**МИНОБРНАУКИ РОССИИ**

федеральное государственное автономное образовательное учреждение

высшего образования

«Санкт-Петербургский политехнический университет Петра Великого»

(ФГАОУ ВО «СПбПУ»)

**Институт среднего профессионального образования**

**ОТЧЕТ**

**по учебной практике УП.02.01 (по профилю специальности)**

по профессиональному модулю ПМ.02 «Осуществление интеграции программных модулей»

(код и наименование)

Специальность09.02.07 Информационные системы и программирование

(код и наименование специальности)

Студент(ка) 4 курса 42919/7 группы

Кащаев Никита Витальевич

(Фамилия, имя, отчество)

Место прохождения практики: УВЦ, пр. Энгельса, 23

(наименование и адрес организации)

Период прохождения практики

с «28» октября 2024 г. по «09» ноября 2024 г.

Руководитель практики Хисамутдинова А.С.

(подпись) (расшифровка подписи)

Итоговая оценка по практике \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Санкт-Петербург

2024

**ЗАДАНИЕ**

**на учебную практику (по профилю специальности)**

по профессиональному модулю ПМ.02 «Осуществление интеграции программных модулей»

(код и наименование)

Специальность09.02.07 Информационные системы и программирование

(код и наименование специальности)

Студент(ка) 4 курса 42919/7 группы

Кащаев Никита Витальевич

(Фамилия, имя, отчество)

Место прохождения практики: УВЦ, пр. Энгельса, 23

(наименование и адрес организации)

Период прохождения практики

с «28» октября 2024 г. по «09» ноября 2024 г.

**Виды работ, обязательные для выполнения** *(переносится из программы, соответствующего ПМ):*

* Участие в выработке требований к программному обеспечению;
* Стадии проектирования программного обеспечения. Проектирование UML-диаграмм;
* Конструирование пользовательского интерфейса. Разработка модулей программного обеспечения;
* Разработка рабочего проекта и технологической документации.

**Индивидуальное задание: ВАРИАНТ 5**

Задание выдал «28» октября 2024 г. \_\_\_\_\_\_\_\_\_\_\_\_\_ Хисамутдинова А.С.

(подпись) (Ф.И.О.)

Задание получил «28» октября 2024 г. \_\_\_\_\_\_\_\_\_\_\_\_ Кащаев Н.В.

**МИНОБРНАУКИ РОССИИ**

федеральное государственное автономное образовательное учреждение

высшего образования

«Санкт-Петербургский политехнический университет Петра Великого»

(ФГАОУ ВО «СПбПУ»)

**Институт среднего профессионального образования**

**ДНЕВНИК**

**прохождения учебной практики УП.02.01**

**(по профилю специальности)**

по профессиональному модулю ПМ.02 «Осуществление интеграции программных модулей»

(код и наименование)

Специальность09.02.07 Информационные системы и программирование

(код и наименование специальности)

Студент(ка) 4 курса 42919/7 группы

Кащаев Никита Витальевич

(Фамилия, имя, отчество)

Место прохождения практики: УВЦ, пр. Энгельса, 23

(наименование и адрес организации)

Период прохождения практики

с «28» октября 2024 г. по «09» ноября 2024 г.

Руководитель практики Хисамутдинова А.С.

(подпись) (расшифровка подписи)

Итоговая оценка по практике

Санкт-Петербург

2024

**Содержание дневника**

|  |  |  |
| --- | --- | --- |
| **Дата** | **Виды выполненных работ и заданий по программе практики** | **Подпись руководителя практики** |
| **1** | **2** | **3** |
| 28.10.2024 | Анализ предметной области. Анализ методов решения. Анализ и выбор средств. |  |
| 29.10.2024 | Анализ и выбор средств. Исследовательские работы. Разработка технического задания. |  |
| 30.10.2024 | UML. Проектирование диаграммы вариантов использования и последовательности. UML. Проектирование диаграммы вариантов активности и состояния. |  |
| 31.10.2024 | Моделирование структуры ПО. Проектирование инфологической и даталогической модели данных. |  |
| 01.11.2024 | Проектирование интерфейса пользователя. |  |
| 02.11.2024 | Разработка схемы алгоритма программного продукта. |  |
| 05.11.2024 | Разработка модели данных. Разработка программного продукта. Создание функциональности ПО. |  |
| 06.11.2024 | Модульное тестирование. Создание тестовых случаев. |  |
| 07.11.2024 | Создание тестовых случаев. Отладка программных модулей. |  |
| 08.11.2024 | Разработка документа «Руководство пользователя» в соответствии с ЕСПД. Разработка документа «Программа и методика испытаний» в соответствии с ЕСПД. |  |
| 09.11.2024 | Подготовка отчета. |  |

СОДЕРЖАНИЕ

[ВВЕДЕНИЕ 6](#_Toc181970986)

[Задание 1. Разработка технического задания 7](#_Toc181970987)

[Задание 2. Разработка алгоритмов и диаграмм 10](#_Toc181970988)

[Задание 3. Разработка макетов программы 18](#_Toc181970989)

[Задание 4. Разработка программы 24](#_Toc181970990)

[Задание 5. Тестирование 45](#_Toc181970991)

[Задание 6. GitHub. Оценка проекта 56](#_Toc181970992)

[ЗАКЛЮЧЕНИЕ 57](#_Toc181970993)

[СПИСОК ИСПОЛЬЗОВАНЫХ ИСТОЧНИКОВ 58](#_Toc181970994)

ВВЕДЕНИЕ

Описание предметной области является важной части и основой создания технического задания(ТЗ). А техническое задание позволяет создать информационную систему. Оно служит для создания качественного нужного продукта. Является опорой для разработки. Модели данных позволяют визуализировать представление о данных системы, ее связей между сущностями. Создание Wireframe и Mockup дает возможность воспроизвести макет продукта, дает возможность заказчику оценить внешний вид системы. Далее исходя из ТЗ и макетов с моделями данных создается информационная система.

Задание 1. Разработка технического задания

Основной целью данного модуля является автоматизация процесса учета и обработок заявок на проведение мероприятий. Модуль позволяет клиентам быстро и без чужой помощи оставить заявку на проведение своего мероприятия.

Разработка программного модуля для обработки заявок и подачи заявок на проведение торжеств — это процесс создания программного продукта, который позволяет праздничным агентствам эффективно отслеживать и учет заявок на проведение праздников.

Задача разработки такого программного модуля состоит в создании удобного и эффективного инструмента для учета, отслеживания и управления заявками на проведение мероприятий, что помогает минимизировать время на обработку, заполнение и ответ на заявки.

Программный модуль позволяет агенству повысить эффективность и точность учета заявок, сократить время обработки заявок, оптимизировать использование ресурсов и повысить удовлетворенность клиентов.

Основные функции и возможности модуля включают:

1. Заявка на проведение мероприятий: это информация, предоставленная клиентом о торжестве, которое требует провести. Клиент оставляет новую заявку. Заявка может содержать данные о виде мероприятия, количестве гостей, место проведения, дату проведения, личную информацию (ФИО клиента и номер телефона). Клиент может отредактировать свою заявку.

2. Регистрация заявки: этот процесс включает приём и регистрацию заявки сотрудником в системе учёта. Важными аспектами регистрации являются присвоение уникального идентификатора заявке, сохранение информации о заявке.

3. Обработка заявки: процесс, включающий анализ заявки, определение её приоритетности и назначение исполнителя для проведения. В процессе обработки может потребоваться дополнительная информация или уточнение деталей у клиента.

4. Исполнение заявки: перевод заявки в статус «Выполнение»

5. Отчётность и информирование: важной составляющей учёта заявок на проведение мероприятий является фиксация и отчёт о выполненной работе. После завершения торжества исполнитель должен предоставить отчет о празднике, затраченных ресурсах, времени проведения.

Техническое задание

1. Общие сведения

1.1. Наименование проекта: Разработка программного модуля для учета заявок на страхование вкладов.

1.2. Заказчик: Орлов Александр Александрович.

1.3. Исполнитель: Кащаев Никита Витальевич.

2. Функциональные требования

2.1. Возможность добавления заявок в базу данных с указанием следующих параметров:

- Номер заявки;

- Дата добавления;

- Банковские реквизиты;

- ФИО клиента;

- Номер телефона;

2.2. Возможность отслеживания статуса заявки:

- Отображение списка заявок;

2.3. Расчет статистики работы отдела обслуживания:

- Количество выполненных заявок;

- Среднее время выполнения заявки;

3. Нефункциональные требования

3.1. Кроссплатформенность:

- Поддержка работы на ОС семейства Windows.

3.2. Безопасность:

- Логин и пароль для доступа к приложению;

- Доступ к данным должен быть ограничен в зависимости от роли пользователя.

3.3. Удобство использования:

- Простой и интуитивный интерфейс;

3.4. Производительность:

- Приложение должно иметь быстрый доступ к данным;

- Минимальное время отклика на запросы пользователя.

4. Требования к реализации

4.1. Язык программирования: c#

4.2. СУБД: MS SQL

5. Требования к документации

5.1. Техническое задание на разработку программного модуля.

6. Руководство по стилю

6.1. Шрифт: Times New Roman 14 .

6.2. Цветовая схема: фон – белый(#FFFFFF), Основной текст – черный (#000000), заголовки – темно-серый (#333333), кнопки – бордовый (#9C2A2FA) с белым текстом(#FFFFFF), акценты – серо-голубой (#B0BEC5) для информационных панелей

Задание 2. Разработка алгоритмов и диаграмм

Чтобы лучше понять структуру будущего приложения, требуется создать UML-диаграммы вариантов использования (рисунок 1), последовательности(рисунок 2) и активностей(рисунок 3). Это позволит лучше понять как и кто будет пользоваться программой.



Рисунок 1 – Диаграмма вариантов использования



Рисунок 2 – Диаграмма последовательностей



Рисунок 3 – Диаграмма активностей

Для лучшего создания и понимаю как правильно реализовать Базу данных(БД), понадобится создать ER-диаграмму в виде датологической модели(рисунок 4).



Рисунок 4 – Датологическая модель

Чтобы раскрыть все поля базы данных требуется словарь данных – инфологическая модель. Он представлен на таблицах 1-5.

