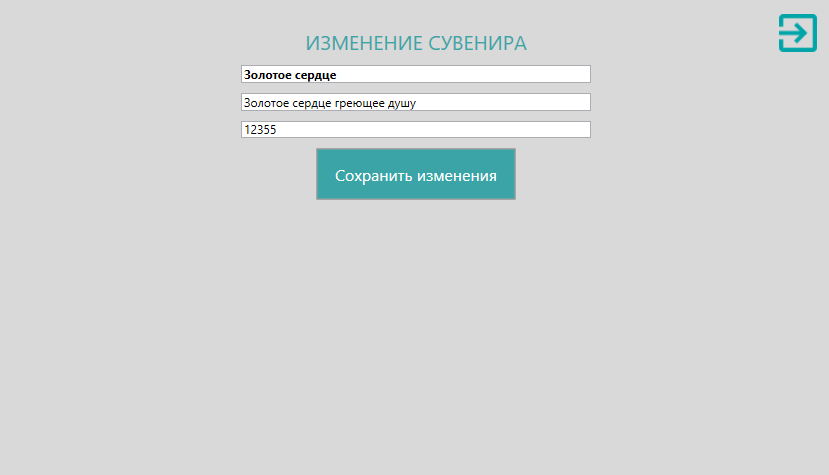


Рисунок 21 Страница изменения сувенира (Для администратора)

## Добавления сувенира

После создания страницы изменения сувенира было решено перейти созданию концепт добавления сувенира (Рис.20). После создания концепта было решено реализовать его в приложении. (Рис.21).



Рисунок 22 Концепт добавления сувенира

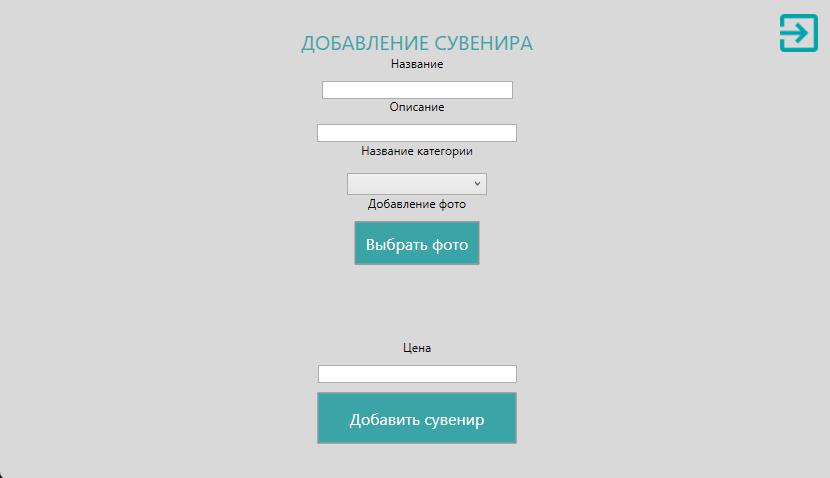


Рисунок 23 Главная страница в приложении (Для администратора)

## ТЕСТИРОВАНИЕ

Листинг 1

[TestMethod]

public async Task AddEmployee\_Successful\_Returns422()

{

string password = "Ss1@";

string login = "sasa-ouy";

int actual = 0;

int expected = 422;

User user = null;

// Авторизация и получение токена

var credentials = new { login = login, password = password };

string token = null;

using (HttpClient client = new HttpClient())

{

try

{

HttpResponseMessage response = await client.PostAsync("http://ladyaev-na.tepk-it.ru/api/login",

new StringContent(JsonConvert.SerializeObject(credentials), Encoding.UTF8, "application/json"));

response.EnsureSuccessStatusCode(); // Гарантирует, что ответ успешный

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

// Парсим ответ в объект

var responseObject = JsonConvert.DeserializeObject<dynamic>(responseBody);

token = JsonConvert.DeserializeObject<string>(responseBody);

// Сохраняем токен в настройках приложения

Token.token = token;

}

catch (Exception)

{

// Ошибка при неправильно введенных данных

MessageBox.Show("Ошибка аутентификации");

}

}

// Вызов метода RegisterUser, передавая токен

user = new User

{

login = "",

password = "validPasswor",

name = "000",

surname = "000",

patronymic = "0000",

phone = "123456789",

role\_id = 2

};

string userJson = JsonConvert.SerializeObject(user);

using (HttpClient client = new HttpClient())

{

try

{

client.DefaultRequestHeaders.Authorization = new AuthenticationHeaderValue("Bearer", token); // Установка заголовка авторизации

HttpResponseMessage response = await client.PostAsync("http://ladyaev-na.tepk-it.ru/api/reg", new StringContent(userJson, Encoding.UTF8, "application/json"));

if (response.IsSuccessStatusCode)

{

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

var createdUser = JsonConvert.DeserializeObject<User>(responseContent);

string createdUserId = createdUser.id.ToString(); // Преобразование ID в строку

try

{

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

HttpResponseMessage deleteResponse = await client.DeleteAsync($"http://ladyaev-na.tepk-it.ru/api/user/delete/{createdUserId}");

if (deleteResponse.IsSuccessStatusCode)

{

MessageBox.Show("Пользователь успешно удален!");

}

else

{

MessageBox.Show($"Ошибка при удалении пользователя: {deleteResponse.StatusCode}");

}

}

catch (Exception ex)

{

MessageBox.Show($"Ошибка при удалении пользователя: {ex.Message}");

}

}

else if (response.StatusCode == (HttpStatusCode)422)

{

actual = (int)response.StatusCode;

}

}

catch (Exception ex)

{

MessageBox.Show($"Ошибка: {ex.Message}");

}

}

Assert.AreEqual(expected, actual);

Листинг 2

[TestMethod]

public async Task AddEmployee\_Successful\_ReturnsTrue()

{

string password = "Ss1@";

string login = "sasa-ouy";

bool actual = false;

bool expected = true;

