**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 42919/5 группы

Гончарова Никиты Юрьевича

(Фамилия, имя, отчество)

Место прохождения практики: УВЦ, пр. Энгельса, 23

(наименование и адрес организации)

Период прохождения практики

с «04» сентября 2023 г. по «16» сентября 2023 г.

Руководитель практики Хисамутдинова А.С.

(подпись) (расшифровка подписи)

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

Санкт-Петербург

2023

**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 42919/5 группы

Гончаров Никита Юрьевич

(Фамилия, имя, отчество)

Место прохождения практики: УВЦ, пр. Энгельса, 23

(наименование и адрес организации)

Период прохождения практики

с «04» сентября 2023 г. по «16» сентября 2023 г.

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

* Участие в выработке требований к программному обеспечению;
* Стадии проектирования программного обеспечения;
* Разработка модулей программного обеспечения;
* Тестирование программных модулей и их интеграции;
* Разработка программной документации и стандарты кодирования.

**Индивидуальное задание: ВАРИАНТ 17**

Задание выдал «04» сентября 2023 г. \_\_\_\_\_\_\_\_\_\_\_\_ Хисамутдинова А.С.

(подпись) (Ф.И.О.)

Задание получил «04» сентября 2023 г. \_\_\_\_\_\_\_\_\_\_\_\_ Гончаров Н.Ю(подпись) (Ф.И.О.)

**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 42919/5 группы

Гончаров Никита Юрьевич

(Фамилия, имя, отчество)

Место прохождения практики: УВЦ, пр. Энгельса, 23

(наименование и адрес организации)

Период прохождения практики

с «04» сентября 2023 г. по «16» сентября 2023 г.

Руководитель практики Хисамутдинова А.С.

(подпись) (расшифровка подписи)

Итоговая оценка по практике

Санкт-Петербург

2023

**Содержание дневника**

|  |  |  |
| --- | --- | --- |
| **Дата** | **Виды выполненных работ и заданий по программе практики** | **Подпись руководителя практики** |
| **1** | **2** | **3** |
| 04.09.2023 | Анализ предметной области. Выявление требований к программе. Разработка технического задания |  |
| 05.09.2023 | UML. Проектирование диаграммы вариантов использования. UML. Проектирование диаграммы последовательности. UML. Проектирование диаграммы активности |  |
| 06.09.2023 | Моделирование структуры ПО. Проектирование инфологической и даталогической модели данных |  |
| 07.09.2023 | Проектирование интерфейса пользователя. Создание Wireframe эскизов. Разработка дизайна программы в соответствии с руководством по стилю |  |
| 08.09.2023 | Разработка базы данных. Разработка словаря данных |  |
| 11.09.2023 | Создание приложения. Форма авторизации. Создание приложения. Форма заказов |  |
| 12.09.2023 | Создание приложения. Основные формы приложения. Разработка библиотеки классов |  |
| 13.09.2023 | Подготовка отчетов и выгрузка документов для печати. Отладка программных модулей |  |
| 14.09.2023 | Модульное тестирование. Создание тестовых случаев |  |
| 15.09.2023 | Интеграционное тестирование. Разработка самодокументирующегося кода |  |
| 16.09.2023 | Инспекция кода на соответствие стандартам кодирования. Подготовка отчета и размещение результатов в репозитории контроля версий |  |

**CОДЕРЖАНИЕ**

1. Введение......................................................................................................5;
2. Общая часть ................................................................................................8;
   1. Проектирование системы………………………………………..…8;
      1. ER-диаграмма…….………………………………………..…8;
      2. Словарь данных…..………………………………………..…9;
      3. Диаграмма вариантов использования..…………….………10;
      4. Диаграмма активности; ..…………………………...………10;
      5. Диаграмма последовательности…………………………....12;
   2. Создание базы данных и заполнение таблиц данными………….13;
   3. Разработка библиотеки и подключение её к проекту……………13;
   4. Разработка приложения…………………………………………...13;
   5. Тестирование приложения………………………………………..18;
      1. Создание тестовых случаев………………………………...18;
      2. Модульное тестирование………………………….………..18;
   6. Оценка проекта………………………………………………….…18;
   7. Выгрузка готового проекта в репозиторий Git…………….……19;
3. Заключение……………………………………………………………...20;
4. Приложения……………………………………………………………..21;

**Введение**

Выпускники средних специальных учебных заведений должны владеть современными знаниями и умениями, необходимыми для создания эффективного и конкурентоспособного программного обеспечения.

Чрезвычайно важным является овладение всеми этапами процесса создания программ как продукции производственно-технического назначения, в том числе овладение навыками качественного оформления программной документации.

Важной составной частью подготовки учащихся в этом направлении является прохождение учебной практики студентами.

В настоящей работе содержатся результаты прохождения мной учебной практики в виде индивидуального задания по варианту от 04.09.2023 по 16.09.2023.

По выданному мне номеру варианта №17, я получил описание задания и предметной области. (рис. 1)

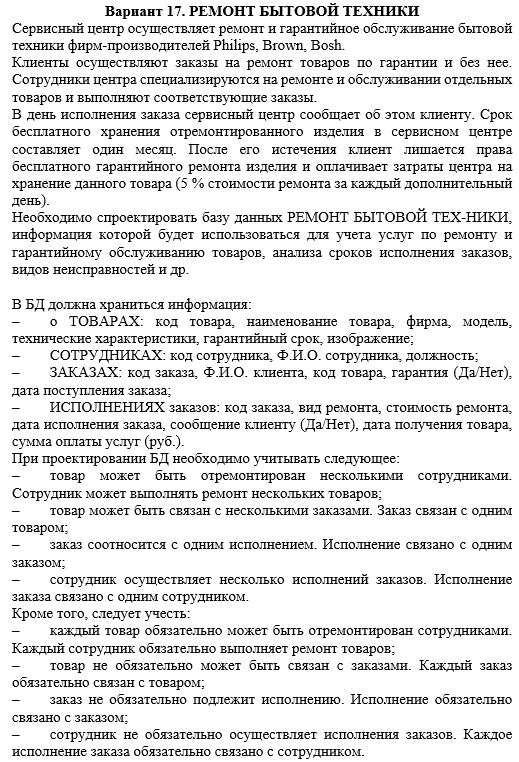


Рисунок 1 – описание предметной области “РЕМОНТ БЫТОВОЙ ТЕХНИКИ”

Для выданной мне предметной области была спроектирована, разработана и протестирована автоматизированная информационная система.