Таблица 1 – InsuranceClaim

|  |  |
| --- | --- |
| Id\_claim PK | Int |
| Id\_client FK | Int |
| Claim\_date | Date |
| Claim\_amount | Money |
| Status | Nvarchar(10) |

Таблица 2 – Client

|  |  |
| --- | --- |
| Id\_client PK | Int |
| Login | Nvarchar(50) |
| Password | Nvarchar(256) |
| FIO | Nvarchar(50) |
| id\_bid FK | Int |
| Phine\_number | Nvarchar(20) |
| Bank\_details | Int |
| Id\_bank FK | int |
| INN | int |

Таблица 3 – Banks

|  |  |
| --- | --- |
| Id\_bank PK | Int |
| Name | Nvarchar(50) |
| INN | Int |
| Status | Nvarchar(25) |

Таблица 4 – InsurancePolicies

|  |  |
| --- | --- |
| Id\_policy PK | Int |
| Id\_bank FK | Int |
| CovarageLimit | Int |
| PremiumRate | float |

Таблица 5 – Bid

|  |  |
| --- | --- |
| Id\_bid PK | Int |
| Date\_added | Datetime |
| Bank\_details | Int |
| Phone\_number | Nvarchar(20) |
| Status | Nvarchar(10) |
| Amount | Money |
| StartDate | Date |
| EndDate | Date |

Скрипты создания таблица:

create table Bid(--заявки

id\_bid int primary key IDENTITY,

date\_added datetime not null,

FIO nvarchar(50) not null,

bank\_details nvarchar(16) not null unique,

Phone\_number nvarchar(20) not null unique CHECK(Phone\_number like '+7 ([0-9][0-9][0-9])[0-9][0-9][0-9]-[0-9][0-9]-[0-9][0-9]'),

Status nvarchar(20) not null default('ожидает'),

Amount money not null,

StartDate date,

EndDate date

);

Insert Bid values

('25.10.2024 10:00','Василиев Василий Васильевич','1111222233334444','+7 (999)111-11-11','ожидает',100000,'25.10.2024','31.12.2025'),

('26.10.2024 9:00','Орлова Александра Александровна','5555666677778888','+7 (888)222-22-22','В обработке',10000,'28.10.24','29.12.2024'),

('12.10.2024 11:00','Михайлов Махаил Михайлович','999900001212323','+7 (777)333-33-33','Принята',10000000,'20.10.24','29.12.2026'),

('14.10.2024 12:03','Петров Петр Петрович','3434454556566767','+7 (666)444-44-44','Закрыта',1000000,'15.10.24','27.12.2025'),

('30.10.2024 12:30','Щевцова Анастасия Алексеевна','78788989909011','+7 (555)555-55-55','В обработке',30000,'28.10.24','15.9.2025')

;

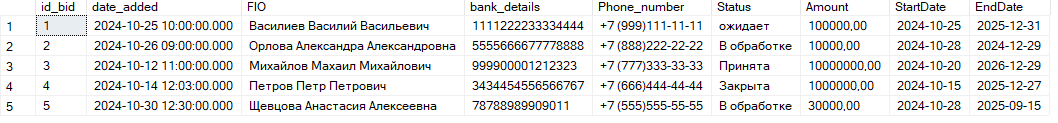


Рисунок 5 – Таблица заявок в БД

create table Banks(--банки

id\_bank int primary key IDENTITY,

Name nvarchar(50) not null,

INN nvarchar(12) not null,

Status nvarchar(20) default ('Действующий'),

);

insert Banks values ('Рысь','111122223333','Действующий'),

('Соболь','111122223333','Действующий'),

('Газ','444455556666','Действующий'),

('Фунек','777788889999','Действующий'),

('Ферзь','000012122323','Ликвидируемый');

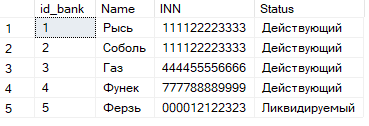


Рисунок 6 – Таблица банков в БД

CREATE TABLE Client(--клиенты

id\_client INT PRIMARY KEY IDENTITY,

Login nvarchar(50) not null unique,

Password nvarchar(256) not null unique,

FIO nvarchar(50) not null,

Role nvarchar(10)not null,

id\_bid int FOREIGN KEY REFERENCES Bid(id\_bid),

Phone\_number nvarchar(20) NOT NULL CHECK(Phone\_number like '+7 ([0-9][0-9][0-9])[0-9][0-9][0-9]-[0-9][0-9]-[0-9][0-9]') unique,

bank\_details nvarchar(16) not null unique,

id\_bank int FOREIGN KEY REFERENCES Banks(id\_bank),

INN nvarchar(12) not null

);

insert Client values ('Vas','a320480f534776bddb5cdb54b1e93d210a3c7d199e80a23c1b2178497b184c76','Петров Петр Петрович','Admin',3,'+7 (555)555-55-56','78788989909013',2,'000012122325'),--123321

('Pet','f6e0a1e2ac41945a9aa7ff8a8aaa0cebc12a3bcc981a929ad5cf810a090e11ae','Василиев Василий Васильевич','user',1,'+7 (666)445-44-44','3434454556563767',3,'777788889199'),--111

('CClar','9b871512327c09ce91dd649b3f96a63b7408ef267c8cc5710114e629730cb61f','Михайлов Махаил Михайлович','user',5,'+7 (777)310-33-33','999900001211123',4,'444455556611'),--222

('Kit','556d7dc3a115356350f1f9910b1af1ab0e312d4b3e4fc788d2da63668f36d017','Щевцова Анастасия Алексеевна','менеджер',4,'+7 (888)222-11-22','555566667777823',1,'991122223333'),--333

('swant','79f06f8fde333461739f220090a23cb2a79f6d714bee100d0e4b4af249294619','Орлова Александра Александровна','user',2,'+7 (999)111-13-21','1112222333344440',5,'111122223345');--444

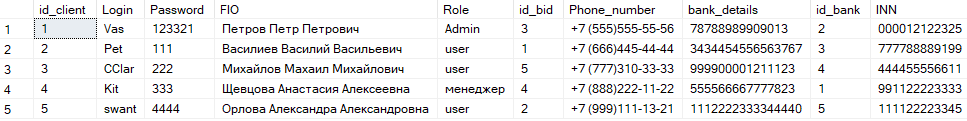


Рисунок 7 – Таблица пользователей в БД

create table InsuranceClaims(--страховые выплаты

id\_claim int primary key identity,

id\_client int foreign key references Client(id\_client),

Claim\_date date not null,

Claim\_amount money not null,

Status nvarchar(10) not null default('Ожидает')

);

insert InsuranceClaims values (3,'3.11.2024',500000,'Ожидает'),

(1,'13.11.2024',500000,'Ожидает'),

(2,'3.12.2025',5000000,'Ожидает'),

(5,'23.11.2024',5000,'Ожидает'),

(4,'30.11.2024',15000,'Ожидает');

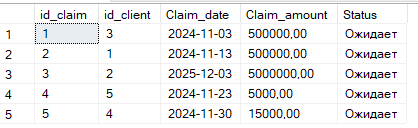


Рисунок 8 – Таблица страховые выплаты

create table InsurancePolicies(--страховые полисы

id\_policy int primary key IDENTITY,

id\_bank int foreign key references Banks(id\_bank),

CovarageLimit int not null,

PremiumRate float not null

);

insert InsurancePolicies values (1,50,3.0),

(2,30,15.0),

(3,55,10.0),

(4,40,5.0),

(5,47,27.0);

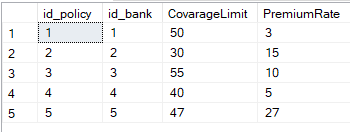


Рисунок 9 – Таблица страховые полисы

Задание 3. Разработка макетов программы

Для проектирования пользовательского интерфейса будем использовать Wireframe – черно-белый подробный план страницы сайта. Здесь намечается расположение элементов: кнопок, изображений, текстов. В Mockup уже появляются цвета, подбираются изображения, продумывается типографика. Получается красивая картинка приложения. Wireframe представлены на рисунках 10-13



Рисунок 10 – Wireframe главной страницы



Рисунок 11 – Wireframe формы входа



Рисунок 12 – Wireframe формы регистрации



Рисунок 13 – Wireframe формы заявки

На рисунках 14-17 представлены Mockup приложения.



Рисунок 14 – Mockup главной страницы



Рисунок 15 – Mockup формы входа



Рисунок 16 – Mockup формы регистрации



Рисунок 17 – Mockup формы заявки

Также важной часть является карта навигации, которая определит основные маршруты по приложению (Рисунок 18).



Рисунок 18 – Карта навигации

Задание 4. Разработка программы

Разработка приложения начинается с создания формы, добавления на неё элементов, таких как: textbox, panel, picturebox и т.д. Далее в коде формы создаются события, в которых позже прописывается то что будет происходит после указанного события. Код ниже будет описывать реализацию событий на главной форме откуда все начинается.

namespace deposit

{

public partial class Form1 : Form

{

public Form1()

{

InitializeComponent();

}

private Register reg = new Register();

private Log\_in log = new Log\_in();

private void button1\_Click(object sender, EventArgs e)

{

this.Hide();

reg.Show();

reg.FormClosed += (closedSender, closedE) =>

{

this.Show();

};

}

private void button2\_Click(object sender, EventArgs e)

{

this.Hide();

log.Show();

log.FormClosed += (closedSender, closedE) =>

{

this.Show();

};

}

}

}

Внешний вид главной формы представлен на рисунке 19.

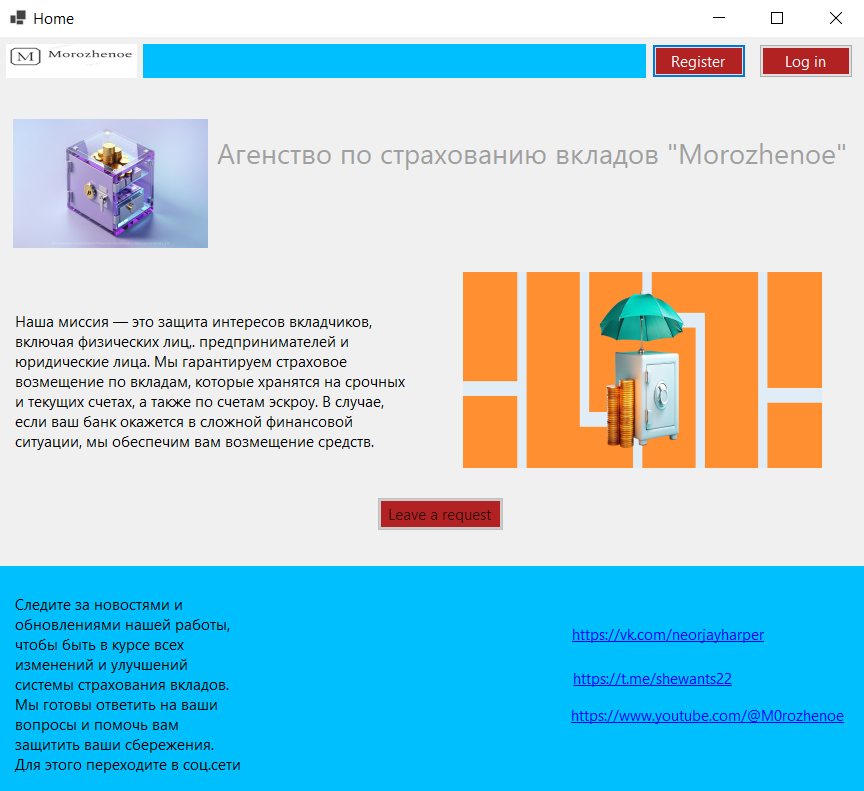


Рисунок 19 – Вид главной формы

Далее представлен код формы регистрации, где происходят похожие события, также происходит запись в базу данных с помощью запросов.