User user = null;

// Авторизация и получение токена

var credentials = new { login = login, password = password };

string token = null;

using (HttpClient client = new HttpClient())

{

try

{

HttpResponseMessage response = await client.PostAsync("http://ladyaev-na.tepk-it.ru/api/login",

new StringContent(JsonConvert.SerializeObject(credentials), Encoding.UTF8, "application/json"));

response.EnsureSuccessStatusCode(); // Гарантирует, что ответ успешный

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

// Парсим ответ в объект

var responseObject = JsonConvert.DeserializeObject<dynamic>(responseBody);

token = JsonConvert.DeserializeObject<string>(responseBody);

// Сохраняем токен в настройках приложения

Token.token = token;

}

catch (Exception)

{

// Ошибка при неправильно введенных данных

MessageBox.Show("Ошибка аутентификации");

}

}

// Вызов метода RegisterUser, передавая токен

user = new User

{

login = "validLogin1",

password = "validPassword1+",

name = "John",

surname = "Doe",

patronymic = "Smith",

phone = "123456789",

role\_id = 2

};

string userJson = JsonConvert.SerializeObject(user);

using (HttpClient client = new HttpClient())

{

try

{

client.DefaultRequestHeaders.Authorization = new AuthenticationHeaderValue("Bearer", token); // Установка заголовка авторизации

HttpResponseMessage response = await client.PostAsync("http://ladyaev-na.tepk-it.ru/api/reg", new StringContent(userJson, Encoding.UTF8, "application/json"));

if (response.IsSuccessStatusCode)

{

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

var createdUser = JsonConvert.DeserializeObject<User>(responseContent);

string createdUserId = createdUser.id.ToString(); // Преобразование ID в строку

actual = true;

try

{

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

HttpResponseMessage deleteResponse = await client.DeleteAsync($"http://ladyaev-na.tepk-it.ru/api/user/delete/{createdUserId}");

if (deleteResponse.StatusCode == HttpStatusCode.Gone)

{

}

else

{

MessageBox.Show($"Ошибка при удалении пользователя: {deleteResponse.StatusCode}");

}

}

catch (WebException ex)

{

if (ex.Response is HttpWebResponse webResponse && webResponse.StatusCode == HttpStatusCode.Gone)

{

MessageBox.Show("Ошибка 410: Ресурс удален. Пользователь не найден.");

}

else

{

MessageBox.Show($"Ошибка при удалении пользователя: {ex.Message}");

}

}

}

else

{

MessageBox.Show($"{response}");

actual = false;

}

}

catch (Exception ex)

{

MessageBox.Show($"Ошибка: {ex.Message}");

actual = false;

}

}

Assert.AreEqual(expected, actual);

}

Листинг 3

public async Task AddSouvenir\_Successful\_ReturnsTrue()

{

// Arrange

string password = "Ss1@";

string login = "sasa-ouy";

bool actual = false;

bool expected = true;

// Авторизация и получение токена

var credentials = new { login = login, password = password };

string token = null;

using (HttpClient client = new HttpClient())

{

try

{

HttpResponseMessage response = await client.PostAsync("http://ladyaev-na.tepk-it.ru/api/login",

new StringContent(JsonConvert.SerializeObject(credentials), Encoding.UTF8, "application/json"));

response.EnsureSuccessStatusCode(); // Гарантирует, что ответ успешный

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

// Парсим ответ в объект

var responseObject = JsonConvert.DeserializeObject<dynamic>(responseBody);

token = JsonConvert.DeserializeObject<string>(responseBody);// Предполагается, что токен доступен здесь

// Сохраняем токен в настройках приложения

Token.token = token;

}

catch (Exception)

{

// Ошибка при неправильно введенных данных

}

}

try

{

// Создание объекта данных для сувенира

var souvenirData = new

{

name = "Sample Souvenir",

price = 14.00,

description = "Sample Description",

category\_souvenir\_id = 1

};

// Сериализация объекта данных в формат JSON

string jsonData = JsonConvert.SerializeObject(souvenirData);

// Создание HttpClient для отправки запроса

using (HttpClient client = new HttpClient())

{

// Добавление токена в заголовок запроса

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

// Добавление типа содержимого в заголовок

client.DefaultRequestHeaders.Accept.Add(new MediaTypeWithQualityHeaderValue("application/json"));

// Отправка запроса на сервер

HttpResponseMessage response = await client.PostAsync("http://ladyaev-na.tepk-it.ru/api/addSouvenir",

new StringContent(jsonData, Encoding.UTF8, "application/json"));

// Проверка успешности запроса

if (response.IsSuccessStatusCode)

{

actual = true;

}

else

{

actual = false;

}

}

}

catch (Exception ex)

{

Console.WriteLine($"Ошибка: {ex.Message}");

}

Assert.AreEqual(expected, actual);

}

Листинг 4

[TestMethod]

public async Task AddSouvenir\_WithoutToken()

{

// Arrange

int actual = 0;

int expected = 401;

try

{

// Создание объекта данных для сувенира

var souvenirData = new

{

name = "ADS",

price = 13.00,

description = "asda",

category\_souvenir\_id = 0

};

// Сериализация объекта данных в формат JSON

string jsonData = JsonConvert.SerializeObject(souvenirData);

// Создание HttpClient для отправки запроса

using (HttpClient client = new HttpClient())

{

// Добавление токена в заголовок запроса

// Добавление типа содержимого в заголовок

client.DefaultRequestHeaders.Accept.Add(new MediaTypeWithQualityHeaderValue("application/json"));

// Отправка запроса на сервер

HttpResponseMessage response = await client.PostAsync($"http://ladyaev-na.tepk-it.ru/api/addSouvenir",

new StringContent(jsonData, Encoding.UTF8, "application/json"));

// Проверка успешности запроса

actual = (int)response.StatusCode;

Assert.AreEqual(expected, actual);

}

}

catch (Exception ex)

