**Агентство недвижимости**

Описание предметной области

Основной целью данного проекта является разработка программного модуля для управления агентством недвижимости. Это приложение должно обеспечивать пользователям (агентам, клиентам и администраторам) удобный интерфейс для добавления, поиска и управления объявлениями о недвижимости, а также взаимодействия между участниками.

Программный продукт должен позволить агентству недвижимости эффективно управлять своим портфолио объектов, обеспечивая высокий уровень удобства и информативности для пользователей.

Основные функции и возможности модуля включают:

1. Регистрация и авторизация: Пользователи могут создавать учетные записи и входить в систему в зависимости от своей роли (клиент, агент, администратор).

2. Добавление объектов недвижимости: Агенты могут добавлять новые объявления с указанием всех необходимых параметров (тип недвижимости, цена, описание, фотографии).

3. Поиск объектов: Пользователи могут искать объекты по различным критериям, таким как цена, местоположение, тип и другие параметры.

4. Управление объявлениями: Возможность редактирования и удаления собственных объявлений.

5. Взаимодействие между клиентами и агентами: Клиенты могут оставлять заявки на просмотр объектов, а агенты могут отвечать на них.

6. Статистика и отчетность: Агенты и администраторы могут видеть статистику по продажам и просмотрам объектов.

Техническое задание

**1. Общие сведения**

1.1. Наименование проекта: Разработка программного модуля для управления агентством недвижимости.

1.2. Заказчик: Агентство недвижимости "Комфорт".

1.3. Исполнитель: Команда разработчиков "ТехноСервис".

**2. Функциональные требования**

2.1. Регистрация пользователей:

- Поля: имя, фамилия, электронная почта, пароль, роль (клиент, агент).

- Валидация данных и подтверждение по email.

2.2. Авторизация:

- Логин и пароль.

2.3. Добавление объектов недвижимости:

- Поля: тип недвижимости, цена, адрес, описание, фотографии.

- Возможность указания статуса (доступен, продан).

2.4. Поиск объектов:

- Поля для фильтрации: цена, тип, местоположение, количество комнат.

2.5. Управление объявлениями:

- Возможность редактирования и удаления собственных объявлений.

- Отображение списка всех объявлений агента.

2.6. Взаимодействие с клиентами:

- Клиенты могут отправлять заявки на просмотр объектов.

- Агенты могут отвечать на заявки и оставлять комментарии.

2.7. Статистика и отчетность:

- Просмотр статистики по количеству объявлений, сделок и запросов на просмотр.

**3. Нефункциональные требования**

3.1. Кроссплатформенность:

- Поддержка работы на ОС семейства Windows.

3.2. Безопасность:

- Шифрование данных при передаче.

- Защита от несанкционированного доступа.

3.3. Удобство использования:

- Интуитивно понятный интерфейс.

- Информативные уведомления и подсказки.

3.4. Производительность:

- Быстрый доступ к данным.

- Минимальное время отклика на запросы пользователя.

**4. Требования к реализации**

4.1. Язык программирования: на усмотрение разработчика (C#).

4.2. СУБД: на усмотрение разработчика (MS SQL).

**5. Требования к документации**

5.1. Техническое задание на разработку программного модуля.

**6. Руководство по стилю**

6.1. Шрифт: Arial, 14pt.

6.2. Цветовая схема: светлый фон (белый), цвет текста - темно-синий; кнопки - зеленый с белым текстом.

**Задание 2. Разработка алгоритмов и диаграмм**



**Диаграмма вариантов использования**



**Диаграмма последовательности (Сохранение объекта)**



**Диаграмма активности (Сохранение объекта)**



Таблица Заявка

|  |  |  |
| --- | --- | --- |
| Поле | Тип данных | Значения NULL |
| ID (Первичный ключ) | int | Запрещены |
| ID\_клиента | int | Разрешены |
| ID\_объекта | int | Разрешены |
| Статус | nvarchar(20) | Разрешены |
| Комментарии | nvarchar(MAX) | Разрешены |

Таблица Объект недвижимости

|  |  |  |
| --- | --- | --- |
| Поле | Тип данных | Значения NULL |
| ID (Первичный ключ) | int | Запрещены |
| Тип | nvarchar(50) | Разрешены |
| Цена | decimal(10, 2) | Разрешены |
| Адрес | nvarchar(255) | Разрешены |
| Описание | nvarchar(MAX) | Разрешены |
| Фотографии | nvarchar(MAX) | Разрешены |
| Статус | nvarchar(20) | Разрешены |
| ID\_агента | int | Разрешены |

Таблица Пользователей

|  |  |  |
| --- | --- | --- |
| Поле | Тип данных | Значения NULL |
| ID (Первичный ключ) | int | Запрещены |
| Имя | nvarchar(50) | Разрешены |
| Фамилия | nvarchar(50) | Разрешены |
| Email | nvarchar(100) | Разрешены |
| Пароль | nvarchar(255) | Разрешены |
| Роль | nvarchar(20) | Разрешены |

Таблица Статистика

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Поле | Тип данных | | Значения NULL | |
| ID (Первичный ключ) | | int | | Запрещены |
| ID\_агента | | int | | Разрешены |
| Количество\_объявлений | | int | | Разрешены |
| Количество\_сделок | | int | | Разрешены |
| Количество\_заявок | | int | | Разрешены |