Этапами моей работы выступили:

1. Проектирование системы:
   1. Диаграмма вариантов использования;
   2. Диаграмма активности;
   3. Диаграмма последовательности;
2. Создание базы данных;
3. Разработка клиентского приложения:
   1. Форма авторизации и главной формы;
   2. Форма работы с заказом и соответствующие ей подчиненные формы;
   3. Формы просмотра и редактирования остальных данных;
   4. Создание отчетов и выгрузка их в документ;
4. Тестирование приложения:
   1. Создание тестовых случаев;
   2. Модульное тестирование;

Оформление результатов;**2.Общая часть**

**2.1.Проектирование системы**

**2.1.1.ER-диаграмма**

ER-диаграмма отображает основные сущности в предметной области их атрибуты взаимосвязи между собой. (Рис. 2)



Рисунок 2 – ER-диаграмма ПО “РЕМОНТ БЫТОВОЙ ТЕХНИКИ”

**2.1.2.Словарь данных**

Проектировка словаря данных позволила продумать и обозначить типы данных для атрибутов каждой сущности. (Рис. 3)



Рисунок 3 – Словарь данных

**2.1.3.Диаграмма вариантов использования**

Диаграмма прецедентов помогает представить доступным схематичным языком представить варианты взаимодействия систем\людей с иформационной системой. (Рис. 4)



Рисунок 4 – Диаграмма вариантов использования

**2.1.4.Диаграмма активности**

Диаграмма активности при проектирование позволяет представить последовательные дествия разных частей системы относительно друг друга и последовательности выполнения. (Рис. 5)



Рисунок 5 – Диаграмма активности

**2.1.5.Диаграмма последовательности**

Диаграмма последовательности позволяет разделять процессы не только между частями системы во время исполнения задач, но и позволяет наблюдать за процессами относительно времени выполнения их взаимного выполнения. (Рис. 6)(Рис. 7)



Рисунок 6 – Диаграмма последовательности для вхола пользователя



Рисунок 7 - Диаграмма последовательности для оформления исполнения заказов

**2.2.Создание базы данных и заполнение таблиц данными**

Мной была создана и заполнена база данных, при помощи использования платформуы Microsoft SQL Server. (см. ПРИЛОЖЕНИЕ 1) (см. ПРИЛОЖЕНИЕ 2)

**2.3.Разработка библиотеки и подключение её к проекту**

Был изучен предоставленный пример библиотеки классов для реализации контроллера базы данных. (см. ПРИЛОЖЕНИЕ 3)

**2.4.Разработка приложение**

Была разработана информационная система на языке программирования C# в среде разработки Microsoft Visual Studio с подкдючение к СУБД Microsoft SQL Server Management Studio.(см ПРИЛОЖЕНИЕ 4)

Был разработан интерфейс формы входа. При запуске приложения окно входа – первое, что видит пользователь. На ней пользователю предлагается ввести свой логин и пароль. Только после удачной авторизации пользователь получает доступ к соответствующим его роли модулям системы. (Рис 8)

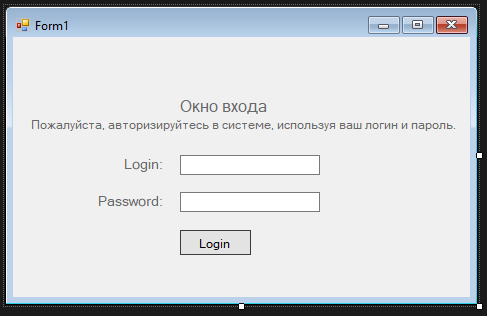


Рисунок 8 – Форма входа в систему

Реализованы формы для трёх разных ролей в системе: администратор, клиент, пользователь(работник). (Рис. 9)

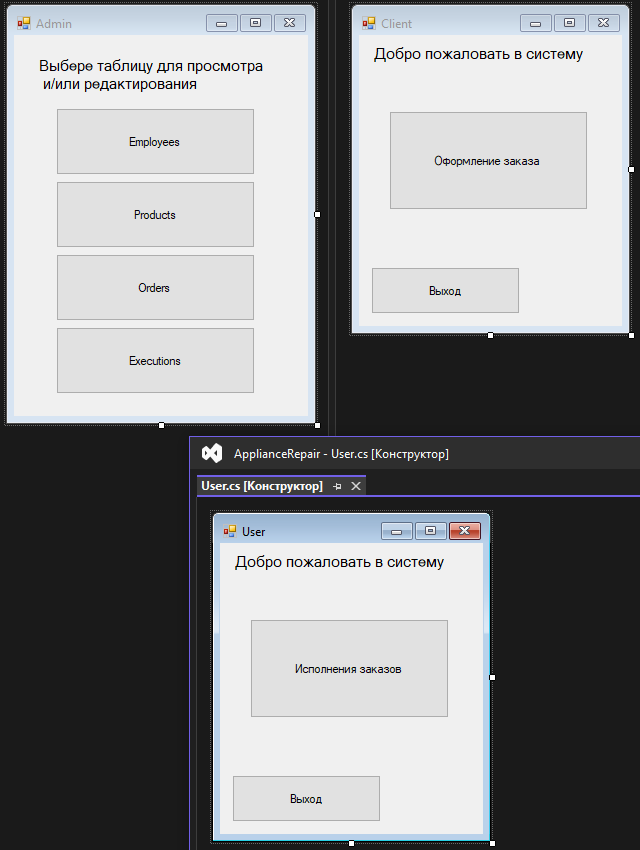


Рисунок 9 – Формы для входа под разными ролями в системе

Интерфейс для администрирования системы представлен четырьмя формами(см. Рис. 10), каждая из которых реализует добавление, изменеине и удаление полей из основных таблиц БД.