using System.Data;

using System.Data.SqlClient;

using System.Security.Cryptography;

using System.Text;

namespace deposit

{

public partial class Register : Form

{

private string con = "Server=DESKTOP-FMOKVC8\\SQLEXPRESS;Database=deposit;Trusted\_Connection=true";

private bool changed = false;

public Register()

{

InitializeComponent();

textBox2.UseSystemPasswordChar = true;

textBox3.UseSystemPasswordChar = true;

}

public static string HashPassword(string toEncrypt)

{

using (SHA256 sha256Hash = SHA256.Create())

{

byte[] bytes = sha256Hash.ComputeHash(Encoding.UTF8.GetBytes(toEncrypt));

StringBuilder builder = new StringBuilder();

for (int i = 0; i < bytes.Length; i++)

{

builder.Append(bytes[i].ToString("x2"));

}

return builder.ToString();

}

}

private main main = new main();

public void button2\_Click(object sender, EventArgs e)

{

this.Close();

}

public void button1\_Click(object sender, EventArgs e)

{

string log = textBox1.Text;

string pas = textBox2.Text;

string pas1 = textBox3.Text;

string FIO = textBox4.Text;

string phone = textBox5.Text;

string inn = textBox6.Text;

string bank\_details = textBox7.Text;

int bank = int.Parse(comboBox1.SelectedValue.ToString());

if (textBox6.Text.Length > 12 || textBox6.Text.Length < 12)

{

MessageBox.Show("Поле 'ИНН' должно иметь 12 символов", "сообщение", MessageBoxButtons.OK);

}

else if (textBox7.Text.Length > 12 || textBox7.Text.Length < 12)

{

MessageBox.Show("Поле 'Банковские реквизиты' должно иметь 12 символов", "сообщение", MessageBoxButtons.OK);

}

else if (pas != pas1)

{

MessageBox.Show("Пароли не совпадают!!!", "сообщение", MessageBoxButtons.OK);

}

else

{

pas = HashPassword(pas);

using (SqlConnection connection = new SqlConnection(con))

{

if (textBox1.Text != "" && textBox2.Text != "" && textBox3.Text != "" && textBox4.Text != "" && textBox5.Text != "" && textBox6.Text != "" && textBox7.Text != "" && comboBox1.Text != "")

{

connection.Open();

SqlDataAdapter adapter = new SqlDataAdapter();

DataSet dataSet = new DataSet();

SqlCommand command1 = new SqlCommand("SELECT \* FROM Client WHERE Login=@log or Password=@pas or Phone\_number=@phone or bank\_details=@bank\_details or INN=@inn ");

command1.Parameters.Add("@log", SqlDbType.NVarChar).Value = log;

command1.Parameters.Add("@pas", SqlDbType.NVarChar).Value = pas;

command1.Parameters.Add("@phone", SqlDbType.NVarChar).Value = phone;

command1.Parameters.Add("@inn", SqlDbType.NVarChar).Value = inn;

command1.Parameters.Add("@bank\_details", SqlDbType.NVarChar).Value = bank\_details;

command1.Parameters.Add("@bank", SqlDbType.Int).Value = bank;

adapter.SelectCommand = command1;

adapter.SelectCommand.Connection = connection;

adapter.Fill(dataSet);

if (dataSet.Tables[0].Rows.Count > 0)

{

MessageBox.Show("Такой пользователь уже существует!!! Проверьте поля: Логин, Пароль, Номер телефона, Паспортные данные или Номер водительского удостоверения.", "сообщение", MessageBoxButtons.OK);

}

else

{

SqlCommandBuilder builder = new SqlCommandBuilder(adapter);

SqlCommand command = new SqlCommand("INSERT Client VALUES(@log,@pas,@FIO,@role,@bid,@phone,@bank\_details,@bank,@inn)");

command.Parameters.Add("@log", SqlDbType.NVarChar).Value = log;

command.Parameters.Add("@pas", SqlDbType.NVarChar).Value = pas;

command.Parameters.Add("@FIO", SqlDbType.NVarChar).Value = FIO;

command.Parameters.Add("@role", SqlDbType.NVarChar).Value = "user";

command.Parameters.Add("@bid", SqlDbType.Int).Value = 1;

command.Parameters.Add("@phone", SqlDbType.NVarChar).Value = phone;

command.Parameters.Add("@bank\_details", SqlDbType.NVarChar).Value = bank\_details;

command.Parameters.Add("@bank", SqlDbType.Int).Value = bank;

command.Parameters.Add("@inn", SqlDbType.NVarChar).Value = inn;

adapter.SelectCommand = command;

adapter.SelectCommand.Connection = connection;

command.ExecuteNonQuery();

MessageBox.Show("Вы успешно зарегестрировались!");

this.Hide();

main.Show();

main.FormClosed += (closedSender, closedE) =>

{

this.Show();

};

}

}

else

{

MessageBox.Show("Заполните все поля", "сообщение", MessageBoxButtons.OK);

}

}

}

}

public void comboBox1\_KeyPress(object sender, KeyPressEventArgs e)

{

e.Handled = true;

}

public void button3\_Click(object sender, EventArgs e)

{

if (changed)

{

button3.BackgroundImage = Properties.Resources.\_2;

button3.BackgroundImageLayout = ImageLayout.Stretch;

textBox2.UseSystemPasswordChar = true;

}

else

{

button3.BackgroundImage = Properties.Resources.\_1;

button3.BackgroundImageLayout = ImageLayout.Stretch;

textBox2.UseSystemPasswordChar = false;

}

changed = !changed;

}

public void button4\_Click(object sender, EventArgs e)

{

if (changed)

{

button4.BackgroundImage = Properties.Resources.\_2;

button4.BackgroundImageLayout = ImageLayout.Stretch;

textBox3.UseSystemPasswordChar = true;

}

else

{

button4.BackgroundImage = Properties.Resources.\_1;

button4.BackgroundImageLayout = ImageLayout.Stretch;

textBox3.UseSystemPasswordChar = false;

}

changed = !changed;

}

public void Register\_Load(object sender, EventArgs e)

{

using (SqlConnection connection = new SqlConnection(con))

{

connection.Open();

SqlDataAdapter adapter = new SqlDataAdapter();

DataTable data = new DataTable();

data.Columns.Add("Name");

data.Columns.Add("id\_avto");

SqlCommand command = new SqlCommand("Select Name,id\_bank from Banks");

adapter.SelectCommand = command;

adapter.SelectCommand.Connection = connection;

adapter.Fill(data);

comboBox1.DataSource = data;

comboBox1.DisplayMember = "Name";

comboBox1.ValueMember = "id\_bank";

}

}

public void textBox6\_KeyPress(object sender, KeyPressEventArgs e)

{

char number = e.KeyChar;

e.Handled = false;

if (!Char.IsDigit(number) && number != 8)

{

e.Handled = true;

}

}

public void textBox7\_KeyPress(object sender, KeyPressEventArgs e)

{

char number = e.KeyChar;

e.Handled = false;

if (!Char.IsDigit(number) && number != 8)

{

e.Handled = true;

}

}

public void textBox5\_KeyPress(object sender, KeyPressEventArgs e)

{

char number = e.KeyChar;

e.Handled = false;

if (!Char.IsDigit(number) && number != 8 && e.KeyChar != '+' && e.KeyChar != '(' && e.KeyChar != ')' && e.KeyChar != '-' && e.KeyChar != ' ')

{

e.Handled = true;

}

}

}

}

Внешний вид формы регистрации представлен на рисунке 20.

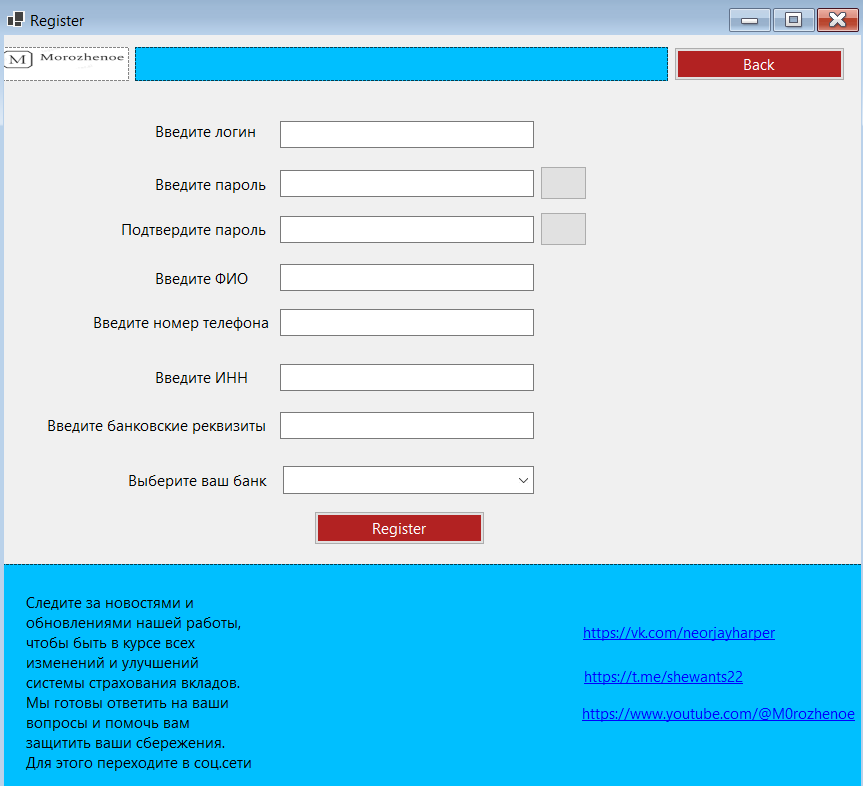


Рисунок 20 – Вид формы регистрации

Код, представленный ниже, описывает структуру формы входа в свой аккаунт.

using System.Data;

using System.Data.SqlClient;

using System.Security.Cryptography;

using System.Text;

namespace deposit

{

public partial class Log\_in : Form

{

public Log\_in()

{

InitializeComponent();

textBox2.UseSystemPasswordChar = true;

}

private string con = "Server=DESKTOP-FMOKVC8\\SQLEXPRESS;Database=deposit;Trusted\_Connection=true";

private bool changed = false;

private int time = 0;

private int sec = 900;

private int a = 0;

public static string HashPassword(string toEncrypt)

{

using (SHA256 sha256Hash = SHA256.Create())

{

byte[] bytes = sha256Hash.ComputeHash(Encoding.UTF8.GetBytes(toEncrypt));