Таблица Заявка

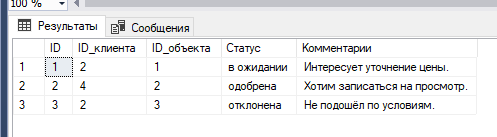


Таблица Объект недвижимости

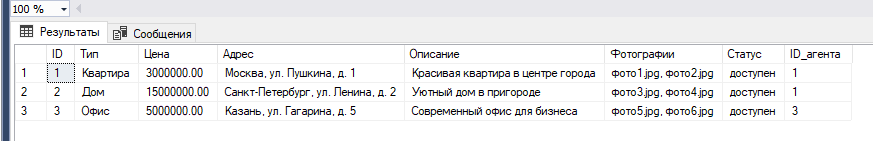


Таблица Пользователей

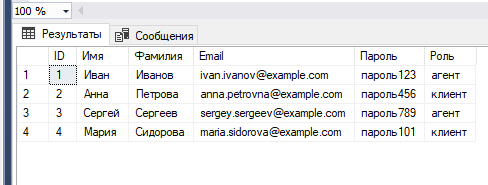
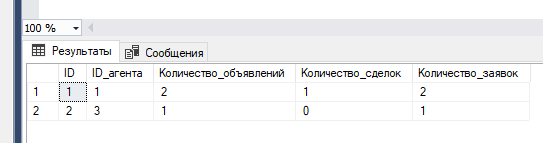
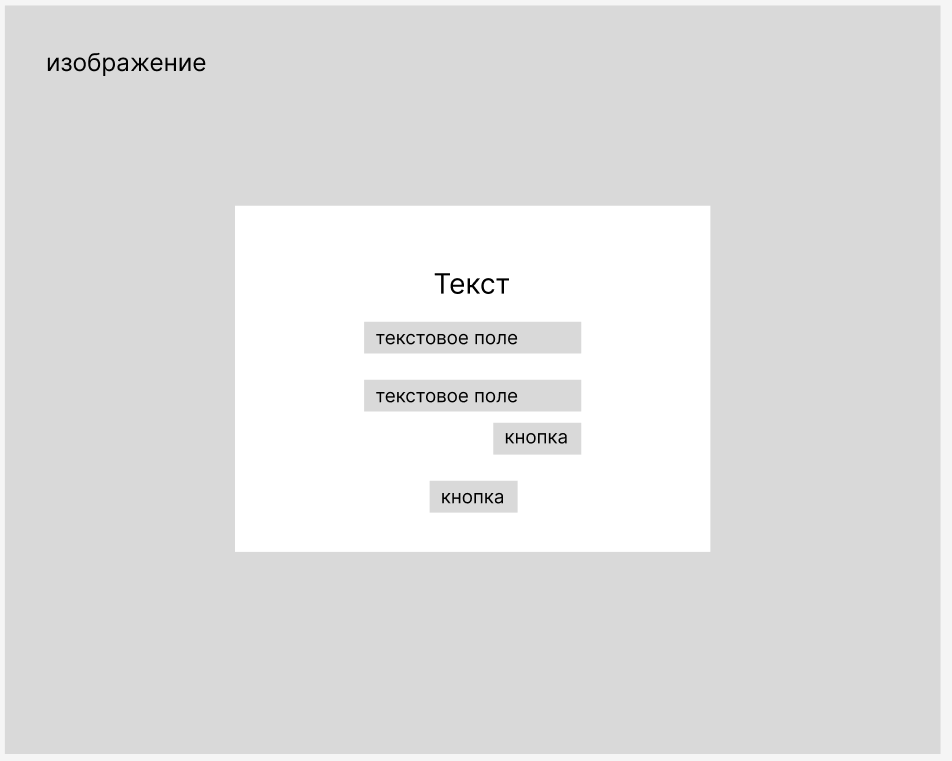