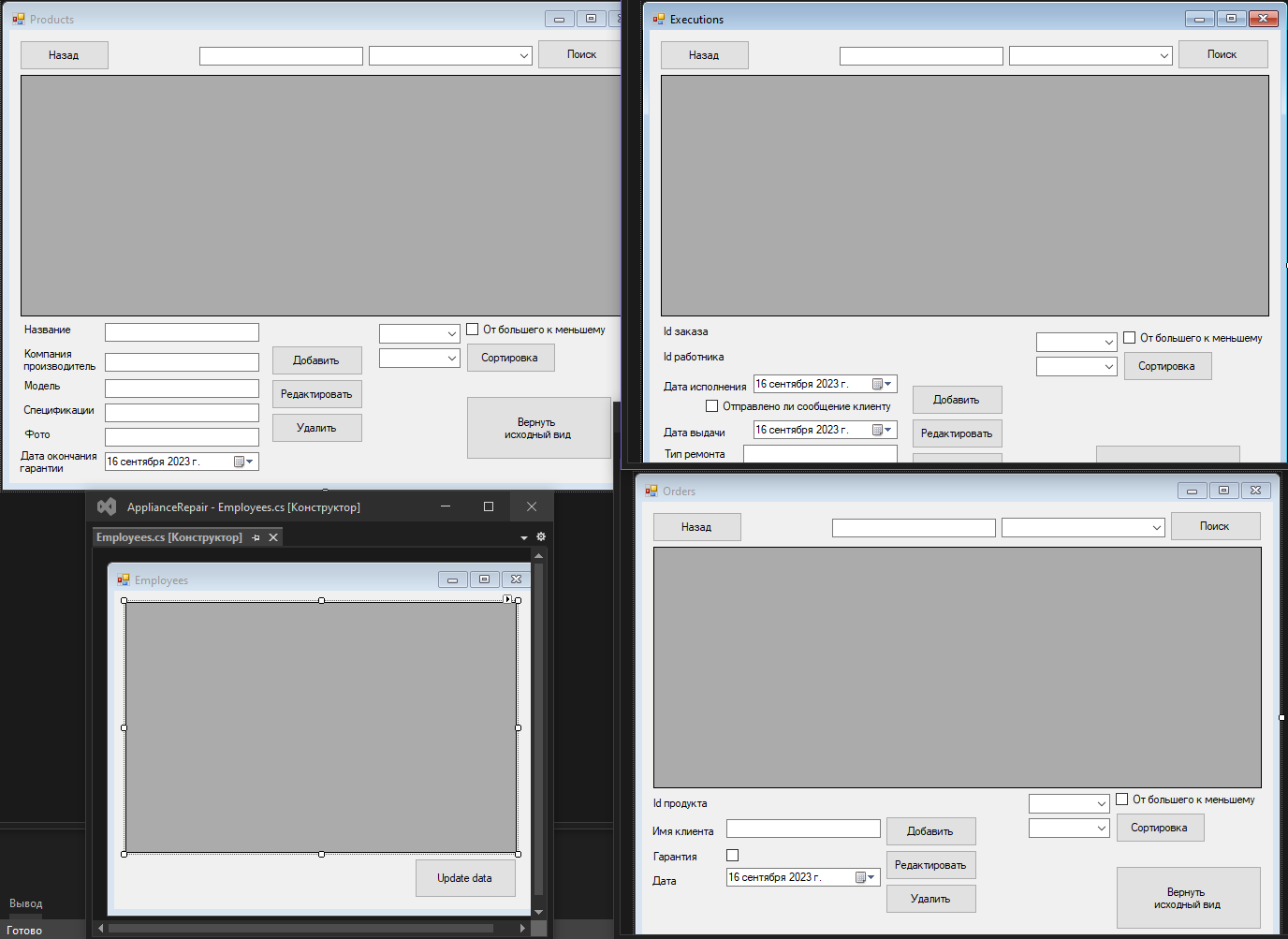


Рисунок 10 – Формы администратора

Форма оформления заказа на ремонт бытовой техникик (Рис. 11)

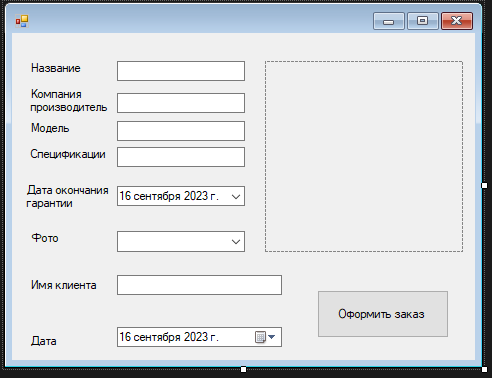


Рисунок 11 – Форма оформление заказа

Работники могут просматривать и редактировать таблицу исполнения заказов через админский интерфейс добавления/изменения/удаления полей из таблицы исполнения.

(Рис 12)

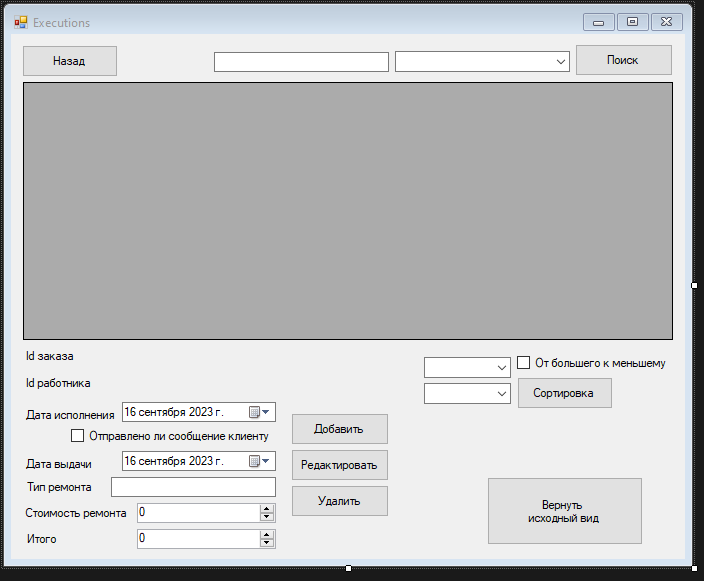


Рисунок 12 – Форма таблицы исполнения заказов

В каждой форме администратора реализованы поиск в таблицы по значению поля и сортировка таблицы по нескольким критериям.(Рис 13)

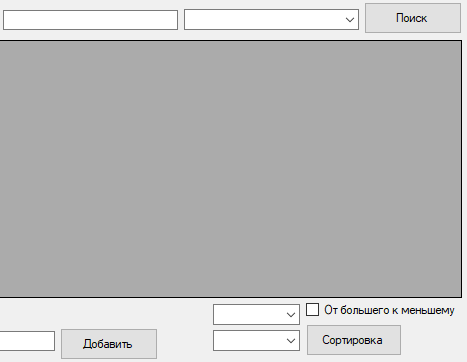


Рисунок 13 – Реализация поиска и сортировки

Реализация подключения системы к БД с помощью кода смотри в ПРИЛОЖЕНИИ 4.

**2.5.Тестирование приложения**

**2.5.1.Создание тестовых случаев**

Для пользовательского интерфейса и основных функций системы разработаны и проведены тесткейсы.(см ПРИЛОЖЕНИЕ 5)

**2.5.2.Модульное тестирование**

Был изучен предоставленный пример тестирования библиотеки классов для реализации контроллера базы данных. (см. ПРИЛОЖЕНИЕ 3)

**4.6.Оценка проекта**

**2.7.Выгрузка готового проекта в репозиторий Git**

Информационная система и учебные материалы выгружены в Git репозиторий.(Рис. 14)

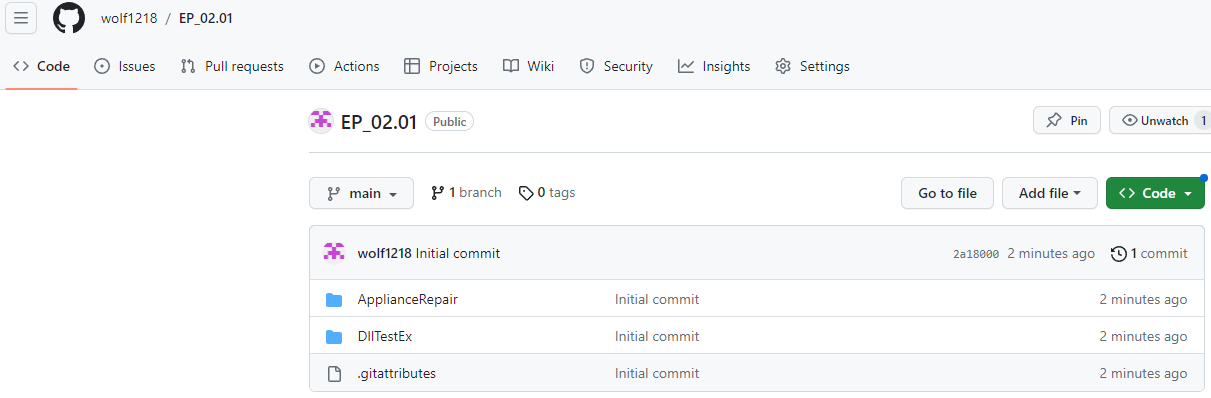


Рисунок 14 – Git репозиторий

**3.Заключение**

Во время прохождения учебной практики, разработки информационной системы и программного продукта по выданной предметной области были разобраны все этапы процесса создания программ как продукции производственно-технического назначения.

Получены навыки пректирования и создания базы данных, реализации клиентского приложения и тестирования продукта.