{

Console.WriteLine($"Ошибка: {ex.Message}");

}

}

}

Листинг 5

[TestMethod]

public async Task AddSouvenir\_Successful\_Returns422()

{

// Arrange

string password = "Ss1@";

string login = "sasa-ouy";

bool actual = true;

bool expected = false;

// Авторизация и получение токена

var credentials = new { login = login, password = password };

string token = null;

using (HttpClient client = new HttpClient())

{

try

{

HttpResponseMessage response = await client.PostAsync("http://ladyaev-na.tepk-it.ru/api/login",

new StringContent(JsonConvert.SerializeObject(credentials), Encoding.UTF8, "application/json"));

response.EnsureSuccessStatusCode(); // Гарантирует, что ответ успешный

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

// Парсим ответ в объект

var responseObject = JsonConvert.DeserializeObject<dynamic>(responseBody);

token = JsonConvert.DeserializeObject<string>(responseBody);// Предполагается, что токен доступен здесь

// Сохраняем токен в настройках приложения

Token.token = token;

}

catch (Exception)

{

// Ошибка при неправильно введенных данных

MessageBox.Show("Ошибка аутентификации");

}

}

try

{

// Создание объекта данных для сувенира

var souvenirData = new

{

name = "",

price = 0.00,

description = "",

category\_souvenir\_id = 0

};

// Сериализация объекта данных в формат JSON

string jsonData = JsonConvert.SerializeObject(souvenirData);

// Создание HttpClient для отправки запроса

using (HttpClient client = new HttpClient())

{

// Добавление токена в заголовок запроса

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

// Добавление типа содержимого в заголовок

client.DefaultRequestHeaders.Accept.Add(new MediaTypeWithQualityHeaderValue("application/json"));

// Отправка запроса на сервер

HttpResponseMessage response = await client.PostAsync("http://ladyaev-na.tepk-it.ru/api/addSouvenir",

new StringContent(jsonData, Encoding.UTF8, "application/json"));

// Проверка успешности запроса

if (response.IsSuccessStatusCode)

{

MessageBox.Show("Сувенир успешно добавлен!");

actual = true;

}

else

{

actual = false;

}

}

}

catch (Exception ex)

{

Console.WriteLine($"Ошибка: {ex.Message}");

}

Assert.AreEqual(expected, actual);

}

Листинг 6

[TestMethod]

public async Task AuthenticateAsync\_EmptyCredentials\_ReturnsFalse()

{

// Arrange

var authorizationPage = new AutorizationPage(null);

// Act

bool isAuthenticated = await authorizationPage.AuthenticateAsync("", "");

// Assert

Assert.IsFalse(isAuthenticated);

}

Листинг 7

[TestMethod]

public async Task AuthenticateAsync\_EmptyLogin\_ReturnsFalse()

{

// Arrange

var authorizationPage = new AutorizationPage(null);

// Act

bool isAuthenticated = await authorizationPage.AuthenticateAsync("", "asdasdad");

// Assert

Assert.IsFalse(isAuthenticated);

}

Листинг 8

[TestMethod]

public async Task AuthenticateAsync\_EmptyLogin\_ReturnsFalse()

{

// Arrange

var authorizationPage = new AutorizationPage(null);

// Act

bool isAuthenticated = await authorizationPage.AuthenticateAsync("", "asdasdad");

// Assert

Assert.IsFalse(isAuthenticated);

}

Листинг 9

[TestMethod]

public async Task AuthenticateAsync\_InvalidCredentials\_ReturnsFalse()

{

// Arrange

var authorizationPage = new AutorizationPage(null);

// Act

bool isAuthenticated = await authorizationPage.AuthenticateAsync("invalidLogin", "invalidPassword");

// Assert

Assert.IsFalse(isAuthenticated);

}

Листинг 10

[TestMethod]

public async Task AuthenticateAsync\_ValidCredentials\_ReturnsTrue()

{

// Arrange

var authorizationPage = new AutorizationPage(null);

// Act

bool isAuthenticated = await authorizationPage.AuthenticateAsync("sasa-ouy", "Ss1@");

Assert.IsTrue(isAuthenticated);

}

Листинг 11

[TestMethod]

public async Task EditSouvenir\_Successful\_ReturnsTrue()

{

// Arrange

string password = "Ss1@";

string login = "sasa-ouy";

int souvenirId = 13;

bool actual = false;

bool expected = true;

// Авторизация и получение токена

var credentials = new { login = login, password = password };

string token = null;

using (HttpClient client = new HttpClient())

{

try

{

HttpResponseMessage response = await client.PostAsync("http://ladyaev-na.tepk-it.ru/api/login",

new StringContent(JsonConvert.SerializeObject(credentials), Encoding.UTF8, "application/json"));

response.EnsureSuccessStatusCode(); // Гарантирует, что ответ успешный

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

// Парсим ответ в объект

var responseObject = JsonConvert.DeserializeObject<dynamic>(responseBody);

token = JsonConvert.DeserializeObject<string>(responseBody);// Предполагается, что токен доступен здесь

// Сохраняем токен в настройках приложения

Token.token = token;

}

catch (Exception)

{

// Ошибка при неправильно введенных данных

MessageBox.Show("Ошибка аутентификации");

}

}

try

{

// Создание объекта данных для сувенира

var souvenirData = new

{

name = "Sample Souvenir13",

price = 13.00,

description = "Sample Description13",

};

// Сериализация объекта данных в формат JSON

string jsonData = JsonConvert.SerializeObject(souvenirData);

// Создание HttpClient для отправки запроса

using (HttpClient client = new HttpClient())

{

// Добавление токена в заголовок запроса

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

// Добавление типа содержимого в заголовок

client.DefaultRequestHeaders.Accept.Add(new MediaTypeWithQualityHeaderValue("application/json"));

// Отправка запроса на сервер

HttpResponseMessage response = await client.PostAsync($"http://ladyaev-na.tepk-it.ru/api/updateSouvenir/{souvenirId}",

new StringContent(jsonData, Encoding.UTF8, "application/json"));