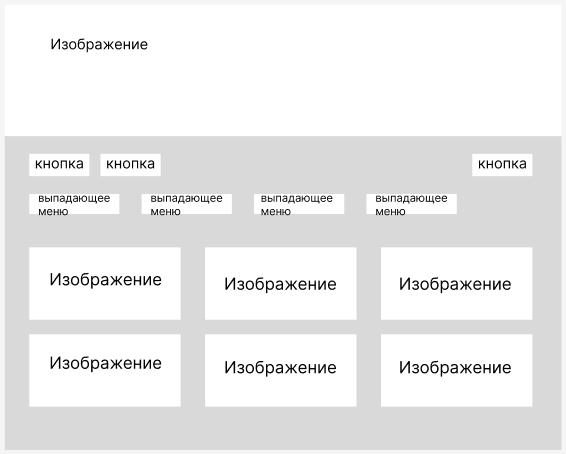
Таблица Статистика



**Задание 3. Разработка макетов программы**



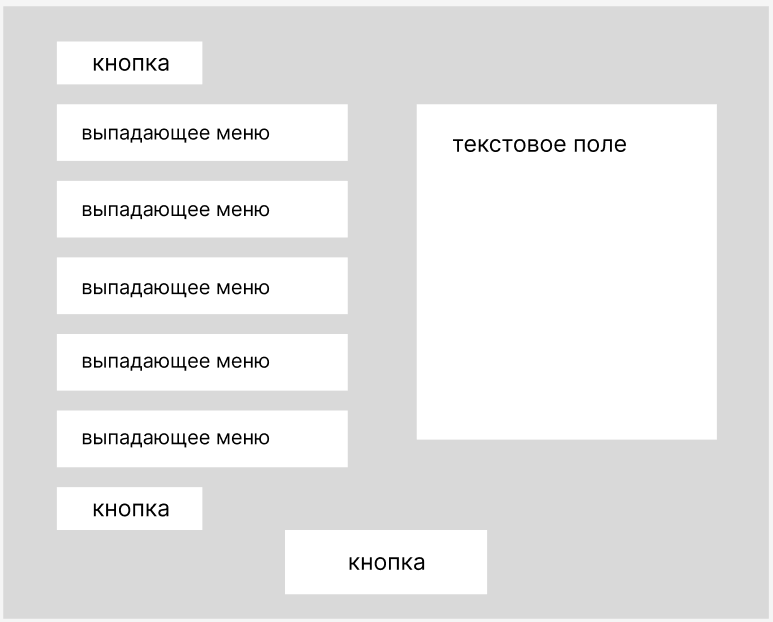
Wireframe макет авторизации/входа



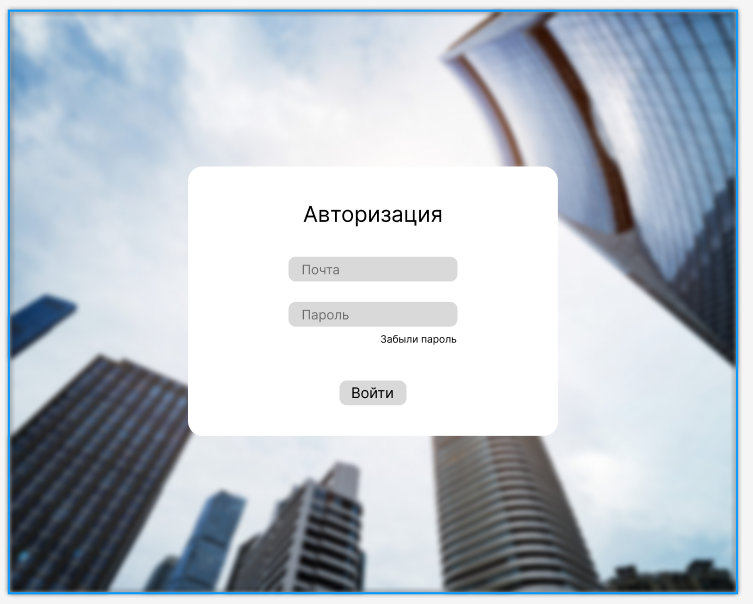
Wireframe макет главной страницы



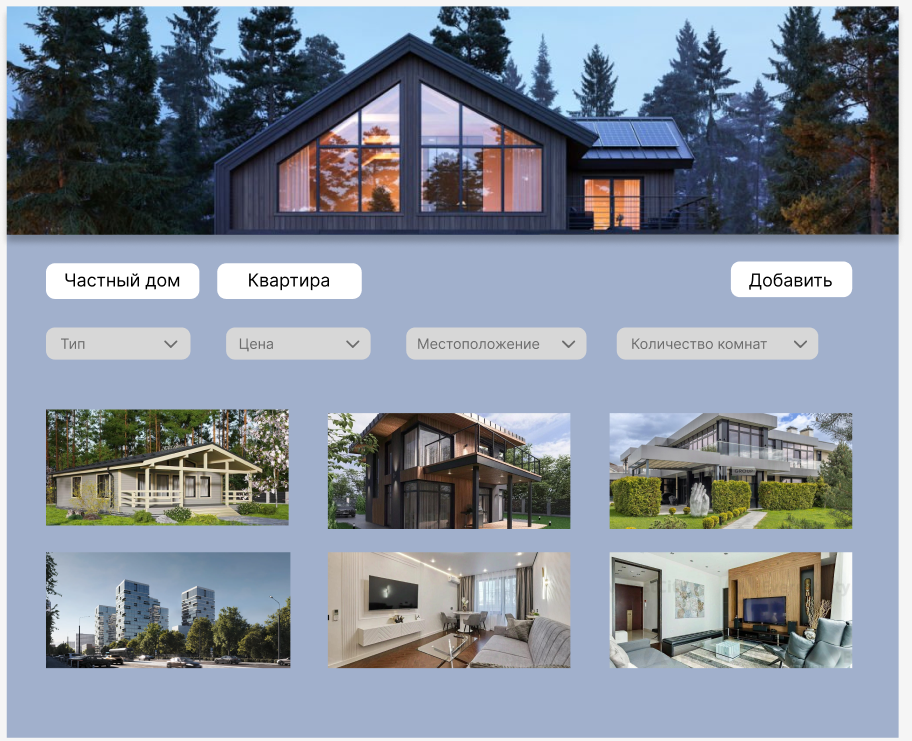
Wireframe макет объекта



Wireframe макет для добавления объекта



Mockup макет авторизации/входа



Mockup макет главной страницы



Mockup макет объекта



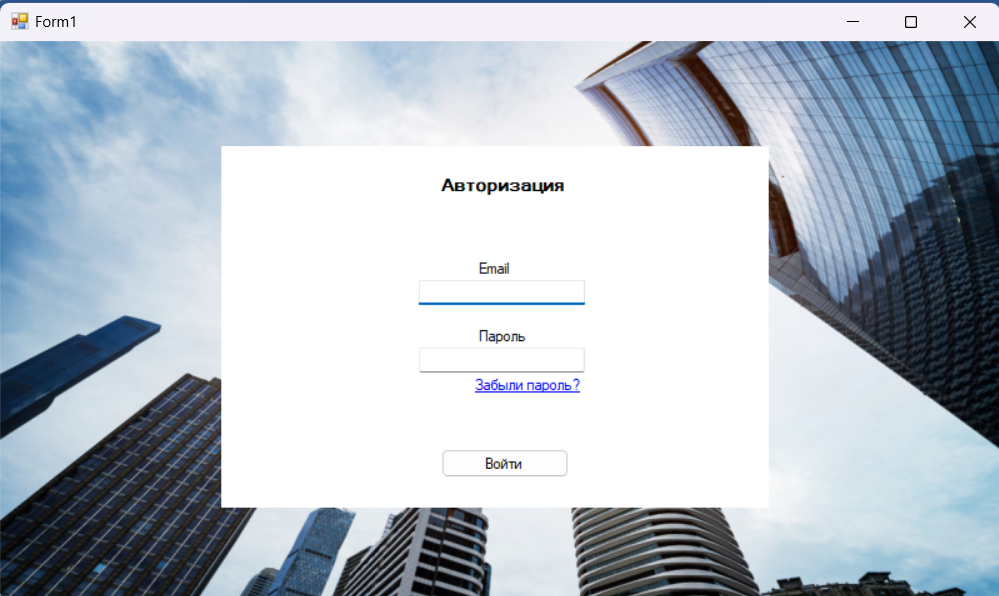
Mockup макет для добавления объекта