ПРИЛОЖЕНИЕ 1

Create Table Employees (

EmployeeKey int NOT NULL IDENTITY(1,1) PRIMARY KEY,

FullName varchar(100) ,

PositionAtWork varchar (100),

);

Create Table Products (

ProductKey int NOT NULL IDENTITY(1,1) PRIMARY KEY,

Name varchar (100) NOT NULL,

Company varchar (100) NOT NULL,

Model varchar (100) NOT NULL,

Specifications varchar (100) NOT NULL,

WarrantyDateEnd date NOT NULL,

Image varchar (100),

);

Create Table Orders (

OrderKey int NOT NULL IDENTITY(1,1) PRIMARY KEY,

ProductKey int NOT NULL,

ClientFullName varchar (100) NOT NULL,

Warranty bit NOT NULL,

OrderDate date NOT NULL,

FOREIGN KEY (ProductKey) REFERENCES Products(ProductKey)

);

Create Table Executions (

ExecutionKey int NOT NULL IDENTITY(1,1) PRIMARY KEY,

OrderKey int NOT NULL,

EmployeeKey int NOT NULL,

ExecutionDate date NOT NULL,

MessageToClient bit NOT NULL,

DateOfReceipt date NOT NULL,

RepairType varchar (100) NOT NULL,

RepairCost int NOT NULL,

PaymentResult int NOT NULL,

FOREIGN KEY (OrderKey) REFERENCES Orders(OrderKey),

FOREIGN KEY (EmployeeKey) REFERENCES Employees(EmployeeKey)

);

Create Table Users (

Login varchar (50) NOT NULL PRIMARY KEY,

Password varchar(50) ,

Role varchar (50)

);

Create Table LogInHistory (

LogInId int NOT NULL IDENTITY(1,1) PRIMARY KEY,

Login varchar (50) NOT NULL,

LogDate datetime NOT NULL,

FOREIGN KEY (Login) REFERENCES Users(Login)

);

ПРИЛОЖЕНИЕ 2

INSERT INTO Employees VALUES

('Ванасян Григорий Ервандович', 'Обслуживание стрильных машин'),

('Губин Дмитрий Васильевич', 'Обслуживание посудомоечных машин'),

('Сидоров Иван Григорьевич', 'Обслуживание стиральных машин'),

('Лобов Никита Гротович', 'Обслуживание стиральных машин'),

('Огнев Василий Иванович', 'Менеджер');

INSERT INTO Products VALUES

('Стиральная машина', 'Bosch', 'VN12-2', 'Мощная стиральная машина', '2023-07-08', NULL),

('посудомоечная машина', 'Bosch', '211-318', 'Мощная посудомоечная машина', '2023-12-20', NULL),

('Стиральная машина', 'Ferra', 'v001', 'Слабая стиральная машина', '2025-12-05', NULL),

('Стиральная машина', 'Bosch', 'VN1', 'Средняя стиральная машина', '2022-12-05', NULL),

('Пылесос', 'Bolba', 'VN12-2', 'Сильный пылесос', '2023-07-08', NULL);

INSERT INTO Orders VALUES

(2, 'Васичкин Василий Васильвич', 1, '2022-11-11'),

(1, 'Васичкин Василий Васильвич', 1, '2023-09-04'),

(5, 'Докторов Иван Григорьевич', 0, '2021-09-25'),

(3, 'Иванов Иван Иванович', 1, '2023-07-01'),

(4, 'Добров Сергей Владимирович', 0, '2022-12-06');

INSERT INTO Executions VALUES

(1, 2, '2023-09-05', 1, '2023-09-06', 'Замена барабана', 1000, 1000),

(2, 1, '2023-09-05', 1, '2023-09-06', 'Починка', 500, 500);

ПРИЛОЖЕНИЕ 3

using System;

using System.Collections.Generic;

using System.Data.SqlClient;

namespace DatabaseController

{

public class Database

{

private Database() { }

private static Database Instance;

private string sqlConnectionString;

private delegate object Handler(SqlCommand cmd);

private void ExecuteCommand(Handler handler, string query, out object result, Parameter[] parameters = null)

{

result = null;

try

{

SqlConnection connection = new SqlConnection(sqlConnectionString);

connection.Open();

SqlCommand cmd = new SqlCommand(query, connection);

if (parameters != null)

{

foreach (Parameter param in parameters)

{

cmd.Parameters.AddWithValue(param.Name, param.Value);

}

}

if (handler != null) result = handler.Invoke(cmd);

connection.Close();

}

catch (Exception ex) { /\* ignore \*/ }

}

/// <summary>

/// Получить единственный экземпляр Database.

/// </summary>

/// <param name="sqlconnectionString">строка подключения к БД</param>

/// <returns></returns>

public static Database GetInstance(string sqlconnectionString = null)

{

// Если экзепляр Instance не существует, инициализировать новый.

if (Instance == null)

{

Instance = new Database();

}

if (sqlconnectionString != null) Instance.sqlConnectionString = sqlconnectionString;

return Instance;

}

/// <summary>

/// Выполнить запрос.

/// </summary>

/// <param name="query">SQL запрос</param>

/// <param name="parameters">Список параметров</param>

/// <returns>Количество затронутых строк</returns>

public int Execute(string query, Parameter[] parameters = null)

{

// Выполнить команду cmd.ExecuteNonQuery() в качестве функции handler

Handler handler = (SqlCommand cmd) => cmd.ExecuteNonQuery();

ExecuteCommand(handler, query, out object result, parameters);

return result != null ? (int)result : 0;

}

/// <summary>

/// Получить скалярное значение.

/// </summary>

/// <param name="query">SQL запрос</param>

/// <param name="parameters">Список параметров</param>

/// <returns>Первый столбец первой строки</returns>

public object GetScalar(string query, Parameter[] parameters = null)

{

// Выполнить команду cmd.ExecuteScalar() в качестве функции handler

Handler handler = (SqlCommand cmd) => cmd.ExecuteScalar();

ExecuteCommand(handler, query, out object result, parameters);

return result;

}

/// <summary>

/// Получить таблицу значений.

/// </summary>

/// <param name="query"></param>

/// <param name="parameters"></param>

/// <returns>Массив из строк, в каждой из которых - значения столбцов</returns>

public object[][] GetRowsData(string query, Parameter[] parameters = null)

{

// Выполнить пакет команд в качестве функции handler

Handler handler = (SqlCommand cmd) =>

{

SqlDataReader dr = cmd.ExecuteReader();

List<object[]> list = new List<object[]>();

// Перезаписать значения из `SqlDataReader dr` в list

while (dr.Read())

{

object[] values = new object[dr.FieldCount];

for (int i = 0; i < dr.FieldCount; i++)

{

values[i] = dr.GetValue(i);

}

list.Add(values);

}

// вернуть массив объектов вместо листа.

return list.ToArray();

};

ExecuteCommand(handler, query, out object result, parameters);

return (object[][])result;

}

}

}

using DatabaseController;

using Microsoft.VisualStudio.TestTools.UnitTesting;

using System;

namespace UnitTestProject1

{

[TestClass]

public class TestDatabase

{

private string sqlstr = @"Data Source=localhost\SQLEXPRESS;Initial Catalog=test\_db;Integrated Security=true;";

private Database database;

public TestDatabase()

{

database = Database.GetInstance(sqlstr);

}

[TestMethod]

public void TestExecuteSqlScript()

{

object result = database.GetScalar("SELECT 124");

Assert.AreEqual(result.ToString(), "124");

}

[TestMethod]

public void ExecuteCommandWithParameters()

{

string query = "INSERT INTO users (login, password) VALUES (@login, @password)";

Parameter[] par =

{

new Parameter("@login", "user1"),

new Parameter("@password", "12345")

};

int rows = database.Execute(query, par);

Assert.AreEqual(rows, 1);

}

[TestMethod]

public void GetScalarFromExistingTable()

{

string query = "SELECT COUNT(\*) FROM users";

object result = database.GetScalar(query);

Assert.IsNotNull(

result,

"Не удалось получить число записей."