// Проверка успешности запроса

if (response.IsSuccessStatusCode)

{

actual = true;

}

else

{

actual = false;

}

}

}

catch (Exception ex)

{

Console.WriteLine($"Ошибка: {ex.Message}");

}

Assert.AreEqual(expected, actual);

}

Листинг 12

[TestMethod]

public async Task EditSouvenir\_Successful\_Returns422()

{

// Arrange

string password = "Ss1@";

string login = "sasa-ouy";

int souvenirId = 13;

int actual = 0;

int expected = 422;

// Авторизация и получение токена

var credentials = new { login = login, password = password };

string token = null;

using (HttpClient client = new HttpClient())

{

try

{

HttpResponseMessage response = await client.PostAsync("http://ladyaev-na.tepk-it.ru/api/login",

new StringContent(JsonConvert.SerializeObject(credentials), Encoding.UTF8, "application/json"));

response.EnsureSuccessStatusCode(); // Гарантирует, что ответ успешный

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

// Парсим ответ в объект

var responseObject = JsonConvert.DeserializeObject<dynamic>(responseBody);

token = JsonConvert.DeserializeObject<string>(responseBody);// Предполагается, что токен доступен здесь

// Сохраняем токен в настройках приложения

Token.token = token;

}

catch (Exception)

{

// Ошибка при неправильно введенных данных

MessageBox.Show("Ошибка аутентификации");

}

}

try

{

// Создание объекта данных для сувенира

var souvenirData = new

{

name = "",

price = 13.00,

description = "",

};

// Сериализация объекта данных в формат JSON

string jsonData = JsonConvert.SerializeObject(souvenirData);

// Создание HttpClient для отправки запроса

using (HttpClient client = new HttpClient())

{

// Добавление токена в заголовок запроса

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

// Добавление типа содержимого в заголовок

client.DefaultRequestHeaders.Accept.Add(new MediaTypeWithQualityHeaderValue("application/json"));

// Отправка запроса на сервер

HttpResponseMessage response = await client.PostAsync($"http://ladyaev-na.tepk-it.ru/api/updateSouvenir/{souvenirId}",

new StringContent(jsonData, Encoding.UTF8, "application/json"));

// Проверка успешности запроса

actual = (int)response.StatusCode;

}

}

catch (Exception ex)

{

Console.WriteLine($"Ошибка: {ex.Message}");

}

Assert.AreEqual(expected, actual);

}

Листинг 13

[TestMethod]

public async Task EditSouvenir\_WithoutToken()

{

// Arrange

int souvenirId = 13;

int actual = 0;

int expected = 401;

try

{

// Создание объекта данных для сувенира

var souvenirData = new

{

name = "ADS",

price = 13.00,

description = "asda",

};

// Сериализация объекта данных в формат JSON

string jsonData = JsonConvert.SerializeObject(souvenirData);

// Создание HttpClient для отправки запроса

using (HttpClient client = new HttpClient())

{

// Добавление токена в заголовок запроса

// Добавление типа содержимого в заголовок

client.DefaultRequestHeaders.Accept.Add(new MediaTypeWithQualityHeaderValue("application/json"));

// Отправка запроса на сервер

HttpResponseMessage response = await client.PostAsync($"http://ladyaev-na.tepk-it.ru/api/updateSouvenir/{souvenirId}",

new StringContent(jsonData, Encoding.UTF8, "application/json"));

// Проверка успешности запроса

actual = (int)response.StatusCode;

Assert.AreEqual(expected, actual);

}

}

catch (Exception ex)

{

Console.WriteLine($"Ошибка: {ex.Message}");

}

}

}

Листинг 14

[TestMethod]

public async Task Employeelist\_Successful\_ReturnsTrue()

{

// Arrange

string password = "Ss1@";

string login = "sasa-ouy";

bool actual = false;

bool expected = true;

// Авторизация и получение токена

var credentials = new { login = login, password = password };

string token = null;

using (HttpClient client = new HttpClient())

{

try

{

HttpResponseMessage response = await client.PostAsync("http://ladyaev-na.tepk-it.ru/api/login",

new StringContent(JsonConvert.SerializeObject(credentials), Encoding.UTF8, "application/json"));

response.EnsureSuccessStatusCode(); // Гарантирует, что ответ успешный

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

// Парсим ответ в объект

var responseObject = JsonConvert.DeserializeObject<dynamic>(responseBody);

token = JsonConvert.DeserializeObject<string>(responseBody);// Предполагается, что токен доступен здесь

// Сохраняем токен в настройках приложения

Token.token = token;

}

catch (Exception)

{

// Ошибка при неправильно введенных данных

MessageBox.Show("Ошибка аутентификации");

}

}

try

{

using (HttpClient client = new HttpClient())

{

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

try

{

// Запрос на получение списка пользователей

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

response.EnsureSuccessStatusCode();

if (response.IsSuccessStatusCode)

{

actual = true;

}

else

{

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

actual = false;

}

}

catch (HttpRequestException ex)

{

actual = false;

}

}

}

catch (Exception ex)

{

MessageBox.Show($"Непредвиденная ошибка: {ex.Message}");

}

Assert.AreEqual(expected, actual);

}

Листинг 15

[TestMethod]

public async Task Employeelist\_WithoutToken()

{

int actual = 0;

int expected = 500;

try

{

using (HttpClient client = new HttpClient())

{

// Нет авторизации, токен не отправляется

// Запрос на получение списка пользователей

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

actual = (int)response.StatusCode;

}

}

catch (Exception ex)

{

MessageBox.Show($"Непредвиденная ошибка: {ex.Message}");

}

// Assert

Assert.AreEqual(expected, actual);

}

}

Листинг 16

[TestMethod]

public async Task ProfitPageTests()