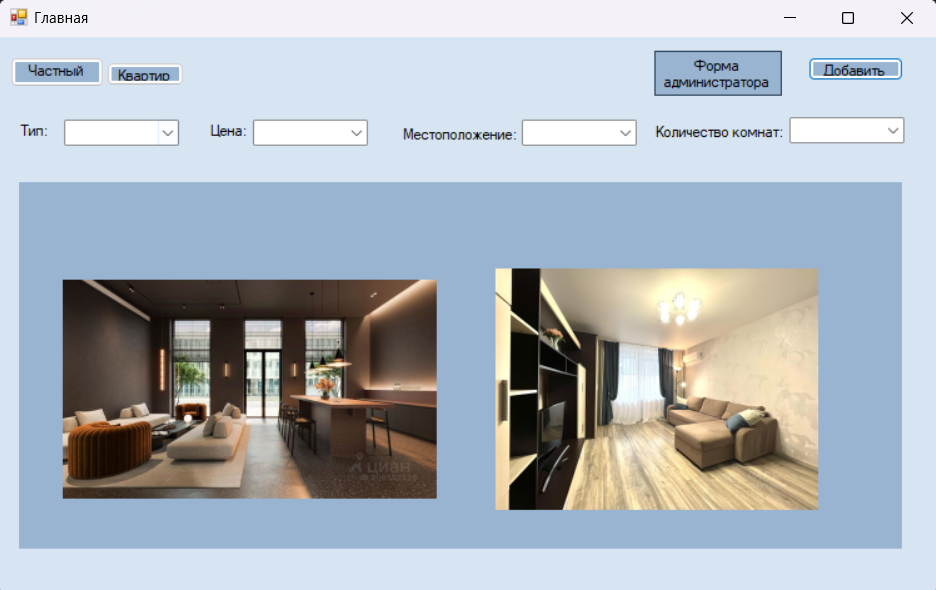
Карта навигации

**Задание 4. Разработка программы**

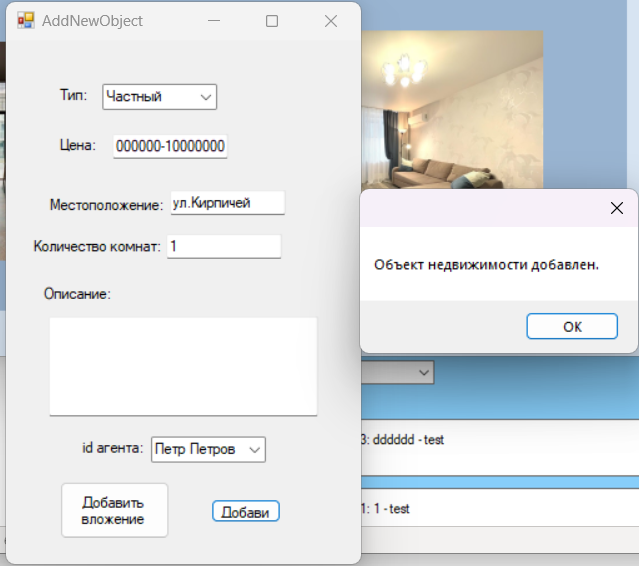
Изначально пользователю нужно войти в систему:



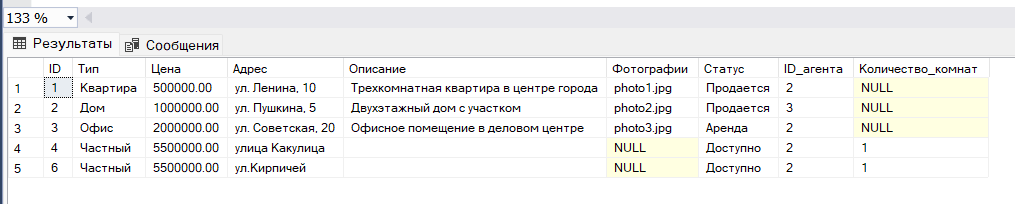
Пользователь попадает на главную страницу со всеми объектами недвижимости:



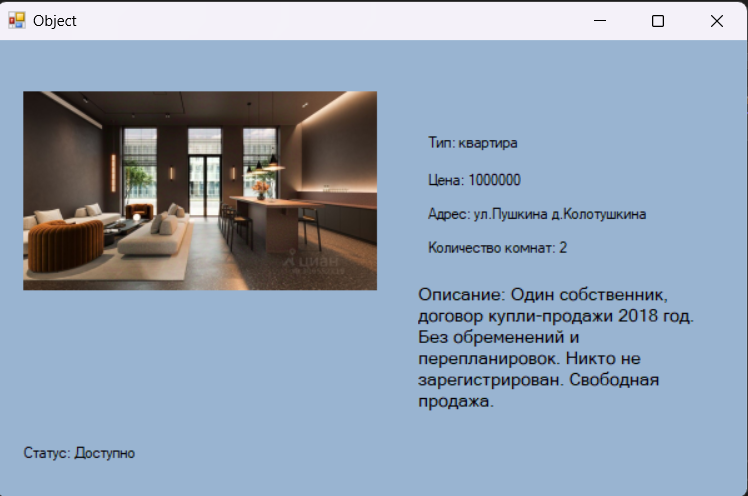
Добавление объекта недвижимости:



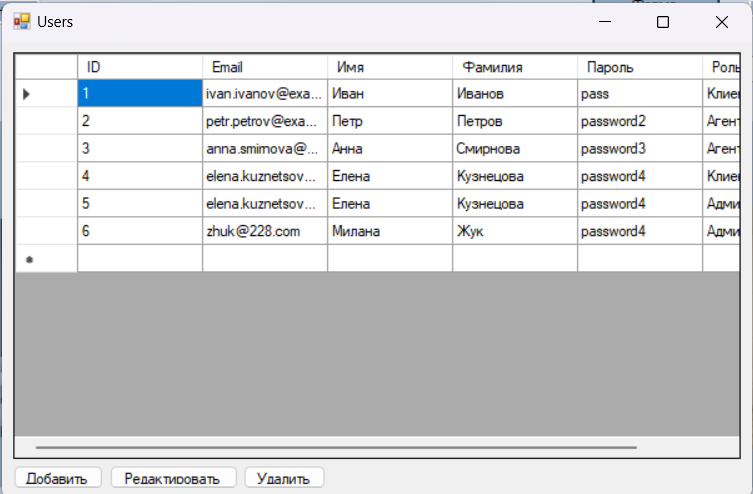
Отображение нового объекта в SQL:



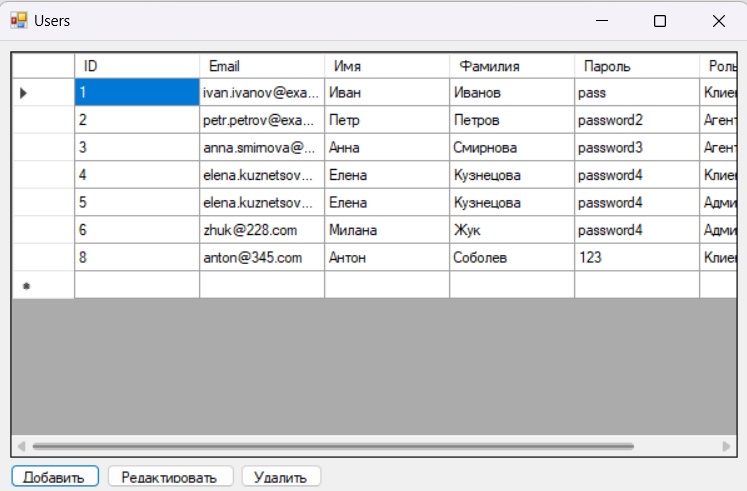
Страница объекта недвижимости:

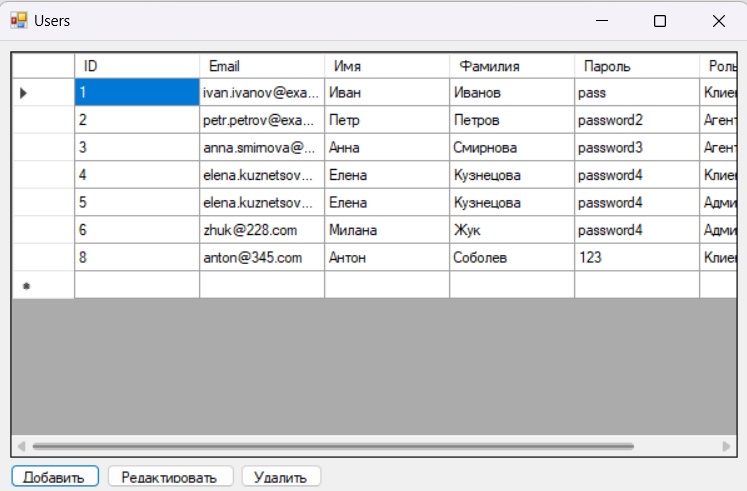


Заходим в форму администратора:

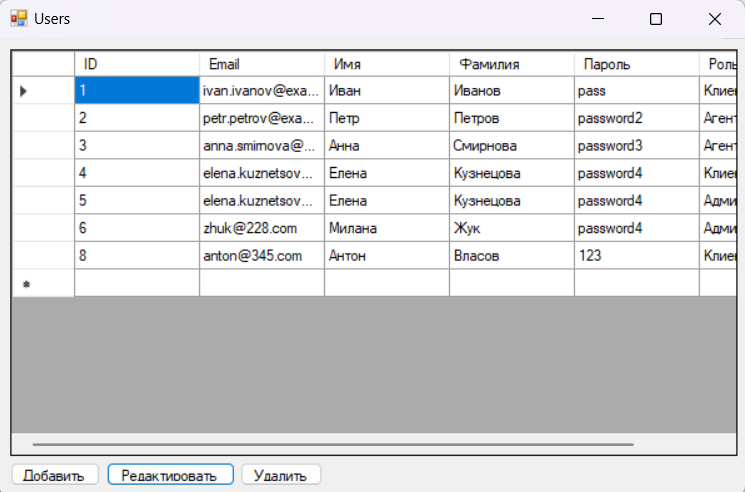


Добавление клиента:

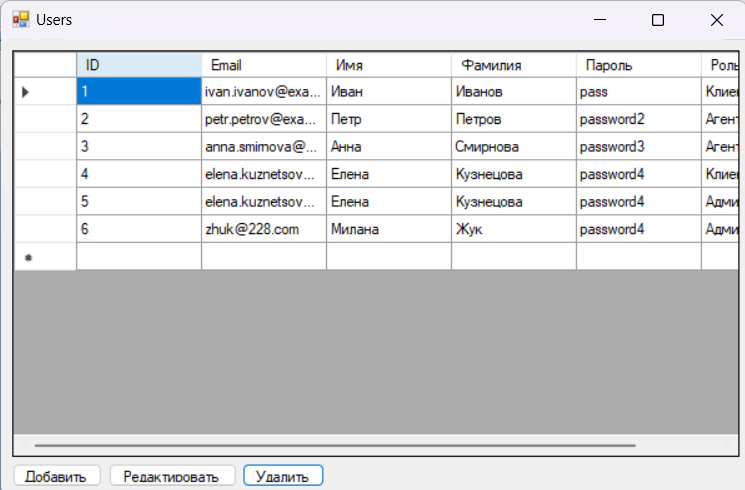




Изменение клиента:



Удаление клиента:



Задание 5. Тестирование

Аннотация теста:

|  |  |
| --- | --- |
| **Поле** | **Описание** |
| **Название проекта** | Агентство недвижимости |
| **Рабочая версия** | 1.0 |
| **Имя тестирующего** | Жук Милана |
| **Дата(ы) теста** | 16.10.2024 |

5 Test-Case ( Таблица 1 – 5):

**Таблица 1 - Тестовый случай для успешной аутентификации**

|  |  |
| --- | --- |
| **Поле** | **Описание** |
| **Тестовый пример #** | TC\_UI\_1 |
| **Приоритет тестирования**  *(Низкий/Средний/Высокий)* | Высокий |
| **Заголовок/название теста** | Успешная аутентификация |
| **Краткое изложение теста** | Проверка успешной аутентификации для действующего пользователя. |
| **Этапы теста** | 1. Установить email: [ivanov@mail.com](mailto:ivanov@mail.com)  2. Установить пароль: password123  3. Нажать кнопку "Войти" |
| **Тестовые данные** | Email: [ivanov@mail.com](mailto:ivanov@mail.com)  Пароль: password123 |
| **Ожидаемый результат** | Показано сообщение о успешной аутентификации и открыта форма MainForm |
| **Фактический результат** | Показано сообщение о успешной аутентификации и открыта форма MainForm |
| **Предварительное условие** | Пользователь с логином [ivanov@mail.com](mailto:ivanov@mail.com) существует в базе данных |
| **Постусловие** | Аутентификация прошла успешно |
| **Статус**  *(Зачет/Незачет)* | Зачет |

**Таблица 2 - Тестовый случай для обработки неверных учетных данных**

|  |  |
| --- | --- |
| **Поле** | **Описание** |
| **Тестовый пример #** | TC\_UI\_2 |
| **Приоритет тестирования**  *(Низкий/Средний/Высокий)* | Высокий |
| **Заголовок/название теста** | Неверные учетные данные |
| **Краткое изложение теста** | Проверка отображения сообщения об ошибке при неправильных учетных данных |
| **Этапы теста** | 1. Установить email: [invalid\_email@example.com](mailto:invalid_email@example.com)  2. Установить пароль: incorrect\_password  3. Нажать кнопку "Войти" |
| **Тестовые данные** | Email: [invalid\_email@example.com](mailto:invalid_email@example.com)  Пароль: incorrect\_password |
| **Ожидаемый результат** | Показано сообщение о том, что пользователь не найден |
| **Фактический результат** | Показано сообщение о том, что пользователь не найден |
| **Предварительное условие** | Пользователь с указанными данными не существует в базе данных. |
| **Постусловие** | Аутентификация не прошла |
| **Статус**  *(Зачет/Незачет)* | Зачет |