);

}

[TestMethod]

public void GetRowsFromDatabase()

{

string query = "SELECT id, login FROM users";

object[][] result = database.GetRowsData(query);

CollectionAssert.AllItemsAreNotNull(

result,

"Не удалось получить записи из таблицы."

);

}

}

}

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace DatabaseController

{

public class Parameter

{

public string Name { get; set; }

public string Value { get; set; }

public Parameter(string name, object value)

{

// Избавиться от всех лишних "@" в строке.

name = name.Replace("@", "");

Name = $"@{name}";

Value = value.ToString();

}

}

}

ПРИЛОЖЕНИЕ 4

using System;

using System.Data.SqlClient;

using System.Windows.Forms;

namespace ApplianceRepair

{

public partial class Authorization : Form

{

public Authorization()

{

InitializeComponent();

}

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

{

string login = maskedTextBox1.Text;

string password = maskedTextBox2.Text;

int commNum = 0;

if (maskedTextBox1.Text != "" && maskedTextBox2.Text != "")

{

SqlConnection connection = new SqlConnection(@"Server=DESKTOP-8FID4AN\SQLEXPRESS;Database=Goncharov\_419/5\_17;Trusted\_Connection=True;");

connection.Open();

SqlCommand command = new SqlCommand($"SELECT \* FROM [Users] WHERE [Login] = {login} and [Password] = {password} ", connection);

SqlDataReader reader = command.ExecuteReader();

reader.Read();

if (reader.HasRows)

{

string getRole = (string)reader.GetValue(2);

reader.Close();

string sqlCurrentTime = DateTime.Now.ToString("yyyyMMdd HH:mm:ss.fff");

try

{

command = new SqlCommand($"INSERT INTO [LogInHistory] VALUES ({login}, '{sqlCurrentTime}')", connection);

commNum = command.ExecuteNonQuery();

}

catch (Exception ex)

{

MessageBox.Show(ex.Message, "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

if (getRole == "Admin")

{

this.Hide();

Admin admin = new Admin(this);

admin.Show();

}

else if (getRole == "User")

{

this.Hide();

User user = new User(this);

user.Show();

}

else if (getRole == "Client")

{

this.Hide();

Client client = new Client(this);

client.Show();

}

else

{

MessageBox.Show("Вы не зарегистрированы, обратитесь к администратору");

}

}

else

{

MessageBox.Show("Введенные данные неверны", "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Hand);

maskedTextBox1.Clear();

maskedTextBox2.Clear();

}

connection.Close();

}

else

{

MessageBox.Show("Заполните все поля для авторизации!");

}

}

}

}

using System;

using System.Windows.Forms;

namespace ApplianceRepair

{

public partial class Admin : Form

{

private Authorization authorization;

public Admin(Authorization au)

{

InitializeComponent();

authorization = au;

}

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

{

Employees updateEmployees = new Employees(this);

updateEmployees.Show();

this.Hide();

}

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

{

Products updateProducts = new Products(this);

updateProducts.Show();

this.Hide();

}

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

{

Executions updateExecutions = new Executions(this);

updateExecutions.Show();

this.Hide();

}

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

{

Orders updateOrders = new Orders(this);

updateOrders.Show();

this.Hide();

}

private void Admin\_FormClosed(object sender, FormClosedEventArgs e)

{

authorization.Show();

}

}

}

using System;

using System.Windows.Forms;

namespace ApplianceRepair

{

public partial class Client : Form

{

private Authorization authorization;

public Client(Authorization au)

{

InitializeComponent();

authorization = au;

}

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

{

authorization.Close();

}

private void Client\_FormClosed(object sender, FormClosedEventArgs e)

{

authorization.Show();

}

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

{

ClientOrders newOrder = new ClientOrders(this);

newOrder.Show();

this.Hide();

}

}

}

using System;

using System.Windows.Forms;

namespace ApplianceRepair

{

public partial class User : Form

{

private Authorization authorization;

public User(Authorization au)

{

InitializeComponent();

this.authorization = au;

}

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

{

authorization.Close();

}

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

{

Executions updateExecutions = new Executions(this);

updateExecutions.Show();

this.Hide();

}

private void User\_FormClosed(object sender, FormClosedEventArgs e)

{

authorization.Show();

}

}

}

using System;

using System.Data.SqlClient;

using System.Drawing;

using System.Windows.Forms;

namespace ApplianceRepair

{

public partial class ClientOrders : Form

{

private Client client;

int b;

public ClientOrders(Client cl)

{

InitializeComponent();

client = cl;

}

private void ClientOrders\_FormClosed(object sender, FormClosedEventArgs e)

{

client.Show();

}

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

{

if(comboBox1.SelectedIndex != 0)

{

pictureBox1.Image = Image.FromFile(comboBox1.Text);

pictureBox1.SizeMode = PictureBoxSizeMode.StretchImage;

}

else

{

pictureBox1.Image = null;

}

}

private void ClientOrders\_Load(object sender, EventArgs e)

{

comboBox1.SelectedIndex = 0;

}

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

{

if (dateTimePicker1.Value < dateTimePicker2.Value)

{

b = 0;

}

else

{

b = 1;

}

using (SqlConnection connection = new SqlConnection(@"Server=DESKTOP-8FID4AN\SQLEXPRESS;Database=Goncharov\_419/5\_17;Trusted\_Connection=True;"))

{

try

{

connection.Open();

int getKey;

SqlCommand command = new SqlCommand

($"INSERT INTO Products VALUES ('{textBox1.Text}', '{textBox2.Text}', '{textBox3.Text}', '{textBox4.Text}'," +

$" '{dateTimePicker1.Value}', '{comboBox1.Text}')", connection);

command.ExecuteNonQuery();

command = new SqlCommand

($"SELECT ProductKey FROM Products WHERE Name = '{textBox1.Text}' AND Company = '{textBox2.Text}' AND " +

$" Model = '{textBox3.Text}' AND Specifications = '{textBox4.Text}'", connection);

command.ExecuteNonQuery();

SqlDataReader reader = command.ExecuteReader();

reader.Read();

getKey = (int)reader[0];

reader.Close();

command = new SqlCommand

($"INSERT INTO Orders VALUES ({getKey}, '{textBox5.Text}', {b}, '{dateTimePicker2.Value}')", connection);

command.ExecuteNonQuery();

MessageBox.Show("Ожидайте подтверждения заказ", "Заявка оформлена", MessageBoxButtons.OK, MessageBoxIcon.Information);

}

catch (Exception ex)

{

MessageBox.Show(ex.Message, "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

}

}

}

using System;

using System.Data;

using System.Data.SqlClient;

using System.Windows.Forms;

namespace ApplianceRepair

{

public partial class Employees : Form

{

private Admin ad;

public Employees(Admin admin)

{

InitializeComponent();

this.ad = admin;

}

SqlConnection connection;

SqlDataAdapter adapter;

DataSet ds;

SqlCommandBuilder commBuild;

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

{

try

{

commBuild = new SqlCommandBuilder(adapter);

adapter.Update(ds);

MessageBox.Show("Бaза данных обновлена", "Информация", MessageBoxButtons.OK, MessageBoxIcon.Information);

}

catch (Exception ex)

{

MessageBox.Show(ex.Message, "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

private void Employees\_Load(object sender, EventArgs e)