{

string password = "Ss1@";

string login = "sasa-ouy";

string expected = "7040";

string responseData = string.Empty;

var startDate = new DateTime(2024, 04, 29, 10, 21, 09);

var endDate = new DateTime(2024, 05, 04, 11, 12, 31);

// Авторизация и получение токена

var credentials = new { login = login, password = password };

string token = null;

using (HttpClient client = new HttpClient())

{

try

{

HttpResponseMessage response = await client.PostAsync("http://ladyaev-na.tepk-it.ru/api/login",

new StringContent(JsonConvert.SerializeObject(credentials), Encoding.UTF8, "application/json"));

response.EnsureSuccessStatusCode(); // Гарантирует, что ответ успешный

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

// Парсим ответ в объект

var responseObject = JsonConvert.DeserializeObject<dynamic>(responseBody);

token = JsonConvert.DeserializeObject<string>(responseBody);// Предполагается, что токен доступен здесь

// Сохраняем токен в настройках приложения

Token.token = token;

}

catch (Exception)

{

// Ошибка при неправильно введенных данных

MessageBox.Show("Ошибка аутентификации");

}

}

try

{

using (HttpClient client = new HttpClient())

{

// Ваш токен здесь

// Добавляем заголовок авторизации

client.DefaultRequestHeaders.Authorization = new System.Net.Http.Headers.AuthenticationHeaderValue("Bearer", token);

var parameters = new Dictionary<string, string>

{

{ "created\_at", startDate.ToString("yyyy-MM-dd HH:mm:ss") },

{ "updated\_at", endDate.ToString("yyyy-MM-dd HH:mm:ss") }

};

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

string url = $"http://ladyaev-na.tepk-it.ru/api/income?{queryString}";

HttpResponseMessage response = await client.GetAsync(url);

if (response.IsSuccessStatusCode)

{

// Обработка успешного ответа

responseData = await response.Content.ReadAsStringAsync();

// Декодирование Unicode-строки в обычную строку

responseData = Regex.Unescape(responseData);

}

else

{

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

MessageBox.Show($"Ошибка при получении дохода: {errorMessage}");

}

}

}

catch (Exception ex)

{

MessageBox.Show($"Ошибка при отправке запроса: {ex.Message}");

}

Assert.AreEqual(expected, responseData);

}

Листинг 17

[TestMethod]

public async Task ProfitPageTests\_TheSameData()

{

string password = "Ss1@";

string login = "sasa-ouy";

string expected = "0";

string responseData = string.Empty;

var startDate = new DateTime(2024, 04, 29, 10, 21, 09);

var endDate = new DateTime(2024, 04, 29, 10, 21, 09);

// Авторизация и получение токена

var credentials = new { login = login, password = password };

string token = null;

using (HttpClient client = new HttpClient())

{

try

{

HttpResponseMessage response = await client.PostAsync("http://ladyaev-na.tepk-it.ru/api/login",

new StringContent(JsonConvert.SerializeObject(credentials), Encoding.UTF8, "application/json"));

response.EnsureSuccessStatusCode(); // Гарантирует, что ответ успешный

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

// Парсим ответ в объект

var responseObject = JsonConvert.DeserializeObject<dynamic>(responseBody);

token = JsonConvert.DeserializeObject<string>(responseBody);// Предполагается, что токен доступен здесь

// Сохраняем токен в настройках приложения

Token.token = token;

}

catch (Exception)

{

// Ошибка при неправильно введенных данных

MessageBox.Show("Ошибка аутентификации");

}

}

try

{

using (HttpClient client = new HttpClient())

{

// Добавляем заголовок авторизации

client.DefaultRequestHeaders.Authorization = new System.Net.Http.Headers.AuthenticationHeaderValue("Bearer", token);

var parameters = new Dictionary<string, string>

{

{ "created\_at", startDate.ToString("yyyy-MM-dd HH:mm:ss") },

{ "updated\_at", endDate.ToString("yyyy-MM-dd HH:mm:ss") }

};

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

string url = $"http://ladyaev-na.tepk-it.ru/api/income?{queryString}";

HttpResponseMessage response = await client.GetAsync(url);

if (response.IsSuccessStatusCode)

{

// Обработка успешного ответа

responseData = await response.Content.ReadAsStringAsync();

// Декодирование Unicode-строки в обычную строку

responseData = Regex.Unescape(responseData);

}

else

{

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

MessageBox.Show($"Ошибка при получении дохода: {errorMessage}");

}

}

}

catch (Exception ex)

{

MessageBox.Show($"Ошибка при отправке запроса: {ex.Message}");

}

Assert.AreEqual(expected, responseData);

}

Листинг 18

[TestMethod]

public async Task ProfitPageTests\_WithoutToken()

{

int actual = 0;

int expected = 500;

string responseData = string.Empty;

var startDate = new DateTime(2024, 04, 29, 10, 21, 09);

var endDate = new DateTime(2024, 04, 29, 10, 21, 09);

try

{

using (HttpClient client = new HttpClient())

{

var parameters = new Dictionary<string, string>

{

{ "created\_at", startDate.ToString("yyyy-MM-dd HH:mm:ss") },

{ "updated\_at", endDate.ToString("yyyy-MM-dd HH:mm:ss") }

};

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

string url = $"http://ladyaev-na.tepk-it.ru/api/income?{queryString}";

HttpResponseMessage response = await client.GetAsync(url);

actual = (int)response.StatusCode;

}

}

catch (Exception ex)

{

MessageBox.Show($"Ошибка при отправке запроса: {ex.Message}");

}

Assert.AreEqual(expected, actual);

}

Листинг 19

[TestMethod]

public async Task OrderList\_WithoutToken()

{

int actual = 0;

int expected = 500;

try

{

using (HttpClient client = new HttpClient())

{

// Нет авторизации, токен не отправляется

// Запрос на получение списка пользователей

HttpResponseMessage response = await client.GetAsync("http://ladyaev-na.tepk-it.ru/api/showCart");

actual = (int)response.StatusCode;