StringBuilder builder = new StringBuilder();

for (int i = 0; i < bytes.Length; i++)

{

builder.Append(bytes[i].ToString("x2"));

}

return builder.ToString();

}

}

private void button3\_Click\_1(object sender, EventArgs e)

{

if (changed)

{

button3.BackgroundImage = Properties.Resources.\_2;

button3.BackgroundImageLayout = ImageLayout.Stretch;

textBox2.UseSystemPasswordChar = true;

}

else

{

button3.BackgroundImage = Properties.Resources.\_1;

button3.BackgroundImageLayout = ImageLayout.Stretch;

textBox2.UseSystemPasswordChar = false;

}

changed = !changed;

}

private main\_m m = new main\_m();

private Admin admin = new Admin();

private main main = new main();

private void button1\_Click(object sender, EventArgs e)

{

string log = textBox1.Text;

string pas = textBox2.Text;

pas = HashPassword(pas);

using (SqlConnection connection = new SqlConnection(con))

{

connection.Open();

DataSet dataSet = new DataSet();

SqlCommand command = new SqlCommand("SELECT \* FROM Client WHERE Login=@uL AND Password=@uP");

command.Parameters.Add("@uL", SqlDbType.NVarChar).Value = log;

command.Parameters.Add("@uP", SqlDbType.NVarChar).Value = pas;

SqlDataAdapter adapter = new SqlDataAdapter();

adapter.SelectCommand = command;

adapter.SelectCommand.Connection = connection;

DataTable table = new DataTable();

table.Columns.Add("Login");

table.Columns.Add("Password");

adapter.Fill(dataSet);

SqlDataReader reader = command.ExecuteReader();

reader.Read();

if (dataSet.Tables[0].Rows.Count > 0 && reader.GetValue(4).ToString() == "менеджер")

{

this.Hide();

m.Show();

m.FormClosed += (closedSender, closedE) =>

{

this.Show();

};

}

else if (dataSet.Tables[0].Rows.Count > 0 && reader.GetValue(4).ToString() == "Admin")

{

this.Hide();

admin.Show();

admin.FormClosed += (closedSender, closedE) =>

{

this.Show();

};

}

else if (dataSet.Tables[0].Rows.Count > 0 && reader.GetValue(4).ToString() == "user")

{

this.Hide();

main.Show();

main.FormClosed += (closedSender, closedE) =>

{

this.Show();

};

}

else if (a < 3)

{

a++;

MessageBox.Show("Неверный логин или пароль!");

}

else

{

timer1.Start();

MessageBox.Show("вы превысили лимит ввода неверных данных.");

button1.Enabled = false;

textBox1.Enabled = false;

textBox2.Enabled = false;

}

}

}

private void timer1\_Tick(object sender, EventArgs e)

{

time++;

if (time == sec)

{

timer1.Stop();

button1.Enabled = true;

textBox1.Enabled = true;

textBox2.Enabled = true;

}

}

private void button2\_Click(object sender, EventArgs e)

{

this.Close();

}

}

}

На рисунке 21 изображен вид формы регистрации.

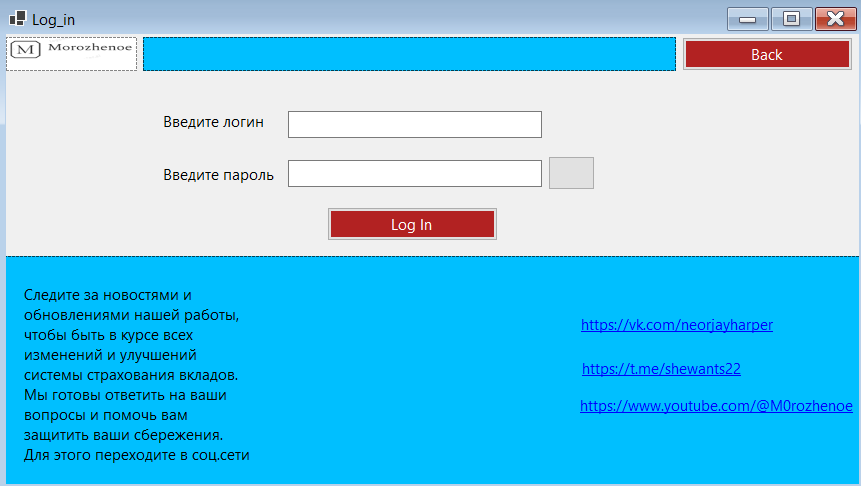


Рисунок 21 – Вид формы вход

Код, представленный ниже, описывает структуру формы пользователя после входа или регистрации. На форме пользовать может перейти и оставить свою заявку на страхование.

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace deposit

{

public partial class main : Form

{

public main()

{

InitializeComponent();

}

private Request request = new Request();

private void button3\_Click(object sender, EventArgs e)

{

this.Hide();

request.Show();

request.FormClosed += (closedSender, closedE) =>

{

this.Show();

};

}

private void button2\_Click(object sender, EventArgs e)

{

this.Close();

}

}

}

Внешний вид формы пользователя представлен на рисунке 22.

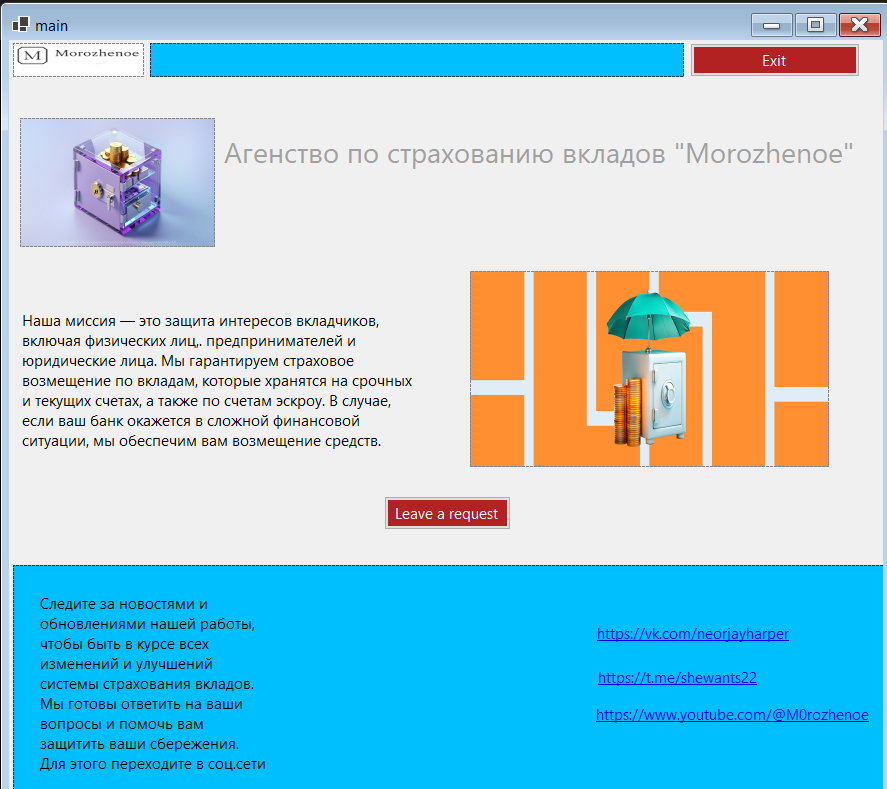


Рисунок 22 – Вид формы пользователя

Код ниже показывает описание событий формы подачи заявки клиентом.

using System.Data;

using System.Data.SqlClient;

namespace deposit

{

public partial class Request : Form

{

public Request()

{

InitializeComponent();

}

private string con = "Server=DESKTOP-FMOKVC8\\SQLEXPRESS;Database=deposit;Trusted\_Connection=true";

private void button2\_Click(object sender, EventArgs e)

{

this.Close();

}

private void button1\_Click(object sender, EventArgs e)

{

string FIO = textBox1.Text;

DateTime date = DateTime.Now;

string phone = textBox2.Text;

string bank\_details = textBox3.Text;

float amount = float.Parse(textBox4.Text);

if (textBox3.Text.Length > 12 || textBox3.Text.Length < 12)

{

MessageBox.Show("Поле 'ИНН' должно иметь 12 символов", "сообщение", MessageBoxButtons.OK);

}

else

{

using (SqlConnection connection = new SqlConnection(con))

{

connection.Open();

SqlDataAdapter adapter = new SqlDataAdapter();

DataSet dataSet = new DataSet();

SqlCommand command1 = new SqlCommand("SELECT \* FROM Bid WHERE Phone\_number=@phone or bank\_details=@bank");

command1.Parameters.Add("@phone", SqlDbType.NVarChar).Value = phone;

command1.Parameters.Add("@bank", SqlDbType.NVarChar).Value = bank\_details;

adapter.SelectCommand = command1;

adapter.SelectCommand.Connection = connection;

adapter.Fill(dataSet);

if (dataSet.Tables[0].Rows.Count > 0)

{

MessageBox.Show("Такая заявка уже есть!!! Дождитесь ее окончания", "сообщение", MessageBoxButtons.OK);

}

else

{

SqlCommand command = new SqlCommand("INSERT Bid VALUES(@dateadd,@FIO,@bank\_details,@Phone,@status,@amount,@start,@end)");

command.Parameters.Add("@dateadd", SqlDbType.DateTime).Value = date;

command.Parameters.Add("@FIO", SqlDbType.NVarChar).Value = FIO;

command.Parameters.Add("@bank\_details", SqlDbType.NVarChar).Value = bank\_details;

command.Parameters.Add("@Phone", SqlDbType.NVarChar).Value = phone;

command.Parameters.Add("@status", SqlDbType.NVarChar).Value = "ожидает";

command.Parameters.Add("@amount", SqlDbType.NVarChar).Value = amount;

command.Parameters.Add(new SqlParameter("@start", DBNull.Value));

command.Parameters.Add(new SqlParameter("@end", DBNull.Value));

adapter.SelectCommand = command;

adapter.SelectCommand.Connection = connection;

command.ExecuteNonQuery();

SqlCommand command2 = new SqlCommand("Select id\_bid FROM Bid Where Phone\_number=@phone");

command2.Parameters.Add("@phone", SqlDbType.NVarChar).Value = phone;

adapter.SelectCommand = command2;

adapter.SelectCommand.Connection = connection;

SqlDataReader reader = command2.ExecuteReader();

reader.Read();

int id = int.Parse(reader.GetValue(0).ToString());

reader.Close();

SqlCommand command3 = new SqlCommand("Update Client SET id\_bid=@id Where Phone\_number=@phone");

command3.Parameters.Add("@id", SqlDbType.Int).Value = id;

command3.Parameters.Add("@phone", SqlDbType.NVarChar).Value = phone;

adapter.SelectCommand = command3;

adapter.SelectCommand.Connection = connection;

command3.ExecuteNonQuery();

MessageBox.Show("Ваша заявка принята!");

this.Close();

}

}

}

}

private void textBox2\_KeyPress(object sender, KeyPressEventArgs e)

{

char number = e.KeyChar;

e.Handled = false;

if (!Char.IsDigit(number) && number != 8 && e.KeyChar != '+' && e.KeyChar != '(' && e.KeyChar != ')' && e.KeyChar != '-' && e.KeyChar != ' ')

{

e.Handled = true;

}

}

private void textBox3\_KeyPress(object sender, KeyPressEventArgs e)

{

char number = e.KeyChar;

e.Handled = false;

if (!Char.IsDigit(number) && number != 8)

{

e.Handled = true;

}

}

private void textBox4\_KeyPress(object sender, KeyPressEventArgs e)

{

char number = e.KeyChar;

e.Handled = false;

if (!Char.IsDigit(number) && number != 8)

{

e.Handled = true;

}

}

}

}

Внешний вид формы подачи заявок у клиента изображен на рисунке 23.