{

try

{

connection = new SqlConnection(@"Server=DESKTOP-8FID4AN\SQLEXPRESS;Database=Goncharov\_419/5\_17;Trusted\_Connection=True;");

adapter = new SqlDataAdapter("SELECT \* FROM Employees", connection);

ds = new DataSet();

adapter.Fill(ds);

dataGridView1.AutoGenerateColumns = true;

dataGridView1.DataSource = ds.Tables[0];

dataGridView1.Columns[0].Visible = false;

}

catch (Exception ex)

{

MessageBox.Show(ex.Message, "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

private void Employees\_FormClosed(object sender, FormClosedEventArgs e)

{

ad.Show();

}

}

}

using System;

using System.Data;

using System.Data.SqlClient;

using System.Drawing;

using System.Linq;

using System.Windows.Forms;

namespace ApplianceRepair

{

public partial class Executions : Form

{

private Admin admin;

private User user;

SqlConnection connection;

SqlDataAdapter adapter;

DataSet ds;

SqlCommandBuilder commBuild;

DataView dv;

int b = 0;

System.Windows.Forms.ComboBox id1 = new System.Windows.Forms.ComboBox();

System.Windows.Forms.ComboBox id2 = new System.Windows.Forms.ComboBox();

public Executions(Admin ad)

{

InitializeComponent();

admin = ad;

}

public Executions(User us)

{

InitializeComponent();

user = us;

}

private void Executions\_Load(object sender, EventArgs e)

{

try

{

connection = new SqlConnection(@"Server=DESKTOP-8FID4AN\SQLEXPRESS;Database=Goncharov\_419/5\_17;Trusted\_Connection=True;");

connection.Open();

adapter = new SqlDataAdapter("SELECT \* FROM Executions", connection);

ds = new DataSet();

adapter.Fill(ds);

dataGridView1.AutoGenerateColumns = true;

dataGridView1.DataSource = ds.Tables[0];

foreach (System.Windows.Forms.ComboBox cb in this.Controls.OfType<System.Windows.Forms.ComboBox>())

{

foreach (DataColumn column in ds.Tables[0].Columns)

{

cb.Items.Add(column.ColumnName);

}

}

comboBox1.SelectedIndex = 0;

comboBox2.SelectedIndex = 0;

comboBox3.SelectedIndex = 0;

this.Controls.Add(id1);

this.Controls.Add(id2);

id1.Location = new Point(90, 312);

id1.Size = new Size(175, 21);

id2.Location = new Point(90, 339);

id2.Size = new Size(175, 21);

SqlCommand command = new SqlCommand("SELECT OrderKey FROM Orders", connection);

SqlDataReader reader = command.ExecuteReader();

int i = 0;

while (reader.Read())

{

id1.Items.Add(reader.GetInt32(0));

i++;

}

reader.Close();

id1.SelectedIndex = 0;

command = new SqlCommand("SELECT EmployeeKey FROM Employees", connection);

reader = command.ExecuteReader();

while (reader.Read())

{

id2.Items.Add(reader.GetInt32(0));

i++;

}

id2.SelectedIndex = 0;

reader.Close();

}

catch (Exception ex)

{

MessageBox.Show(ex.Message, "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

private void Executions\_FormClosed(object sender, FormClosedEventArgs e)

{

if (admin != null)

{

admin.Show();

}

else

{

user.Show();

}

}

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

{

this.Close();

}

private void checkBox2\_CheckedChanged(object sender, EventArgs e)

{

if (checkBox2.Checked)

{

b = 1;

}

else

{

b = 0;

}

}

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

{

try

{

ds.Tables[0].Rows.Add(1, id1.Text, id2.Text, dateTimePicker1.Value, b, dateTimePicker2.Value, textBox1.Text, numericUpDown1.Value, numericUpDown2.Value);

commBuild = new SqlCommandBuilder(adapter);

adapter.Update(ds);

dataGridView1.DataSource = ds.Tables[0];

}

catch (Exception ex)

{

MessageBox.Show(ex.Message, "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

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

{

try

{

var curruentRow = dataGridView1.CurrentCell.RowIndex;

DataRow updateRow = ds.Tables[0].Rows[curruentRow];

updateRow[1] = id1.Text;

updateRow[2] = id2.Text;

updateRow[3] = dateTimePicker1.Value;

updateRow[4] = b;

updateRow[5] = dateTimePicker2.Value;

updateRow[6] = textBox1.Text;

updateRow[7] = numericUpDown1.Value;

updateRow[8] = numericUpDown2.Value;

commBuild = new SqlCommandBuilder(adapter);

adapter.Update(ds);

}

catch (Exception ex)

{

MessageBox.Show(ex.Message, "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

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

{

try

{

var curruentRow = dataGridView1.CurrentCell.RowIndex;

ds.Tables[0].Rows[curruentRow].Delete();

commBuild = new SqlCommandBuilder(adapter);

adapter.Update(ds);

dataGridView1.DataSource = ds.Tables[0];

}

catch (Exception ex)

{

MessageBox.Show(ex.Message, "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

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

{

try

{

if (checkBox1.Checked)

{

ds.Tables[0].DefaultView.Sort = $"{comboBox2.Text} DESC, {comboBox3.Text} DESC";

}

else

{

ds.Tables[0].DefaultView.Sort = $"{comboBox2.Text} ASC, {comboBox3.Text} ASC";

}

DataTable dt = ds.Tables[0].DefaultView.ToTable();

dataGridView1.DataSource = dt;

}

catch (Exception ex)

{

MessageBox.Show(ex.Message, "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

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

{

dataGridView1.DataSource = ds.Tables[0];

}

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

{

try

{

dv = ds.Tables[0].AsDataView();

if (textBox6.Text == "")

{

dataGridView1.DataSource = ds.Tables[0];

}

else

{

dv.RowFilter = $"{comboBox1.Text} = '{textBox6.Text}'";

dataGridView1.DataSource = dv;

}

}

catch (Exception ex)

{

MessageBox.Show(ex.Message, comboBox1.SelectedText, MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

}

}

using System;

using System.Data;

using System.Data.SqlClient;

using System.Drawing;

using System.Linq;

using System.Windows.Forms;

namespace ApplianceRepair

{

public partial class Orders : Form

{

private Admin admin;

SqlConnection connection;

SqlDataAdapter adapter;

DataSet ds;

SqlCommandBuilder commBuild;

DataView dv;

int b = 0;

System.Windows.Forms.ComboBox id = new System.Windows.Forms.ComboBox();

public Orders(Admin ad)

{

InitializeComponent();

admin = ad;

}

private void Orders\_Load(object sender, EventArgs e)

{

try

{

connection = new SqlConnection(@"Server=DESKTOP-8FID4AN\SQLEXPRESS;Database=Goncharov\_419/5\_17;Trusted\_Connection=True;");

connection.Open();

adapter = new SqlDataAdapter("SELECT \* FROM Orders", connection);

ds = new DataSet();

adapter.Fill(ds);

dataGridView1.AutoGenerateColumns = true;

dataGridView1.DataSource = ds.Tables[0];

foreach (System.Windows.Forms.ComboBox cb in this.Controls.OfType<System.Windows.Forms.ComboBox>())

{

foreach (DataColumn column in ds.Tables[0].Columns)

{

cb.Items.Add(column.ColumnName);

}

}

comboBox1.SelectedIndex = 0;

comboBox2.SelectedIndex = 0;

comboBox3.SelectedIndex = 0;

this.Controls.Add(id);

id.Location = new Point(80, 312);

id.Size = new Size(175, 21);