}

}

catch (Exception ex)

{

MessageBox.Show($"Непредвиденная ошибка: {ex.Message}");

}

// Assert

Assert.AreEqual(expected, actual);

}

Листинг 20

[TestMethod]

public async Task OrderList\_WithToken()

{

// Arrange

string password = "Ss1@";

string login = "sasa-ouy";

bool actual = true;

bool expected = true;

var credentials = new { login = login, password = password };

string token = null;

using (HttpClient client = new HttpClient())

{

try

{

HttpResponseMessage response = await client.PostAsync("http://ladyaev-na.tepk-it.ru/api/login",

new StringContent(JsonConvert.SerializeObject(credentials), Encoding.UTF8, "application/json"));

response.EnsureSuccessStatusCode(); // Гарантирует, что ответ успешный

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

// Парсим ответ в объект

var responseObject = JsonConvert.DeserializeObject<dynamic>(responseBody);

token = JsonConvert.DeserializeObject<string>(responseBody);// Предполагается, что токен доступен здесь

// Сохраняем токен в настройках приложения

Token.token = token;

}

catch (Exception)

{

// Ошибка при неправильно введенных данных

MessageBox.Show("Ошибка аутентификации");

}

}

try

{

using (HttpClient client = new HttpClient())

{

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

// Запрос на получение списка пользователей

HttpResponseMessage response = await client.GetAsync("http://ladyaev-na.tepk-it.ru/api/showCart");

}

}

catch (Exception ex)

{

MessageBox.Show($"Непредвиденная ошибка: {ex.Message}");

actual = false;

}

// Assert

Assert.AreEqual(expected, actual);

}

Листинг 21

[TestMethod]

public async Task PopularTarifTests\_Successful\_ReturnsTrue()

{

// Arrange

string password = "Ss1@";

string login = "sasa-ouy";

List<string> tarifInfo = null;

bool actual = false;

bool expected = true;

// Авторизация и получение токена

var credentials = new { login = login, password = password };

string token = null;

using (HttpClient client = new HttpClient())

{

try

{

HttpResponseMessage response = await client.PostAsync("http://ladyaev-na.tepk-it.ru/api/login",

new StringContent(JsonConvert.SerializeObject(credentials), Encoding.UTF8, "application/json"));

response.EnsureSuccessStatusCode(); // Гарантирует, что ответ успешный

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

// Парсим ответ в объект

var responseObject = JsonConvert.DeserializeObject<dynamic>(responseBody);

token = JsonConvert.DeserializeObject<string>(responseBody);// Предполагается, что токен доступен здесь

// Сохраняем токен в настройках приложения

Token.token = token;

}

catch (Exception)

{

// Ошибка при неправильно введенных данных

MessageBox.Show("Ошибка аутентификации");

}

}

try

{

using (HttpClient client = new HttpClient())

{

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

try

{

HttpResponseMessage response = await client.GetAsync("http://ladyaev-na.tepk-it.ru/api/popular");

response.EnsureSuccessStatusCode(); // Ensure success status code

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

tarifInfo = JsonConvert.DeserializeObject<List<string>>(jsonString);

}

catch (HttpRequestException ex)

{

MessageBox.Show($"Error while requesting API: {ex.Message}");

}

}

foreach (string info in tarifInfo)

{

actual = true;

}

}

catch (Exception)

{

// Ошибка при неправильно введенных данных

actual = false;

}

Assert.AreEqual(expected, actual);

}

Листинг 22

[TestMethod]

public async Task PopularTarifTests\_WithoutToken()

{

// Arrange

int actual = 0;

int expected = 500;

try

{

using (HttpClient client = new HttpClient())

{

HttpResponseMessage response = await client.GetAsync("http://ladyaev-na.tepk-it.ru/api/popular");

actual = (int)response.StatusCode;

}

}

catch (HttpRequestException ex)

{

MessageBox.Show($"Ошибка при запросе к API: {ex.Message}");

}

catch (Exception ex)

{

MessageBox.Show($"Непредвиденная ошибка: {ex.Message}");

}

Assert.AreEqual(expected, actual);

}

Листинг 23

[TestMethod]

public async Task DeleteSouvenir\_Successful\_ReturnsTrue()

{

// Arrange

string password = "Ss1@";

string login = "sasa-ouy";

int actual = 0;

int expected = 410;

// Authenticate user and get token

var credentials = new { login = login, password = password };

string token = null;

using (HttpClient client = new HttpClient())

{

try

{

HttpResponseMessage response = await client.PostAsync("http://ladyaev-na.tepk-it.ru/api/login",

new StringContent(JsonConvert.SerializeObject(credentials), Encoding.UTF8, "application/json"));

response.EnsureSuccessStatusCode(); // Ensure response is successful

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

// Parse response into object

var responseObject = JsonConvert.DeserializeObject<dynamic>(responseBody);

token = JsonConvert.DeserializeObject<string>(responseBody); // Assuming token is available here

// Save token in application settings

Token.token = token;

}

catch (HttpRequestException ex)

{

// Authentication error

Assert.Fail($"Failed to authenticate: {ex.Message}");

}

}

try

{

// Perform operation to add a souvenir

// Here you need to add code to add a souvenir before attempting to delete it

// This depends on your application logic and how souvenirs are added

// After adding a souvenir successfully, you can proceed with deletion

int souvenirId = 19;

using (HttpClient client = new HttpClient())

{

// Add token to request header

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

// Send request to delete souvenir

HttpResponseMessage response = await client.DeleteAsync($"http://ladyaev-na.tepk-it.ru/api/deleteSouvenir/{souvenirId}");

actual = (int)response.StatusCode;

}

}

catch (Exception ex)

{

Console.WriteLine($"Error: {ex.Message}");

}

Assert.AreEqual(expected, actual);

}

Листинг 24

public async Task DeleteEmployee\_Successful\_ReturnsTrue()