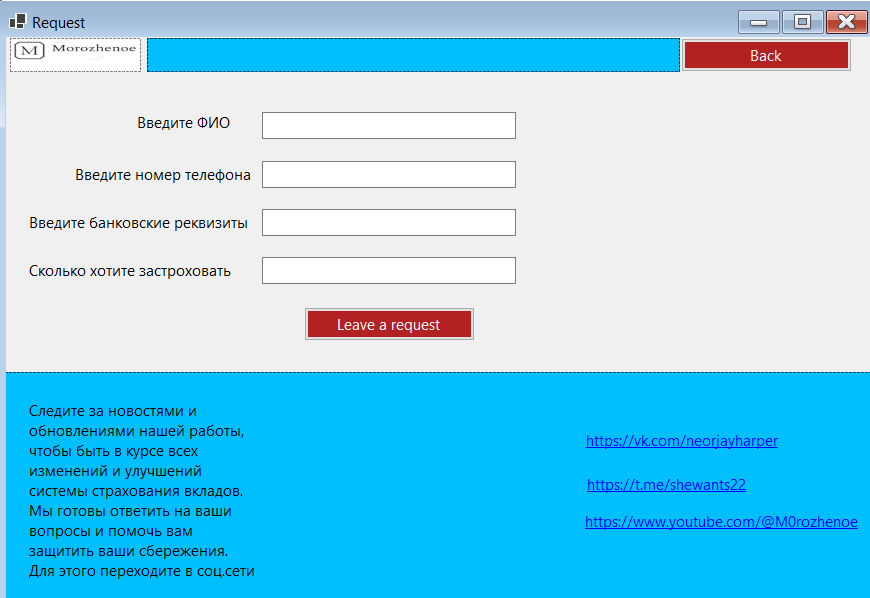


Рисунок 23 – Вид формы заявки

Код ниже показывает то, что происходит на форме у менеджера. Чем он может управлять.

using System.Data;

using System.Data.SqlClient;

using static System.Runtime.InteropServices.JavaScript.JSType;

namespace deposit

{

public partial class main\_m : Form

{

public main\_m()

{

InitializeComponent();

}

private string con = "Server=DESKTOP-FMOKVC8\\SQLEXPRESS;Database=deposit;Trusted\_Connection=true";

private DataSet DataSet = new DataSet();

private void main\_m\_Load(object sender, EventArgs e)

{

using (SqlConnection connection = new SqlConnection(con))

{

connection.Open();

SqlDataAdapter adapter = new SqlDataAdapter();

DataSet = new DataSet();

SqlCommand command = new SqlCommand($"SELECT \* FROM Bid");

adapter.SelectCommand = command;

adapter.SelectCommand.Connection = connection;

adapter.Fill(DataSet);

dataGridView1.DataSource = DataSet.Tables[0];

CreateTextBox(DataSet.Tables[0].Columns);

}

}

private void CreateTextBox(DataColumnCollection columns)

{

int yOffset = dataGridView1.Bottom + 10; // начальное смещение по Y, ниже DataGridView

int xOffset = dataGridView1.Left; // начальное смещение по X

foreach (DataColumn column in columns)

{

// Создаем новый Label

Label label = new Label();

label.Name = "label\_" + column.ColumnName;

label.Text = column.ColumnName;

label.Location = new System.Drawing.Point(xOffset, yOffset);

label.AutoSize = true;

this.Controls.Add(label);

// Создаем новый TextBox

TextBox textBox = new TextBox();

textBox.Name = "textBox\_" + column.ColumnName;

textBox.Location = new System.Drawing.Point(xOffset + label.Width + 10, yOffset - 3);

textBox.Width = 150;

textBox.TextChanged += TextBox\_TextChanged;

textBox.KeyPress += TextBox\_KeyPress;

this.Controls.Add(textBox);

textBox.Enabled = false;

if (textBox.Name == "textBox\_Status" || textBox.Name == "textBox\_StartDate" || textBox.Name == "textBox\_EndDate")

textBox.Enabled = true;

yOffset += 30; // увеличиваем смещение для следующего TextBox

}

}

private void dataGridView1\_SelectionChanged(object sender, EventArgs e)

{

if (dataGridView1.SelectedRows.Count > 0)

{

DataGridViewRow selectedRow = dataGridView1.SelectedRows[0];

foreach (DataGridViewColumn column in dataGridView1.Columns)

{

TextBox textBox = this.Controls.Find("textBox\_" + column.Name, true).FirstOrDefault() as TextBox;

if (textBox != null)

{

textBox.Text = selectedRow.Cells[column.Name].Value?.ToString();

}

}

}

}

private void TextBox\_TextChanged(object sender, EventArgs e)

{

if (dataGridView1.SelectedRows.Count > 0)

{

DataGridViewRow selectedRow = dataGridView1.SelectedRows[0];

TextBox textBox = sender as TextBox;

string columnName = textBox.Name.Substring("textBox\_".Length);

selectedRow.Cells[columnName].Value = textBox.Text;

}

}

public void TextBox\_KeyPress(object sender,KeyPressEventArgs e)

{

e.Handled = false;

TextBox textBox = sender as TextBox;

if (textBox.Name == "textBox\_StartDate" || textBox.Name == "textBox\_EndDate")

{

char number = e.KeyChar;

if (!Char.IsDigit(number) && e.KeyChar !='.' && number!=8)

{

e.Handled = true;

}

}

}

private void button2\_Click(object sender, EventArgs e)

{

using (SqlConnection connection = new SqlConnection(con))

{

SqlDataAdapter dataAdapter = new SqlDataAdapter();

SqlCommandBuilder commandBuilder = new SqlCommandBuilder(dataAdapter);

SqlCommand command = new SqlCommand($"select \* from Bid");

dataAdapter.SelectCommand = command;

dataAdapter.SelectCommand.Connection = connection;

try

{

connection.Open();

dataAdapter.Update(DataSet);

MessageBox.Show("Изменения сохранены успешно.");

}

catch (Exception ex)

{

MessageBox.Show("Произошла ошибка при сохранении изменений: " + ex.Message);

}

}

}

private void button1\_Click(object sender, EventArgs e)

{

this.Close();

}

}

}

На рисунке 24 представлен, то как выглядит внешне форма у менеджера.

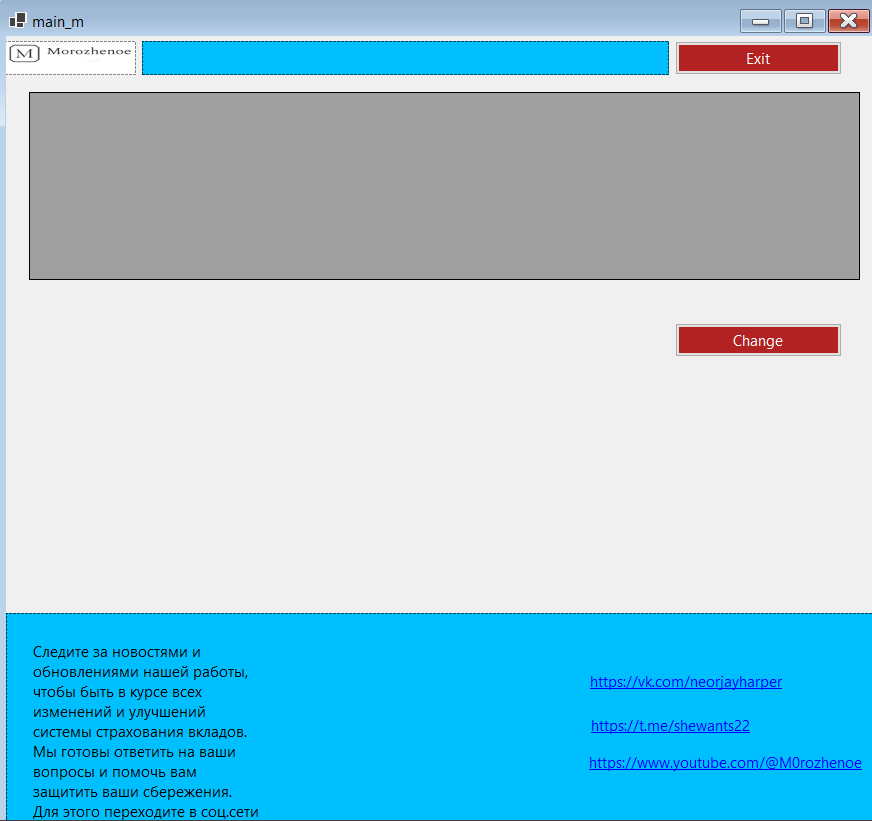


Рисунок 24 – Вид формы менеджера

Последний код описывает события элементов на последней форме – форме администратора.

using System.Data;

using System.Data.SqlClient;

using System.Security.Cryptography;

using System.Text;

namespace deposit

{

public partial class Admin : Form

{

private string con = "Server=DESKTOP-FMOKVC8\\SQLEXPRESS;Database=deposit;Trusted\_Connection=true";

private bool changed = false;

private DataSet DataSet = new DataSet();

private string table;

public Admin()

{

InitializeComponent();

}

private void button1\_Click(object sender, EventArgs e)

{

table = comboBox1.SelectedItem.ToString();

using (SqlConnection connection = new SqlConnection(con))

{

connection.Open();

SqlDataAdapter adapter = new SqlDataAdapter();

DataSet = new DataSet();

SqlCommand command = new SqlCommand($"SELECT \* FROM {table}");

adapter.SelectCommand = command;

adapter.SelectCommand.Connection = connection;

adapter.Fill(DataSet);

dataGridView1.DataSource = DataSet.Tables[0];

CreateTextBox(DataSet.Tables[0].Columns);

}

}

private void CreateTextBox(DataColumnCollection columns)

{

int yOffset = dataGridView1.Bottom + 10; // начальное смещение по Y, ниже DataGridView

int xOffset = dataGridView1.Left; // начальное смещение по X

foreach (DataColumn column in columns)