**Тестовый случай для пустых полей в форме аунтификации**

|  |  |
| --- | --- |
| **Поле** | **Описание** |
| **Тестовый пример #** | TC\_UI\_3 |
| **Приоритет тестирования**  *(Низкий/Средний/Высокий)* | Высокий |
| **Заголовок/название теста** | Пустые поля в форме входа |
| **Краткое изложение теста** | Проверка отображения сообщения об ошибке при пустых полях для email и пароля |
| **Этапы теста** | 1. Оставить email: (пусто) 2. Оставить пароль: (пусто) 3. Нажать кнопку "Войти" |
| **Тестовые данные** | Email: (пусто)  Пароль: (пусто) |
| **Ожидаемый результат** | Показано сообщение об ошибке "Email не может быть пустым". |
| **Фактический результат** | Показано сообщение об ошибке "Email не может быть пустым". |
| **Предварительное условие** | Поля email и пароля пусты. |
| **Постусловие** | Аутентификация не прошла |
| **Статус**  *(Зачет/Незачет)* | Зачет |

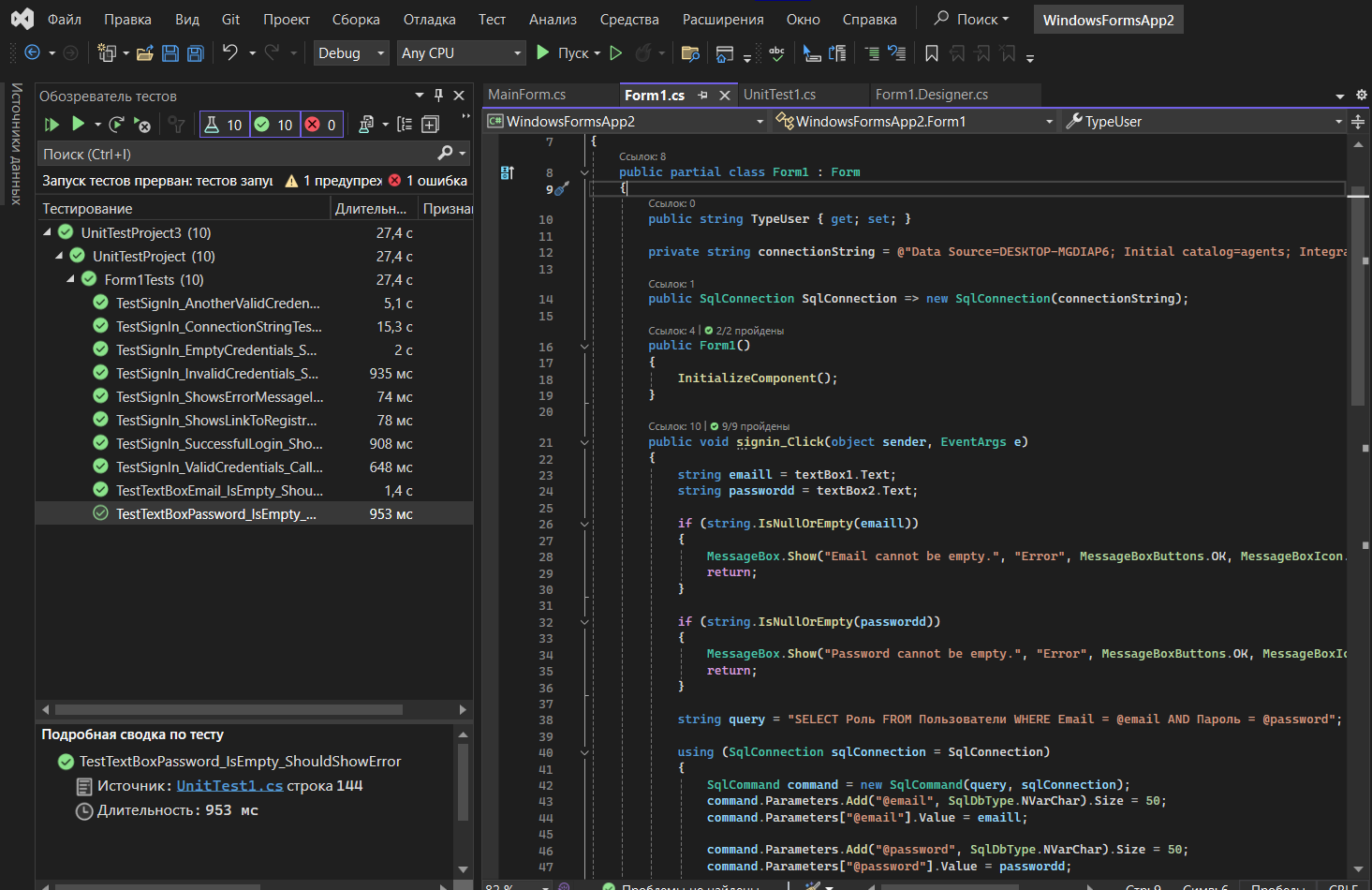
**Таблица 4 - Тестовый случай для обработки исключения при неверной строке подключения**

|  |  |
| --- | --- |
| **Поле** | **Описание** |
| **Тестовый пример #** | TC\_UI\_4 |
| **Приоритет тестирования**  *(Низкий/Средний/Высокий)* | Высокий |
| **Заголовок/название теста** | Неверная строка подключения |
| **Краткое изложение теста** | Проверка отображения сообщения об ошибке при неверной строке подключения |
| **Этапы теста** | 1. Установить email: [ivanov@mail.com](mailto:ivanov@mail.com)  2. Установить пароль: password123  3. Неверная строка подключения  4. Нажать кнопку "Войти" |
| **Тестовые данные** | Email: [ivanov@mail.com](mailto:ivanov@mail.com)  Пароль: password123 |
| **Ожидаемый результат** | Исключение SqlException |
| **Фактический результат** | Исключение SqlException |
| **Предварительное условие** | Проверка на управление неверной строкой подключения. |
| **Постусловие** | Аутентификация не прошла. |
| **Статус**  *(Зачет/Незачет)* | Зачет |