{

// Arrange

string password = "Ss1@";

string login = "sasa-ouy";

int actual = 0;

int expected = 410;

// Authenticate user and get token

var credentials = new { login = login, password = password };

string token = null;

using (HttpClient client = new HttpClient())

{

try

{

HttpResponseMessage response = await client.PostAsync("http://ladyaev-na.tepk-it.ru/api/login",

new StringContent(JsonConvert.SerializeObject(credentials), Encoding.UTF8, "application/json"));

response.EnsureSuccessStatusCode(); // Ensure response is successful

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

// Parse response into object

var responseObject = JsonConvert.DeserializeObject<dynamic>(responseBody);

token = JsonConvert.DeserializeObject<string>(responseBody); // Assuming token is available here

// Save token in application settings

Token.token = token;

}

catch (HttpRequestException ex)

{

// Authentication error

Assert.Fail($"Failed to authenticate: {ex.Message}");

}

}

try

{

// Perform operation to add a souvenir

// Here you need to add code to add a souvenir before attempting to delete it

// This depends on your application logic and how souvenirs are added

// After adding a souvenir successfully, you can proceed with deletion

int userId = 43;

using (HttpClient client = new HttpClient())

{

// Add token to request header

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

// Send request to delete souvenir

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

actual = (int)response.StatusCode;

}

}

catch (Exception ex)

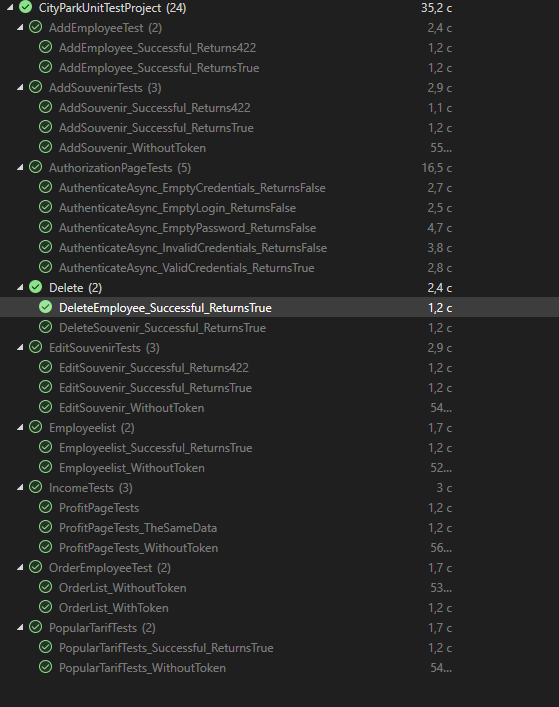
{

Console.WriteLine($"Error: {ex.Message}");

}

Assert.AreEqual(expected, actual);