{

// Создаем новый Label

Label label = new Label();

label.Name = "label\_" + column.ColumnName;

label.Text = column.ColumnName;

label.Location = new System.Drawing.Point(xOffset, yOffset);

label.AutoSize = true;

this.Controls.Add(label);

// Создаем новый TextBox

TextBox textBox = new TextBox();

textBox.Name = "textBox\_" + column.ColumnName;

textBox.Location = new System.Drawing.Point(xOffset + label.Width + 10, yOffset - 3);

textBox.Width = 150;

textBox.TextChanged += TextBox\_TextChanged;

this.Controls.Add(textBox);

yOffset += 30; // увеличиваем смещение для следующего TextBox

}

}

private void ClearDynamicTextBoxes()

{

var textBoxes = this.Controls.OfType<TextBox>().Where(tb => tb.Name.StartsWith("textBox\_")).ToList();

foreach (var textBox in textBoxes)

{

this.Controls.Remove(textBox);

}

}

private void ClearLabel()

{

var labels = this.Controls.OfType<Label>().Where(lbl => lbl.Name.StartsWith("label\_")).ToList();

foreach (var label in labels)

{

this.Controls.Remove(label);

}

}

private void comboBox1\_KeyPress(object sender, KeyPressEventArgs e)

{

e.Handled = true;

}

private void comboBox1\_SelectedValueChanged(object sender, EventArgs e)

{

dataGridView1.Columns.Clear();

dataGridView1.DataSource = null;

dataGridView1.Refresh();

ClearDynamicTextBoxes();

ClearLabel();

}

private void dataGridView1\_SelectionChanged(object sender, EventArgs e)

{

if (dataGridView1.SelectedRows.Count > 0)

{

DataGridViewRow selectedRow = dataGridView1.SelectedRows[0];

foreach (DataGridViewColumn column in dataGridView1.Columns)

{

TextBox textBox = this.Controls.Find("textBox\_" + column.Name, true).FirstOrDefault() as TextBox;

if (textBox != null)

{

textBox.Text = selectedRow.Cells[column.Name].Value?.ToString();

}

}

}

}

private void button2\_Click(object sender, EventArgs e)

{

if (dataGridView1.SelectedRows.Count > 0)

{

foreach (DataGridViewRow row in dataGridView1.SelectedRows)

{

dataGridView1.Rows.RemoveAt(row.Index);

}

}

else

{

MessageBox.Show("Пожалуйста, выберите строку для удаления.");

}

}

private void button3\_Click(object sender, EventArgs e)

{

DataTable currentDataTable = DataSet.Tables[0];

DataRow newRow = currentDataTable.NewRow();

foreach (DataGridViewColumn column in dataGridView1.Columns)

{

TextBox textBox = this.Controls.Find("textBox\_" + column.Name, true).FirstOrDefault() as TextBox;

if (textBox != null)

{

if (column.Name.Equals("Password", StringComparison.OrdinalIgnoreCase))

newRow[column.Name] = HashPassword(textBox.Text);

else

newRow[column.Name] = textBox.Text;

}

}

currentDataTable.Rows.Add(newRow);

}

private void TextBox\_TextChanged(object sender, EventArgs e)

{

if (dataGridView1.SelectedRows.Count > 0)

{

DataGridViewRow selectedRow = dataGridView1.SelectedRows[0];

TextBox textBox = sender as TextBox;

string columnName = textBox.Name.Substring("textBox\_".Length);

selectedRow.Cells[columnName].Value = textBox.Text;

}

}

private void button4\_Click(object sender, EventArgs e)

{

using (SqlConnection connection = new SqlConnection(con))

{

SqlDataAdapter dataAdapter = new SqlDataAdapter();

SqlCommandBuilder commandBuilder = new SqlCommandBuilder(dataAdapter);

SqlCommand command = new SqlCommand($"select \* from {table}");

dataAdapter.SelectCommand = command;

dataAdapter.SelectCommand.Connection = connection;

try

{

connection.Open();

dataAdapter.Update(DataSet);

MessageBox.Show("Изменения сохранены успешно.");

}

catch (Exception ex)

{

MessageBox.Show("Произошла ошибка при сохранении изменений: " + ex.Message);

}

}

}

public static string HashPassword(string toEncrypt)

{

using (SHA256 sha256Hash = SHA256.Create())

{

byte[] bytes = sha256Hash.ComputeHash(Encoding.UTF8.GetBytes(toEncrypt));

StringBuilder builder = new StringBuilder();

for (int i = 0; i < bytes.Length; i++)

{

builder.Append(bytes[i].ToString("x2"));

}

return builder.ToString();

}

}

private void button5\_Click(object sender, EventArgs e)

{

comboBox2.Visible = true;

}

private void comboBox2\_SelectedValueChanged(object sender, EventArgs e)

{

if (comboBox2.SelectedItem == "Количество выполненых заявок")

{

dataGridView1.Columns.Clear();

dataGridView1.DataSource = null;

dataGridView1.Refresh();

ClearDynamicTextBoxes();

ClearLabel();

using (SqlConnection connection = new SqlConnection(con))

{

connection.Open();

SqlDataAdapter Adapter = new SqlDataAdapter();

SqlCommandBuilder commandBuilder = new SqlCommandBuilder(Adapter);

SqlCommand command = new SqlCommand($"select \* from Bid Where Status=@st");

command.Parameters.Add("@st", SqlDbType.NVarChar).Value = "Закрыта";

Adapter.SelectCommand = command;

Adapter.SelectCommand.Connection = connection;

Adapter.Fill(DataSet);

if (DataSet.Tables[0].Rows.Count > 0)

{

dataGridView1.Columns.Add("Column1", "Количество выполненых заявок");

int rowIndex = dataGridView1.Rows.Add();

dataGridView1.Rows[rowIndex].Cells["Column1"].Value = DataSet.Tables[0].Rows.Count;

}

else

{

MessageBox.Show("Еще нет выполненых заявок", "сообщение", MessageBoxButtons.OK);

}

}

}

else

{

dataGridView1.Columns.Clear();

dataGridView1.DataSource = null;

dataGridView1.Refresh();

ClearDynamicTextBoxes();

ClearLabel();

int sum = 0;

using (SqlConnection connection = new SqlConnection(con))

{

connection.Open();

SqlDataAdapter Adapter = new SqlDataAdapter();

SqlCommandBuilder commandBuilder = new SqlCommandBuilder(Adapter);

SqlCommand command = new SqlCommand($"select StartDate,EndDate from Bid");

Adapter.SelectCommand = command;

Adapter.SelectCommand.Connection = connection;

Adapter.Fill(DataSet);

SqlDataReader reader = command.ExecuteReader();

while (reader.Read())

{

DateOnly start = DateOnly.Parse(reader.GetValue(0).ToString().Substring(0, reader.GetValue(0).ToString().Length - 10));//10.11.2024 00:00:00

DateOnly end = DateOnly.Parse(reader.GetValue(1).ToString().Substring(0, reader.GetValue(0).ToString().Length - 10));

int daystart = start.Day;

int dayend = end.Day;

int timefineshed = dayend - daystart;

sum += timefineshed;

}

double sr = (double)sum / DataSet.Tables[0].Rows.Count;

dataGridView1.Columns.Add("Column1", "Среднее время выполнения заявок(дни)");

int rowIndex = dataGridView1.Rows.Add();

dataGridView1.Rows[rowIndex].Cells["Column1"].Value = sr;

}

}

}

private void comboBox2\_KeyPress(object sender, KeyPressEventArgs e)

{

e.Handled = true;

}

}

}

Внешний вид формы администратора представлен на рисунке 25.

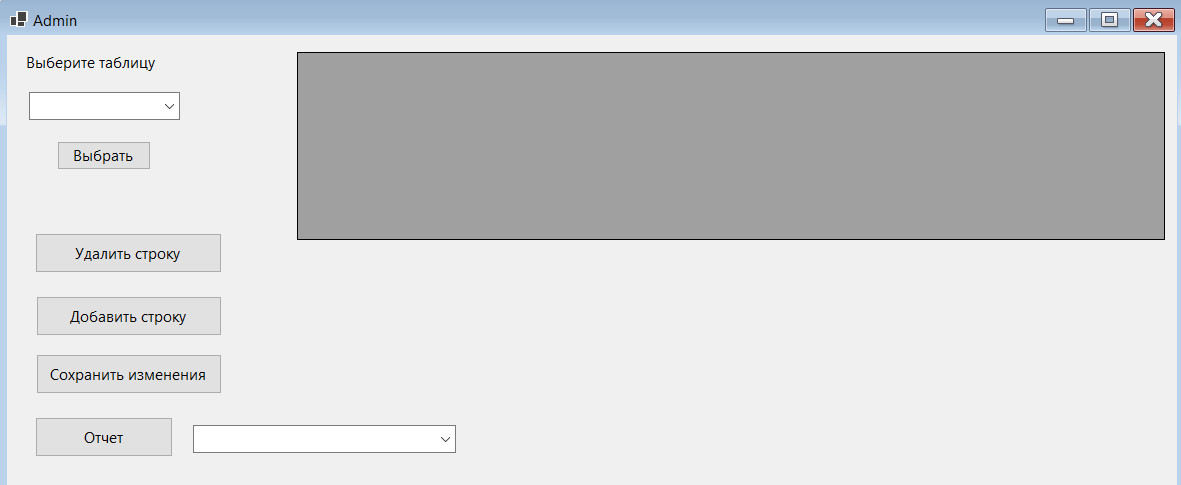


Рисунок 25 – Вид формы администратора

Задание 5. Тестирование

После успешной разработки информационной системы были проведены тест-кейсы. Ниже в таблицах 6-10 представлены результаты.

|  |  |
| --- | --- |
| **Название проекта** | Агентство по страхованию вкладов |
| **Рабочая версия** | 1.0 |
| **Имя тестирующего** | Кащаев Никита |
| **Дата(ы) теста** | 07.11.2024 |

Таблица 6 – Тест-кейс 1

|  |  |
| --- | --- |
| **Тестовый пример #** | 1 |
| **Приоритет тестирования** | Высокий |
| **Заголовок/название теста** | Подтвердить станицу регистрации |
| **Краткое изложение теста** | Осуществить успешную авторизацию |
| **Этапы теста** | 1. Заполнить поля соответствующими данными 2. Отправить форму нажатием на кнопку «Зарегистрироваться» |
| **Тестовые данные** | Имя: Александр  Фамилия: Александрович  Отчество: Александрович  Номер телефона: 8911111111  Адрес почты: [example@test.com](mailto:example@test.com)  Пароль: 123123  Подтверждение пароля: 123456 |
| **Ожидаемый результат** | Ошибки регистрации, из-за несходства паролей |
| **Фактический результат** | Ошибка регистрации |
| **Статус** | Заче |
| **Предварительное условие** | Пользователь должен находиться на странице регистрации |
| **Постусловие** | После выполнения, программа должна работать в штатном режиме |
| **Примечания/комментарии** |  |