**Таблица 5 - Тестовый случай проверки ссылки на регистрацию**

|  |  |
| --- | --- |
| **Поле** | **Описание** |
| **Тестовый пример #** | TC\_UI\_5 |
| **Приоритет тестирования**  *(Низкий/Средний/Высокий)* | Высокий |
| **Заголовок/название теста** | Проверка ссылки на регистрацию |
| **Краткое изложение теста** | Проверка, что при нажатии на ссылку "Зарегистрироваться" открывается окно регистрации. |
| Проверка, что при нажатии на ссылку "Зарегистрироваться" открывается окно регистрации. **Этапы теста** | 1. Нажать на ссылку "Зарегистрироваться". |
| **Тестовые данные** | Н/A |
| **Ожидаемый результат** | Открывается форма регистрации (Form2). |
| **Фактический результат** | Открывается форма регистрации (Form2). |
| **Предварительное условие** | Форма входа отображается. |
| **Постусловие** | Открыта форма регистрации. |
| **Статус**  *(Зачет/Незачет)* | Зачет |

Результат выполнения UnitTestProject1:



Код UnitTest1:

using Microsoft.VisualStudio.TestTools.UnitTesting;

using System;

using System.Data.SqlClient;

using System.Linq;

using System.Windows.Forms;

using WindowsFormsApp2;

namespace UnitTestProject

{

[TestClass]

public class Form1Tests

{

private Form1 form1;

private string connectionString = @"Data Source=SMAGINAANYAA\MSSQLSERVER01; Initial catalog=agents; Integrated Security=True";

[TestInitialize]

public void TestInitialize()

{

form1 = new Form1();

form1.Show(); // Открытие формы перед каждым тестом

}

[TestMethod]

public void TestSignIn\_SuccessfulLogin\_ShowsMainForm()

{

// Arrange

var validEmail = "ivanov@mail.com"; // Учётные данные из базы данных

var validPassword = "password123"; // Соответствующий пароль

var form1 = new Form1();

form1.SetConnectionString("Data Source=SMAGINAANYAA\\MSSQLSERVER01; Initial catalog=agents; Integrated Security=True");

form1.textBox1.Text = validEmail;

form1.textBox2.Text = validPassword;

// Act

form1.signin\_Click(this, EventArgs.Empty);

// Assert

var mainForm = Application.OpenForms.OfType<MainForm>().FirstOrDefault(); // Получаем открывшуюся форму MainForm

Assert.IsNotNull(mainForm); // Убедитесь, что MainForm открыта

Assert.AreEqual("Админ", mainForm.TypeUser); // Проверьте установленное значение (роль)

}

[TestMethod]

public void TestSignIn\_InvalidCredentials\_ShowsErrorMessage()

{

// Arrange

form1.textBox1.Text = "invalid\_email@example.com"; // Эмулируем неправильный email

form1.textBox2.Text = "incorrect\_password"; // Эмулируем неправильный пароль

// Act

form1.signin\_Click(null, null); // Вызываем клик по кнопке

// Assert

// Проверяем, что появляется сообщение об ошибке

var result = MessageBox.Show("Пользователь не найден.", "Error", MessageBoxButtons.OK, MessageBoxIcon.Error);

Assert.AreEqual(DialogResult.OK, result);

}

[TestMethod]

public void TestSignIn\_ValidCredentials\_CallsExecuteScalar()

{

// Arrange

form1.textBox1.Text = "ivanov@mail.com";

form1.textBox2.Text = "password123";

using (SqlConnection connection = new SqlConnection(connectionString))

{

string query = "SELECT Роль FROM Пользователи WHERE Email = @email AND Пароль = @password";

SqlCommand command = new SqlCommand(query, connection);

command.Parameters.AddWithValue("@email", form1.textBox1.Text);

command.Parameters.AddWithValue("@password", form1.textBox2.Text);

connection.Open();

object result = command.ExecuteScalar();

connection.Close();

// Act

form1.signin\_Click(null, null); // Вызываем клик по кнопке

// Assert

Assert.IsNotNull(result); // Проверяем, что SQL-запрос вернул не null

}

}

[TestMethod]

public void TestSignIn\_ShowsErrorMessageIfNoConnection()

{

// Arrange

form1.textBox1.Text = "ivanov@mail.com";

form1.textBox2.Text = "password123";

// Прерывание соединения

string invalidConnectionString = @"Data Source=INVALID\_SERVER; Initial catalog=agents; Integrated Security=True";

form1.SetConnectionString(invalidConnectionString);

// Act & Assert

try

{

form1.signin\_Click(null, null); // Вызываем клик по кнопке

Assert.Fail("Expected exception due to invalid connection");

}

catch (SqlException)

{

Assert.IsTrue(true); // Ожидаем исключение

}

}

[TestMethod]

public void TestSignIn\_ShowsLinkToRegistrationForm()

{

// Arrange

var linkLabel = new LinkLabel();

linkLabel.Text = "Зарегистрироваться";

linkLabel.LinkClicked += form1.linkLabel1\_LinkClicked; // Привязываем обработчик события

// Создаем объект события LinkLabelLinkClickedEventArgs

var eventArgs = new LinkLabelLinkClickedEventArgs(linkLabel.Links[0]);

// Act

// Вручную вызываем обработчик события

form1.linkLabel1\_LinkClicked(linkLabel, eventArgs);

// Assert

// Проверка, что открылся Form2

Assert.IsTrue(Application.OpenForms.OfType<Form2>().Any());

}

[TestMethod]

public void TestTextBoxEmail\_IsEmpty\_ShouldShowError()

{

// Arrange

form1.textBox1.Text = string.Empty; // Пустое поле email

form1.textBox2.Text = "correct\_password";

// Act

form1.signin\_Click(null, null); // Вызываем клик по кнопке

// Assert

var result = MessageBox.Show("Email cannot be empty.", "Error", MessageBoxButtons.OK, MessageBoxIcon.Error);

Assert.AreEqual(DialogResult.OK, result);

}

[TestMethod]

public void TestTextBoxPassword\_IsEmpty\_ShouldShowError()

{

// Arrange

form1.textBox1.Text = "ivanov@mail.com";

form1.textBox2.Text = string.Empty; // Пустое поле пароля

// Act

form1.signin\_Click(null, null); // Вызываем клик по кнопке

// Assert

var result = MessageBox.Show("Password cannot be empty.", "Error", MessageBoxButtons.OK, MessageBoxIcon.Error);

Assert.AreEqual(DialogResult.OK, result);

}

[TestMethod]

public void TestSignIn\_ConnectionStringTest\_ThrowsExceptionIfInvalid()