}

}

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Номер теста** | **Назначение теста** | **Ожидаемый результат** | **Фактический результат** | **Статус теста** |
| [Уникальный номер тест-кейса] | [Краткое описание цели теста] | [Опишите ожидаемые результаты для каждого шага] | [Записывайте фактические результаты после проведения теста] | [Укажите, прошел ли тест успешно или не успешно] |
| TC  001 | Проверить, работоспособность добавления сотрудника | Ожидаемый результат равен true | При успешном выполнении метод возвращает true в переменную actual а она сравнивается с переменной expected которая равна ожидаемому результату | Пройден успешно (Passed) |
| TC  002 | Проверить, работоспособность добавления сотрудника, на исключение 422 Unprocessable Entity | Ожидаемый результат равен 422 | При не успешном выполнении метод возвращает код статуса в переменную actual а она сравнивается с переменной expected которая равна ожидаемому результату | Пройден успешно (Passed) |
| TC  003 | Проверить, работоспособность добавления сувенира | Ожидаемый результат равен true | При успешном выполнении метод возвращает true в переменную actual а она сравнивается с переменной expected которая равна ожидаемому результату | Пройден успешно (Passed) |
| TC  004 | Проверить, работоспособность добавления сувенира, на исключение 422 Unprocessable Entity | Ожидаемый результат равен 422 | При не успешном выполнении метод возвращает код статуса в переменную actual а она сравнивается с переменной expected которая равна ожидаемому результату | Пройден успешно (Passed) |
| TC  005 | Проверить, работоспособность добавления сувенира, на исключение 401 Unauthorized | Ожидаемый результат равен 401 | При не успешном выполнении метод возвращает код статуса в переменную actual а она сравнивается с переменной expected которая равна ожидаемому результату | Пройден успешно (Passed) |
| TC  006 | Проверить, работоспособность авторизации, при пустых значениях | Ожидаемый результат равен false, и всплывающее окно с текстом "Пожалуйста, заполните все поля". | При провальном выполнении метод выводит "Пожалуйста, заполните все поля" и возвращает false в переменную actual, а она сравнивается с переменной expected которая равна ожидаемому результату | Пройден успешно (Passed) |
| TC  007 | Проверить, работоспособность авторизации, при пустом значении пароля | Ожидаемый результат равен false, и всплывающее окно с текстом "Пожалуйста, заполните все поля". | При провальном выполнении метод выводит "Пожалуйста, заполните все поля" и возвращает false в переменную actual, а она сравнивается с переменной expected которая равна ожидаемому результату | Пройден успешно (Passed) |
| TC  008 | Проверить, работоспособность авторизации, при пустом значении логина | Ожидаемый результат равен false, и всплывающее окно с текстом "Пожалуйста, заполните все поля". | При провальном выполнении метод выводит "Пожалуйста, заполните все поля" и возвращает false в переменную actual, а она сравнивается с переменной expected которая равна ожидаемому результату | Пройден успешно (Passed) |
| TC  009 | Проверить, работоспособность авторизации, при неправильных значении пароля и логина | Ожидаемый результат равен false, и всплывающее окно с текстом "Проверьте правильность введенных данных". | При провальном выполнении метод выводит " Проверьте правильность введенных данных " и возвращает false в переменную actual, а она сравнивается с переменной expected которая равна ожидаемому результату | Пройден успешно (Passed) |
| TC  010 | Проверить, работоспособность авторизации | Ожидаемый результат равен true | При успешном выполнении метод возвращает true в переменную actual а она сравнивается с переменной expected которая равна ожидаемому результату | Пройден успешно (Passed) |
| TC  011 | Проверить, работоспособность изменения сувенира | Ожидаемый результат равен true | При успешном выполнении метод возвращает true в переменную actual а она сравнивается с переменной expected которая равна ожидаемому результату | Пройден успешно (Passed) |
| TC  012 | Проверить, работоспособность изменения сувенира, на исключение 422 Unprocessable Entity | Ожидаемый результат равен 422 | При провальном выполнении метод возвращает код статуса в переменную actual а она сравнивается с переменной expected которая равна ожидаемому результату | Пройден успешно (Passed) |
| TC  013 | Проверить, работоспособность изменения сувенира, на исключение 401 Unauthorized | Ожидаемый результат равен 401 | При провальном выполнении метод возвращает код статуса в переменную actual а она сравнивается с переменной expected которая равна ожидаемому результату | Пройден успешно (Passed) |
| TC  014 | Проверить, работоспособность вывода работников | Ожидаемый результат равен true | При успешном выполнении метод возвращает true в переменную actual а она сравнивается с переменной expected которая равна ожидаемому результату | Пройден успешно (Passed) |
| TC  015 | Проверить, работоспособность вывода работников, на исключение 500 Internal Server Error | Ожидаемый результат равен 500 | При провальном выполнении метод возвращает код статуса в переменную actual а она сравнивается с переменной expected которая равна ожидаемому результату | Пройден успешно (Passed) |
| TC  016 | Проверить, работоспособность вывода прибыли | Ожидаемый результат равен 7040, за даты 29.04.2024 10:21:09 и 04.05.2024 11:12:31 | При успешном выполнении метод возвращает количество дохода в переменную responseData, а она сравнивается с переменной expected которая равна ожидаемому результату | Пройден успешно (Passed) |
| TC  017 | Проверить, работоспособность вывода прибыли если выбраны одинаковые данные | Ожидаемый результат равен 0, за даты 29.04.2024 10:21:09 и 29.04.2024 10:21:09 | При успешном выполнении метод возвращает количество дохода в переменную responseData а она сравнивается с переменной expected которая равна ожидаемому результату | Пройден успешно (Passed) |
| TC  018 | Проверить, работоспособность вывода прибыли, на исключение 500 Internal Server Error | Ожидаемый результат равен 500 | При провальном выполнении метод возвращает код статуса в переменную actual а она сравнивается с переменной expected которая равна ожидаемому результату | Пройден успешно (Passed) |
| TC  019 | Проверить, работоспособность вывода заказов, на исключение 500 Internal Server Error | Ожидаемый результат равен 500 | При провальном выполнении метод возвращает код статуса в переменную actual а она сравнивается с переменной expected которая равна ожидаемому результату | Пройден успешно (Passed) |
| TC  020 | Проверить, работоспособность вывода заказов | Ожидаемый результат равен true | При успешном выполнении метод возвращает true в переменную actual а она сравнивается с переменной expected которая равна ожидаемому результату | Пройден успешно (Passed) |
| TC  021 | Проверить, работоспособность вывода популярного тарифа | Ожидаемый результат равен true | При успешном выполнении метод возвращает true в переменную actual а она сравнивается с переменной expected которая равна ожидаемому результату | Пройден успешно (Passed) |
| TC  022 | Проверить, работоспособность вывода популярного тарифа, на исключение 500 Internal Server Error | Ожидаемый результат равен 500 | При провальном выполнении метод возвращает код статуса в переменную actual а она сравнивается с переменной expected которая равна ожидаемому результату | Пройден успешно (Passed) |
| TC  023 | Проверить, работоспособность удаления сотрудника | Ожидаемый результат равен 410 | При успешном выполнении метод возвращает код статуса в переменную actual а она сравнивается с переменной expected которая равна ожидаемому результату | Пройден успешно (Passed) |
| TC  024 | Проверить, работоспособность удаления сувенира | Ожидаемый результат равен 410 | При успешном выполнении метод возвращает код статуса в переменную actual а она сравнивается с переменной expected которая равна ожидаемому результату | Пройден успешно (Passed) |

## ВЫВОД ТЕСТИРОВАНИЕ

После проведения тестирования приложения были сделаны следующие выводы:

* Все функциональные возможности, описанные в техническом задании, были успешно реализованы.
* Приложение демонстрирует стабильную работу и отзывчивость интерфейса.
* Не выявлено критических ошибок или сбоев при использовании приложения.
* Приложение соответствует заявленным требованиям к функциональности и реализации.

Таким образом, на основании результатов тестирования можно сделать вывод о успешной реализации поставленных задач и готовности приложения к внедрению в рабочую среду Городского сада Аттракциона.

# ЗАКЛЮЧЕНИЕ

Разработка приложения "Городской сад аттракционов" на основе технологии WPF представляет собой значимый шаг в совершенствовании управления и повышении прибыли аттракционного парка. Внедрение современных инструментов автоматизации и цифровизации в данную сферу деятельности не только оптимизирует рабочие процессы, но и способствует более комфортному и эффективному взаимодействию персонала с информацией.

Создание удобного и функционального интерфейса приложения, а также его тестирование позволят обеспечить удовлетворение потребностей как персонала аттракционного парка, так и посетителей. Приложение будет способствовать более оперативному управлению заказами сувениров, контролю дохода и эффективному распределению персонала, что в итоге приведет к повышению производительности и максимизации прибыли.

Таким образом, разработка приложения "Городской сад аттракционов" является актуальной и важной задачей, направленной на совершенствование управления аттракционным парком в условиях современного бизнеса.