Таблица 7 – Тест-кейс 2

|  |  |
| --- | --- |
| **Тестовый пример #** | 2 |
| **Приоритет тестирования** | Средний |
| **Заголовок/название теста** | Вход |
| **Краткое изложение теста** | Проверить функцию входа в профиль |
| **Этапы теста** | 1. На главной форме нажать кнопку «Log In» 2. На открывшейся форме заполнить поля 3. После заполнения полей нажать кнопку «Log In» |
| **Тестовые данные** | Логин: «nk»; Пароль: «6666»; |
| **Ожидаемый результат** | Переход на форму под аккаунтом |
| **Фактический результат** | Переход на форму под аккаунтом |
| **Статус** | Пройден |
| **Предварительное условие** | Программа открыта |
| **Постусловие** | - |
| **Примечания/комментарии** | - |

Таблица 8 – Тест-кейс 3

|  |  |
| --- | --- |
| **Тестовый пример #** | 3 |
| **Приоритет тестирования** | Средний |
| **Заголовок/название теста** | Подачи заявки |
| **Краткое изложение теста** | Проверить функцию подачи заявки на страхование вклада |
| **Этапы теста** | 1. На главной форме нажать кнопку «Log In» 2. На открывшейся форме заполнить поля 3. После заполнения полей нажать кнопку «Log In» 4. На форме пользователя нажать кнопку «Leave a request» 5. Заполнить все поля 6. Нажать кнопку «Leave a request» |
| **Тестовые данные** | Логин: «nk»; Пароль: «6666»; ФИО: «Кащаев Никита Витальевич»; Номер телефона: «+7 (999)123-21-12»; Банковские реквизиты: «123456789111»; Сумма: «10000» |
| **Ожидаемый результат** | Сообщение «Ваша заявка принята» и закрытие этой формы |
| **Фактический результат** | Сообщение «Ваша заявка принята» и закрытие этой формы |
| **Статус** | Пройден |
| **Предварительное условие** | Программа открыта |
| **Постусловие** | - |
| **Примечания/комментарии** | - |

Таблица 9 – Тест-кейс 4

|  |  |
| --- | --- |
| **Тестовый пример #** | 4 |
| **Приоритет тестирования** | Средний |
| **Заголовок/название теста** | Редактирование заявки |
| **Краткое изложение теста** | Редактирование заявки менеджером |
| **Этапы теста** | 1. На главной форме нажать кнопку «Log In» 2. На открывшейся форме заполнить поля 3. После заполнения полей нажать кнопку «Log In» 4. Выбрать заявку 5. Изменить поля 6. Нажать кнопку «Change» |
| **Тестовые данные** | Логин: «Kit»; Пароль: «333»; первая заявка; Статус: «Выполнение»; Дата начала: «07.11.2024»; Дата окончания: «09.11.2024» |
| **Ожидаемый результат** | Изменение в таблице, сообщение «Изменение сохранены успешно» |
| **Фактический результат** | Изменение в таблице, сообщение «Изменение сохранены успешно» |
| **Статус** | Пройден |
| **Предварительное условие** | Программа открыта |
| **Постусловие** | - |
| **Примечания/комментарии** | - |

Таблица 10 – Тест-кейс 5

|  |  |
| --- | --- |
| **Тестовый пример #** | 5 |
| **Приоритет тестирования** | Средний |
| **Заголовок/название теста** | Удаление строки из таблицы |
| **Краткое изложение теста** | Удаление одной выбранной строки из таблицы администратором |
| **Этапы теста** | 1. На главной форме нажать кнопку «Log In» 2. На открывшейся форме заполнить поля 3. После заполнения полей нажать кнопку «Log In» 4. Выбрать таблицу 5. Нажать кнопку «Выбрать» 6. Выбрать строку 7. Нажать кнопку «Удалить строку» 8. Нажать кнопку «Сохранить изменения» |
| **Тестовые данные** | Логин: «Vas»; Пароль: «123321»; Таблица: Bank; строка 1 |
| **Ожидаемый результат** | Изменение в таблице, сообщение «Изменение сохранены успешно» |
| **Фактический результат** | Изменение в таблице, сообщение «Изменение сохранены успешно» |
| **Статус** | Пройден |
| **Предварительное условие** | Программа открыта |
| **Постусловие** | - |
| **Примечания/комментарии** | - |

Для тестирования приложения, событий будут использоваться тесты Nunit. Это позволит проверить есть ли ограничения на поля ввода на некоторых формах. Тесты будут пробовать ввести данные и ожидать ответа от приложения.

Тест 1

[Test]

public void TextBox5\_KeyPress\_OnlyDigitsAndSymbolsAllowed()

{

var e = new KeyPressEventArgs('a');

register.textBox5\_KeyPress(register.textBox5, e);

ClassicAssert.IsTrue(e.Handled);

}

При успешном выполнении теста(рисунок 26) поля номера телефона на форме регистрации. Иные символы, кроме цифр, знаков «+», «(», «)», «-» будут не вноситься в поле.

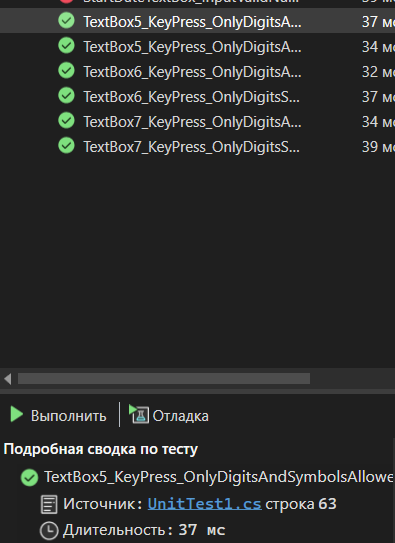


Рисунок 26 – Номер телефона

Тест 2

[Test]

public void TextBox5\_KeyPress\_OnlyDigitsAndValidSymbolsShouldBeAllowed()

{

var e1 = new KeyPressEventArgs('1');

register.textBox5\_KeyPress(register.textBox5, e1);

ClassicAssert.IsFalse(e1.Handled);

var e2 = new KeyPressEventArgs('+');

register.textBox5\_KeyPress(register.textBox5, e2);

ClassicAssert.IsFalse(e2.Handled);

var e3 = new KeyPressEventArgs('-');

register.textBox5\_KeyPress(register.textBox5, e3);

ClassicAssert.IsFalse(e3.Handled);

var e4 = new KeyPressEventArgs(' ');

register.textBox5\_KeyPress(register.textBox5, e4);

ClassicAssert.IsFalse(e4.Handled);

}

Если данные разрешены то они будут отображаться в поле. Данный тест это демонстрирует(рисунок 27).

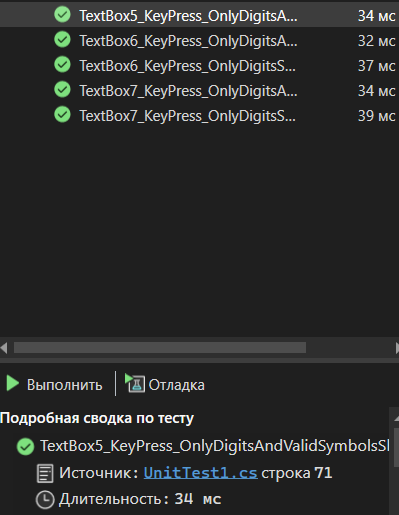


Рисунок 27 – Допустимые

Тест 3

[Test]

public void TextBox6\_KeyPress\_OnlyDigitsAllowed()

{

var e = new KeyPressEventArgs('a');

register.textBox6\_KeyPress(register.textBox6, e);

ClassicAssert.IsTrue(e.Handled);

}

Данный тест выполняет ввод неверных символов в поле ИНН на форме регистрации (рисунок 28).

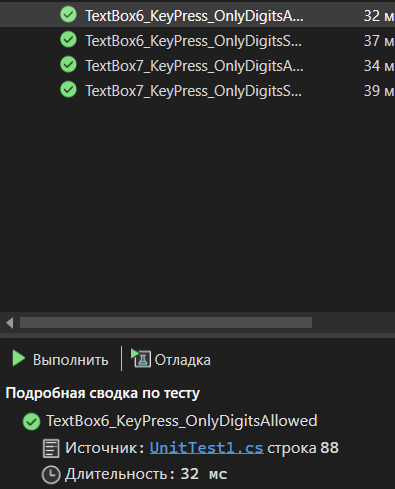


Рисунок 28 - Неправильное значение

Тест 4.

[Test]

public void TextBox6\_KeyPress\_OnlyDigitsShouldBeAllowed()

{

var e1 = new KeyPressEventArgs('1');

register.textBox6\_KeyPress(register.textBox6, e1);

ClassicAssert.IsFalse(e1.Handled);

}

Тест 4 проверяет ввод валидных данных в поле ИНН на форме регистрации (рисунок 29).

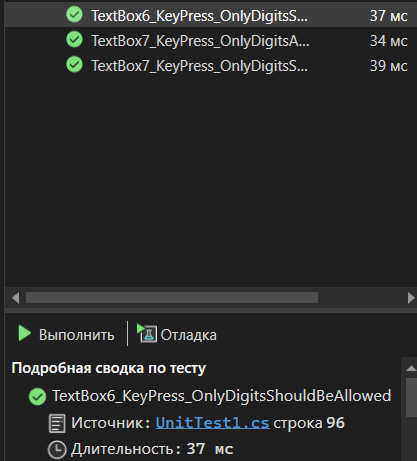


Рисунок 29 – ИНН допустимые

Тест 5

[Test]

public void TextBox7\_KeyPress\_OnlyDigitsAllowed()

{

var e = new KeyPressEventArgs('a');

register.textBox7\_KeyPress(register.textBox7, e);

ClassicAssert.IsTrue(e.Handled);

}

Данный тест проверяет ввод не валидных данных в поле Банковские реквизиты на форме регистрации(рисунок 30).

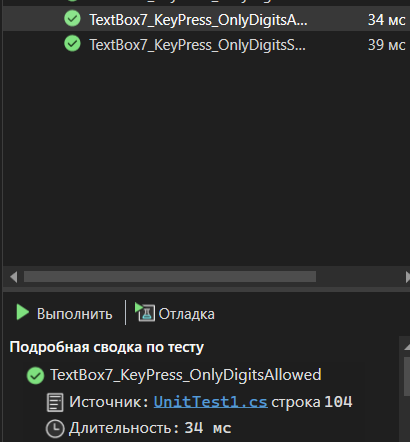


Рисунок 30 – Банковские реквизиты недопустимые

Тест 6

[Test]

public void TextBox7\_KeyPress\_OnlyDigitsShouldBeAllowed()

{

var e1 = new KeyPressEventArgs('1');

register.textBox7\_KeyPress(register.textBox7, e1);

ClassicAssert.IsFalse(e1.Handled);

}

Тест показывает ввод правильных данных в поле Банковские реквизиты на форме регистрации(рисунок 31).