SqlCommand command = new SqlCommand("SELECT ProductKey FROM Products", connection);

SqlDataReader reader = command.ExecuteReader();

int i = 0;

while (reader.Read())

{

id.Items.Add(reader.GetInt32(0));

i++;

}

id.SelectedIndex = 0;

reader.Close();

}

catch (Exception ex)

{

MessageBox.Show(ex.Message, "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

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

{

this.Close();

}

private void Orders\_FormClosed(object sender, FormClosedEventArgs e)

{

admin.Show();

}

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

{

dataGridView1.DataSource = ds.Tables[0];

}

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

{

try

{

ds.Tables[0].Rows.Add(1, id.Text, textBox1.Text, b, dateTimePicker1.Value);

commBuild = new SqlCommandBuilder(adapter);

adapter.Update(ds);

dataGridView1.DataSource = ds.Tables[0];

}

catch (Exception ex)

{

MessageBox.Show(ex.Message, "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

private void checkBox2\_CheckedChanged(object sender, EventArgs e)

{

if (checkBox2.Checked)

{

b = 1;

}

else

{

b = 0;

}

}

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

{

try

{

var curruentRow = dataGridView1.CurrentCell.RowIndex;

DataRow updateRow = ds.Tables[0].Rows[curruentRow];

updateRow[1] = id.Text;

updateRow[2] = textBox1.Text;

updateRow[3] = b;

updateRow[4] = dateTimePicker1.Value;

commBuild = new SqlCommandBuilder(adapter);

adapter.Update(ds);

}

catch (Exception ex)

{

MessageBox.Show(ex.Message, "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

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

{

try

{

var curruentRow = dataGridView1.CurrentCell.RowIndex;

ds.Tables[0].Rows[curruentRow].Delete();

commBuild = new SqlCommandBuilder(adapter);

adapter.Update(ds);

dataGridView1.DataSource = ds.Tables[0];

}

catch (Exception ex)

{

MessageBox.Show(ex.Message, "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

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

{

try

{

if (checkBox1.Checked)

{

ds.Tables[0].DefaultView.Sort = $"{comboBox2.Text} DESC, {comboBox3.Text} DESC";

}

else

{

ds.Tables[0].DefaultView.Sort = $"{comboBox2.Text} ASC, {comboBox3.Text} ASC";

}

DataTable dt = ds.Tables[0].DefaultView.ToTable();

dataGridView1.DataSource = dt;

}

catch (Exception ex)

{

MessageBox.Show(ex.Message, "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

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

{

try

{

dv = ds.Tables[0].AsDataView();

if (textBox6.Text == "")

{

dataGridView1.DataSource = ds.Tables[0];

}

else

{

dv.RowFilter = $"{comboBox1.Text} = '{textBox6.Text}'";

dataGridView1.DataSource = dv;

}

}

catch (Exception ex)

{

MessageBox.Show(ex.Message, comboBox1.SelectedText, MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

}

}

using System;

using System.Data;

using System.Data.SqlClient;

using System.Linq;

using System.Windows.Forms;

namespace ApplianceRepair

{

public partial class Products : Form

{

private Admin admin;

public Products(Admin ad)

{

InitializeComponent();

admin = ad;

}

SqlConnection connection;

SqlDataAdapter adapter;

DataSet ds;

SqlCommandBuilder commBuild;

DataView dv;

private void Products\_Load(object sender, EventArgs e)

{

try

{

connection = new SqlConnection(@"Server=DESKTOP-8FID4AN\SQLEXPRESS;Database=Goncharov\_419/5\_17;Trusted\_Connection=True;");

adapter = new SqlDataAdapter("SELECT \* FROM Products", connection);

ds = new DataSet();

adapter.Fill(ds);

dataGridView1.AutoGenerateColumns = true;

dataGridView1.DataSource = ds.Tables[0];

foreach (ComboBox cb in this.Controls.OfType<ComboBox>())

{

foreach (DataColumn column in ds.Tables[0].Columns)

{

cb.Items.Add(column.ColumnName);

}

}

comboBox1.SelectedIndex = 0;

comboBox2.SelectedIndex = 0;

comboBox3.SelectedIndex = 0;

}

catch (Exception ex)

{

MessageBox.Show(ex.Message, "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

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

{

try

{

ds.Tables[0].Rows.Add(1, textBox1.Text, textBox2.Text, textBox3.Text, textBox4.Text, dateTimePicker1.Value, textBox5.Text);

commBuild = new SqlCommandBuilder(adapter);

adapter.Update(ds);

dataGridView1.DataSource = ds.Tables[0];

}

catch (Exception ex)

{

MessageBox.Show(ex.Message, "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

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

{

try

{

var curruentRow = dataGridView1.CurrentCell.RowIndex;

DataRow updateRow = ds.Tables[0].Rows[curruentRow];

updateRow[1] = textBox1.Text;

updateRow[2] = textBox2.Text;

updateRow[3] = textBox3.Text;

updateRow[4] = textBox4.Text;

updateRow[5] = dateTimePicker1.Value;

updateRow[6] = textBox5.Text;

commBuild = new SqlCommandBuilder(adapter);

adapter.Update(ds);

}

catch (Exception ex)

{

MessageBox.Show(ex.Message, "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

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

{

try

{

var curruentRow = dataGridView1.CurrentCell.RowIndex;

ds.Tables[0].Rows[curruentRow].Delete();

commBuild = new SqlCommandBuilder(adapter);

adapter.Update(ds);

dataGridView1.DataSource = ds.Tables[0];

}

catch (Exception ex)

{

MessageBox.Show(ex.Message, "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

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

{

try

{

dv = ds.Tables[0].AsDataView();

if (textBox6.Text == "")

{

dataGridView1.DataSource = ds.Tables[0];

}

else

{

dv.RowFilter = $"{comboBox1.Text} = '{textBox6.Text}'";

dataGridView1.DataSource = dv;

}

}

catch (Exception ex)

{

MessageBox.Show(ex.Message, comboBox1.SelectedText, MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

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

{

try

{

if (checkBox1.Checked)

{

ds.Tables[0].DefaultView.Sort = $"{comboBox2.Text} DESC, {comboBox3.Text} DESC";

}

else

{

ds.Tables[0].DefaultView.Sort = $"{comboBox2.Text} ASC, {comboBox3.Text} ASC";

}

DataTable dt = ds.Tables[0].DefaultView.ToTable();

dataGridView1.DataSource = dt;

}

catch (Exception ex)

{

MessageBox.Show(ex.Message, comboBox1.SelectedText, MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

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

{

dataGridView1.DataSource = ds.Tables[0];

}

private void Products\_FormClosed(object sender, FormClosedEventArgs e)

{

admin.Show();

}

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

{

this.Close();