{

// Arrange

form1.textBox1.Text = "ivanov@mail.com";

form1.textBox2.Text = "password123";

// Имитируем невалидную строку подключения

string invalidConnectionString = @"Data Source=INVALID\_SERVER; Initial catalog=agents; Integrated Security=True";

form1.SetConnectionString(invalidConnectionString);

// Act & Assert

try

{

form1.signin\_Click(null, null); // Вызываем клик по кнопке

Assert.Fail("Expected SqlException due to invalid connection");

}

catch (SqlException)

{

Assert.IsTrue(true); // Ожидаем исключение

}

}

[TestMethod]

public void TestSignIn\_EmptyCredentials\_ShouldShowErrorMessages()

{

// Arrange

form1.textBox1.Text = string.Empty; // Пустое поле email

form1.textBox2.Text = string.Empty; // Пустое поле пароля

// Act

form1.signin\_Click(null, null); // Вызываем клик по кнопке

// Assert

var resultEmail = MessageBox.Show("Email cannot be empty.", "Error", MessageBoxButtons.OK, MessageBoxIcon.Error);

var resultPassword = MessageBox.Show("Password cannot be empty.", "Error", MessageBoxButtons.OK, MessageBoxIcon.Error);

Assert.AreEqual(DialogResult.OK, resultEmail);

Assert.AreEqual(DialogResult.OK, resultPassword);

}

[TestMethod]

public void TestSignIn\_AnotherValidCredentials\_ShowsMainForm()

{

// Arrange

var validEmail = "petrova@mail.com"; // Учётные данные из базы данных

var validPassword = "password456"; // Соответствующий пароль

var form1 = new Form1();

form1.SetConnectionString("Data Source=SMAGINAANYAA\\MSSQLSERVER01; Initial catalog=agents; Integrated Security=True");

form1.textBox1.Text = validEmail;

form1.textBox2.Text = validPassword;

// Act

form1.signin\_Click(this, EventArgs.Empty);

// Assert

var mainForm = Application.OpenForms.OfType<MainForm>().FirstOrDefault(); // Получаем открывшуюся форму MainForm

Assert.IsNotNull(mainForm); // Убедитесь, что MainForm открыта

Assert.AreEqual("Агент", mainForm.TypeUser); // Проверьте установленное значение (роль)

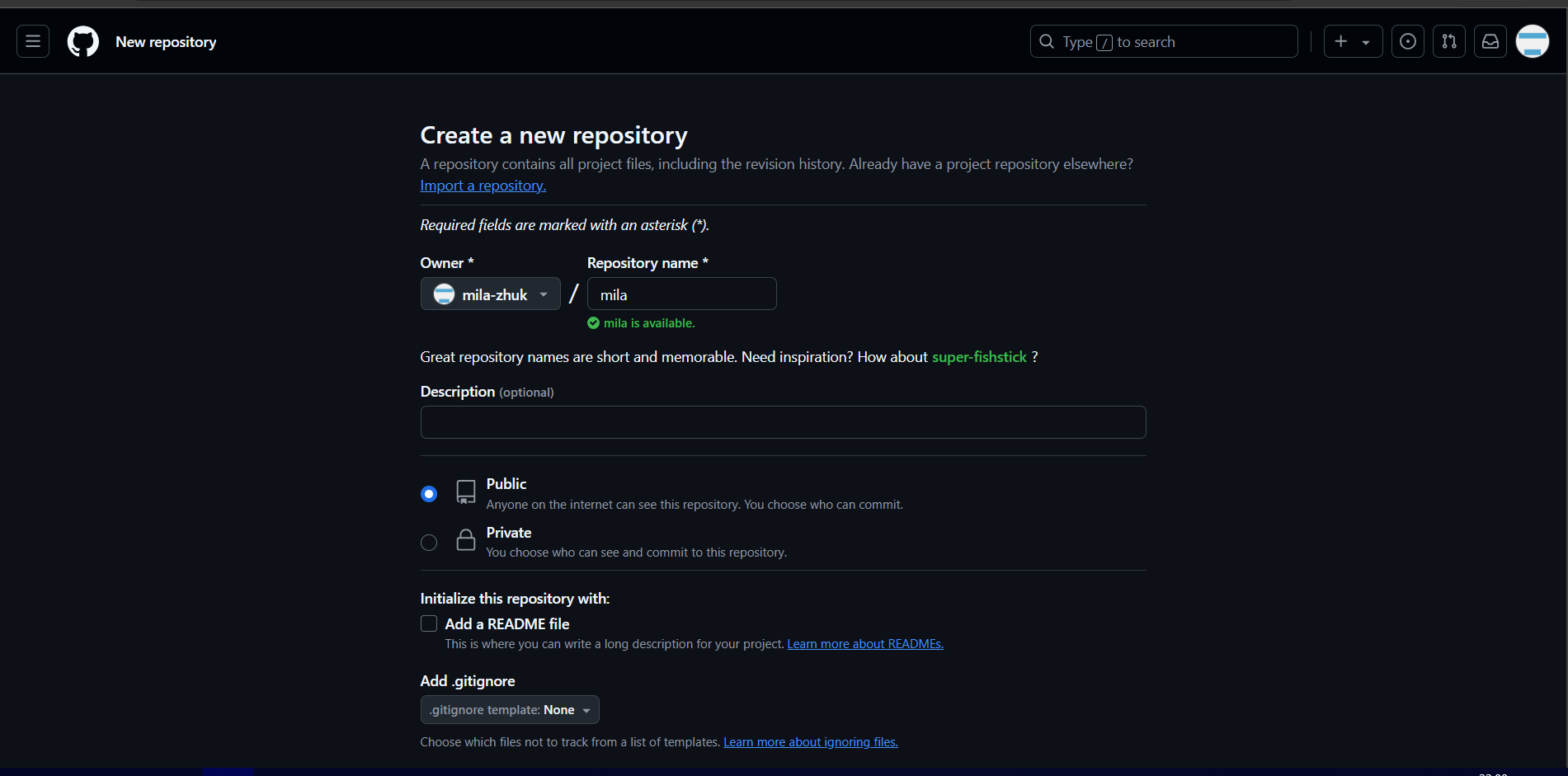
}

}

}

Задание 6. GitHub. Оценка проекта

Отправление приложения с тестами и отчетом на созданный репозиторий на GitHub.

****

Создание репозитория