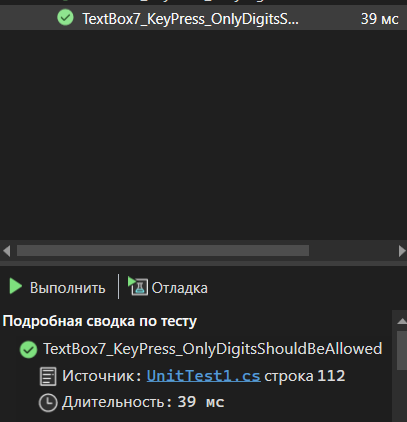


Рисунок 31 – Банковские реквизиты допустимое

Тест 7

[Test]

public void ComboBox1\_KeyPress\_ShouldBlockInput()

{

var e = new KeyPressEventArgs('a');

register.comboBox1\_KeyPress(register.comboBox1, e);

ClassicAssert.IsTrue(e.Handled);

}

Тест показывает заблокирован ли ввод в поле выбора на форме регистрации(рисунок 32).

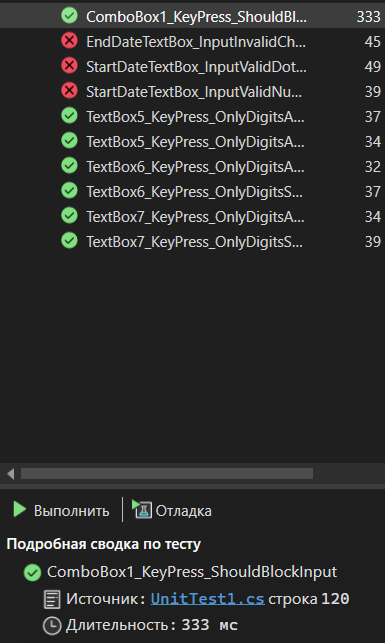


Рисунок 32 – Ввод данных

Тест 8

[Test]

public void StartDateTextBox\_InputValidNumber\_Allowed()

{

var textBox = form.Controls.Find("textBox\_StartDate", true)[0] as TextBox;

var keyPressEvent = new KeyPressEventArgs('1');

form.TextBox\_KeyPress(textBox, keyPressEvent);

ClassicAssert.IsFalse(keyPressEvent.Handled, "Ввод цифры должен быть разрешен.");

}

Тест проверяет ввод цифр в поле с датой на форме менеджера(рисунок 33).

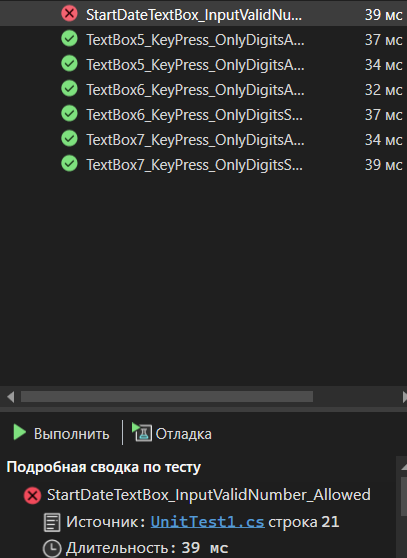


Рисунок 33 – дата допустимые

Тест 9

[Test]

public void StartDateTextBox\_InputValidDot\_Allowed()

{

var textBox = form.Controls.Find("textBox\_StartDate", true)[0] as TextBox;

textBox.Text = "2022";

var keyPressEvent = new KeyPressEventArgs('.');

form.TextBox\_KeyPress(textBox, keyPressEvent);

ClassicAssert.IsFalse(keyPressEvent.Handled, "Ввод точки должен быть разрешен.");

}

Проверка ввода точки в поле с датой на форме менеджера(рисунок 34).

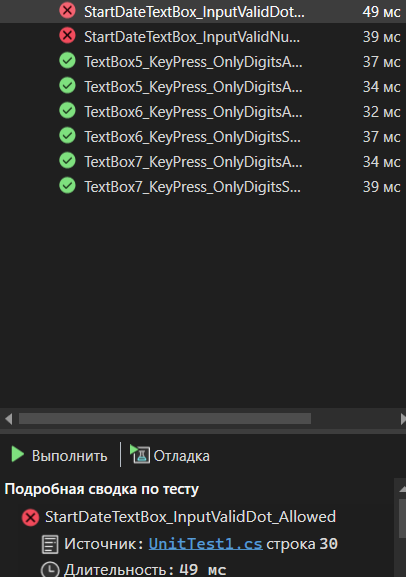


Рисунок 34 – Ввод точки

Тест 10

[Test]

public void EndDateTextBox\_InputInvalidCharacters\_NotAllowed()

{

var textBox = form.Controls.Find("textBox\_EndDate", true)[0] as TextBox;

var keyPressEvent = new KeyPressEventArgs('$');

form.TextBox\_KeyPress(textBox, keyPressEvent);

ClassicAssert.IsTrue(keyPressEvent.Handled, "Ввод специальных символов должен быть запрещен.");

}

Проверка на ввод иных символов в поле даты на форме менеджера(рисунок 35).

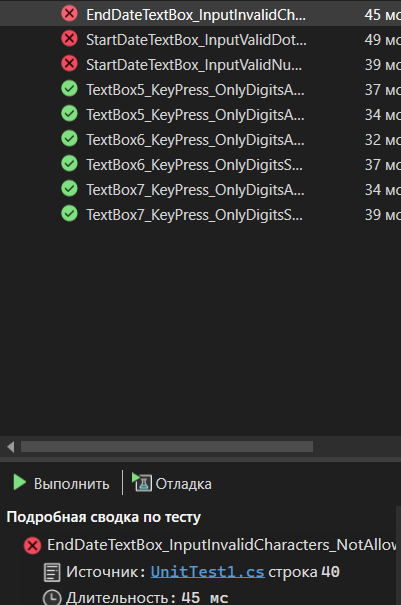


Рисунок 35 – Запрет

Задание 6. GitHub. Оценка проекта

Выгружаем приложение на открытый github репозиторий (Рисунок 36), делая приложение открытым для общего доступа. Надежность повышается, и другие разработчики смогут вносить свои правки, делая приложение лучше, развивая его функционал добавляя новые возможности.

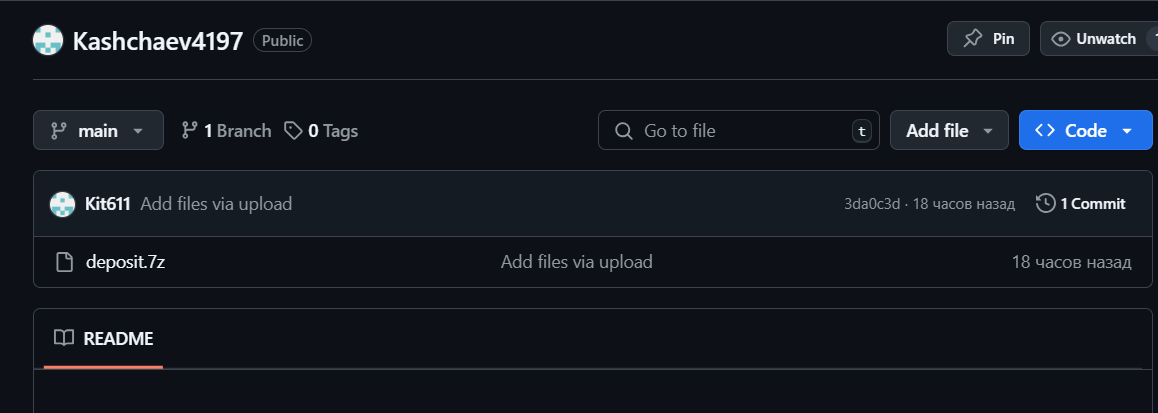


Рисунок 36 – Проект на GitHub

ЗАКЛЮЧЕНИЕ

В ходе учебной практики были освоены основные аспекты разработки программного продукта начиная от разработки технического задания и описания предметной области до тестирования конечного продукта – написания тестовых случаев и разработки Nunit тестов. Освоены навыки создания макетов, типа: Wireframe и Mockup. Получены навыки создания UML-диаграмм и моделей данных.

СПИСОК ИСПОЛЬЗОВАНЫХ ИСТОЧНИКОВ

1. Официальная документация C# [Электронный ресурс]; Microsoft Learn. Режим доступа: https://learn.microsoft.com/ru-ru/dotnet/csharp/, свободный.
2. Официальная документация Windows Forms [Электронный ресурс];Microsoft Learn. Режим доступа: https: https://learn.microsoft.com/ru-ru/dotnet/desktop/winforms/?view=netdesktop-8.0, свободный.
3. Официалная документация MS SQL [Электронный ресурс];Microsoft Learn. Режим доступа: https://learn.microsoft.com/ru-ru/sql/?view=sql-server-ver16, свободный.

**АТТЕСТАЦИОННЫЙ ЛИСТ**

**ПО УЧЕБНОЙ ПРАКТИКЕ УП.02.01 (ПО ПРОФИЛЮ СПЕЦИАЛЬНОСТИ)**

по профессиональному модулю ПМ.02 «Осуществление интеграции программных модулей»

(код и наименование)

Специальность09.02.07 Информационные системы и программирование

(код и наименование специальности)

Студент(ка) 4 курса 42919/7 группы

Кащаев Никита Витальевич

(Фамилия, имя, отчество)

Место прохождения практики: УВЦ, пр. Энгельса, 23

(наименование и адрес организации)

Период прохождения практики

с «28» октября 2024 г. по «09» ноября 2024 г.

**Виды и качество выполнения работ**

|  |  |  |
| --- | --- | --- |
| **Виды выполненных работ обучающимся**  **во время практики** | **Объем работ, час.** | **Качество выполнения работ (оценка по пятибалльной системе)** |
| **Тема 1.** Участие в выработке требований к программному обеспечению. | 12 |  |
| **Тема 2.** Стадии проектирования программного обеспечения. Проектирование UML-диаграмм. | 15 |  |
| **Тема 3.** Конструирование пользовательского интерфейса. Разработка модулей программного обеспечения. | 25 |  |
| **Тема 4.** Тестирование и отладка программных модулей. Модульное тестирование. | 12 |  |
| **Тема 5.** Разработка программной документации. | 9 |  |

**Характеристика учебной/профессиональной деятельности обучающегося во время учебной практики (по профилю специальности):**

Общие и профессиональные компетенции, предусмотренные программой практики, освоены **/** не освоены.

(нужное подчеркнуть)

Итоговая оценка по практике \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Руководитель практики Хисамутдинова А.С. \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(Ф.И.О.) (подпись)

Дата «09» ноября 2024 г.