}

}

}

ПРИЛОЖЕНИЕ 5

|  |  |
| --- | --- |
| **Название проекта** | РЕМОНТ БЫТОВОЙ ТЕХНИКИ |
| **Рабочая версия** | 1.0 |
| **Имя тестирующего** | Гончаров Никита Юрьевич |
| **Дата(ы) теста** | 15.09.2023 – 16.09.2023 |

|  |  |
| --- | --- |
| **Тестовый пример #** | TC\_UI\_1 |
| **Приоритет тестирования** | Средний |
| **Заголовок/название теста** | Вывод соответствующей информации при неуспешной попытке входа в систему |
| **Краткое изложение теста** | Тест должен выявить способность системы оповещать пользователя об неверно указаных данных на этапе авторизации |
| **Этапы теста** | 1. Запуск программы 2. Ввод в соответствующие поля значений логина и пароля незарегестрированных в базе данных 3. Активация кнопки Login(войти) |
| **Тестовые данные** | Имя пользователя(Login) = 4  Пароль(Password) = 4 |
| **Ожидаемый результат** | Вывод системой сообщения о неверности введёных данных |
| **Фактический результат** | Вывод системой отдельного диалогового окна с сообщением о неверности введёных данных |
| **Статус** | Зачет |
| **Предварительное условие** | - |
| **Постусловие** | - |
| **Примечания/комментарии** | - |

|  |  |
| --- | --- |
| **Тестовый пример #** | TC\_UI\_2 |
| **Приоритет тестирования** | Низкий |
| **Заголовок/название теста** | Прекращение работы приложения после закрытия всех форм |
| **Краткое изложение теста** | Выявления возможности корректного прекращения работы системы и отсутствия утечек памяти |
| **Этапы теста** | 1. Запуск программы 2. Вход в аккаунт администратора 3. Вход в окно редактирования данных 4. Поочерёдное закрытия всех последющих форм от последней до первой |
| **Тестовые данные** | Имя пользователя(Login) = 1  Пароль(Password) = 1 |
| **Ожидаемый результат** | Закрытие всех форм и прекращение работы приложения |
| **Фактический результат** | Корректное прекращение работы программы после закрытия всех форм |
| **Статус** | Зачет |
| **Предварительное условие** | - |
| **Постусловие** | - |
| **Примечания/комментарии** | - |

|  |  |
| --- | --- |
| **Тестовый пример #** | TC\_UI\_3 |
| **Приоритет тестирования** | Низкий |
| **Заголовок/название теста** | Прекращение работы приложения после “выхода” из системы |
| **Краткое изложение теста** | Проверка корректного прекращения работы приложения после использования кнопки “выход”в не администраторских аккаунтах |
| **Этапы теста** | 1. Запуск программы 2. Вход в аккаунт работника 3. Активация кнопки “выход” 4. Поочерёдное закрытия всех последющих форм от последней до первой |
| **Тестовые данные** | Имя пользователя(Login) = 1  Пароль(Password) = 1 |
| **Ожидаемый результат** | Закрытие всех форм и прекращение работы приложения |
| **Фактический результат** | Корректное прекращение работы программы после закрытия всех форм |
| **Статус** | Зачет |
| **Предварительное условие** | - |
| **Постусловие** | - |
| **Примечания/комментарии** | - |

|  |  |
| --- | --- |
| **Тестовый пример #** | TC\_DB\_1 |
| **Приоритет тестирования** | Высокий |
| **Заголовок/название теста** | Оформление заказа через пользователя |
| **Краткое изложение теста** | Проверка корректного сохранения данных из пользовательской системы в бд |
| **Этапы теста** | 1. Запуск программы 2. Вход в аккаунт клиента 3. Вход в форму оформления заказов 4. Выставление тестовых данных во все элементы TextBox 5. Активация кнопки “Оформить заказ” |
| **Тестовые данные** | Все элементы TextBox заполняются значением “1” |
| **Ожидаемый результат** | Вывод системой сообщение об удачном выполнении запроса, сохранение данных в БД |
| **Фактический результат** | Вывод системой сообщения “Ожидайте подтверждения заказа, Заявка оформлена”, сохранение данных в БД по нескошльким таблицам |
| **Статус** | Зачет |
| **Предварительное условие** | - |
| **Постусловие** | - |
| **Примечания/комментарии** | - |

|  |  |
| --- | --- |
| **Тестовый пример #** | TC\_DB\_2.1 |
| **Приоритет тестирования** | Высокий |
| **Заголовок/название теста** | Добавление строки в таблицу базы данных через администратора |
| **Краткое изложение теста** | Проверка корректного сохранения данных из администраторской системы в бд |
| **Этапы теста** | 1. Запуск программы 2. Вход в аккаунт администратора 3. Вход в форму изменения/просмотра сотрудников(Employees) 4. Заполнение таблицы 5. Активация кнопки “Update data” |
| **Тестовые данные** | Все ячейки таблицы заполняются значением “1” |
| **Ожидаемый результат** | Добавление записи в системную таблицу, добавление записи в соответствующую таблицу БД |
| **Фактический результат** | Сохранение введённых данных в системную таблицу, Сохранение соответствующих данных в таблицу Employees базы данных |
| **Статус** | Зачет |
| **Предварительное условие** | - |
| **Постусловие** | - |
| **Примечания/комментарии** | - |

|  |  |
| --- | --- |
| **Тестовый пример #** | TC\_DB\_2.2 |
| **Приоритет тестирования** | Высокий |
| **Заголовок/название теста** | Проверка сохранения связанности данных из БД в системе |
| **Краткое изложение теста** | Проверка соответствия внешних ключей одной таблицы в редакторе связанной таблицы |
| **Этапы теста** | 1. Запуск программы 2. Вход в аккаунт администратора 3. Вход в форму изменения/просмотра исполнений(Executions) 4. Активация выпадающего списка “Id работника” |
| **Тестовые данные** | - |
| **Ожидаемый результат** | В выпадающем спискек находятся все имеющиеся ключи таблицы работников, включая тот, что был создан во время выполнения TC\_DB\_2.1 |
| **Фактический результат** | Наличие всех ключей из таблицы БД Employees |
| **Статус** | Зачет |
| **Предварительное условие** | Успешное выполнение условий TC\_DB\_2.1 |
| **Постусловие** | - |
| **Примечания/комментарии** | - |

|  |  |
| --- | --- |
| **Тестовый пример #** | TC\_DB\_3 |
| **Приоритет тестирования** | Высокий |
| **Заголовок/название теста** | Защита БД от некорректных записей |
| **Краткое изложение теста** | Проверка предусмотренных системой мер по защите БД от записи некорректных значений |
| **Этапы теста** | 1. Запуск программы 2. Вход в аккаунт администратора 3. Вход в форму изменения/просмотра продуктов(Products) 4. Активация кнопки “Добавить” |
| **Тестовые данные** | - |
| **Ожидаемый результат** | Предупреждение пользователя о пустых полях для добавления, остановка функции добавления |
| **Фактический результат** | Запись пустых полей в системную таблицу и таблицу Products БД в виде значений “” |
| **Статус** | НеЗачет |
| **Предварительное условие** | - |
| **Постусловие** | - |
| **Примечания/комментарии** | - |

**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 42919/5 группы

Гончаров Никита Юрьевич

(Фамилия, имя, отчество)

Место прохождения практики: УВЦ, пр. Энгельса, 23

(наименование и адрес организации)

Период прохождения практики

с «04» сентября 2023 г. по «16» сентября 2023 г.

**Виды и качество выполнения работ**

|  |  |  |
| --- | --- | --- |
| **Виды выполненных работ обучающимся**  **во время практики** | **Объем работ, час.** | **Качество выполнения работ (оценка по пятибалльной системе)** |
| Участие в выработке требований к программному обеспечению | 9 |  |
| Стадии проектирования программного обеспечения | 21 |  |
| Разработка модулей программного обеспечения | 24 |  |
| Тестирование программных модулей и их интеграции | 9 |  |
| Разработка программной документации и стандарты кодирования | 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 «16» сентября 2023 г.