Разработка библиотеки классов

**Первый класс библиотеки DataBase:**

using System;

using System.Collections.Generic;

using System.Data.SqlClient;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using static System.Net.Mime.MediaTypeNames;

namespace ClassLibraryCarService

{

public class DataBase

{

SqlConnection conn = new SqlConnection(@"Data Source = ADCLG1; Initial catalog = z111; Integrated Security=True;");

public void OpenConection()

{

try

{

if (conn.State == System.Data.ConnectionState.Closed)

conn.Open();

}

catch

{

throw new Exception("Неудалось подключиться к базе данных!\nПриложение будет перезапущено");

}

}

public void CloseConection()

{

if (conn.State == System.Data.ConnectionState.Open)

conn.Close();

}

public string SelectInfoToQuery(string query)

{

DataBase dataBase = new DataBase();

SqlCommand command = new SqlCommand(query, dataBase.GetConection());

dataBase.OpenConection();

string result = command.ExecuteScalar() == null ? string.Empty : command.ExecuteScalar().ToString();

dataBase.CloseConection();

return result;

}

public SqlConnection GetConection() => conn;

}

}

**Второй класс библиотеки CarService\_DataInTadle:**

using System;

using System.Collections.Generic;

using System.Data.SqlClient;

using System.Data;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace ClassLibraryCarService

{

public class CarService\_DataInTadle

{

DataBase dataBase = new DataBase();

public void InsertDataInTablecarType(string type)

{

string ComDel = $" Insert into carType(carTypeName) values ('{type}')";

SqlCommand cmd1 = new SqlCommand(ComDel, dataBase.GetConection());

dataBase.OpenConection();

try

{

cmd1.ExecuteNonQuery();

}

catch

{

throw new Exception("Данные не были сохранены!");

}

finally

{

dataBase.CloseConection();

}

}

public void UpDateDataInTablecarType(int id, string type)

{

string ComDel = $" UpDate carType set carTypeName = '{type}' where carTypeID = {id}";

SqlCommand cmd1 = new SqlCommand(ComDel, dataBase.GetConection());

dataBase.OpenConection();

try

{

cmd1.ExecuteNonQuery();

}

catch

{

throw new Exception("Данные не были сохранены!");

}

finally

{

dataBase.CloseConection();

}

}

}

}

Тестирование приложения

Аннотация теста:

|  |  |
| --- | --- |
| **Название проекта** | CarService |
| **Рабочая версия** | 1.0 |
| **Имя тестирующего** | Любовь |
| **Дата(ы) теста** | 05.11.2024 |

Тест кейс #1:

|  |  |
| --- | --- |
| **Тест кейс #** | 1 |
| **Приоритет тестирования** | Высокий |
| **Заголовок/название теста** | Позитивная авторизация пользователя в системе |
| **Краткое изложение теста** | Запуск приложения, ввод данных в поля авторизации, вход в систему |
| **Этапы теста** | 1. Запуск приложения 2. Ввод входных данных: логин и пароля 3. Нажатие на «Войти» |
| **Тестовые данные** | login: login1  password: pass1 |
| **Ожидаемый результат** | Пользователю выведено приветственное окно входа в систему |
| **Фактический результат** | Пользователю выведено приветственное окно входа в систему |
| **Статус** | Зачет |
| **Предварительное условие** | Убедиться, что пользователь с такими входными данными существует |
| **Постусловие** | Приветственное окно с личными данными пользователя |
| **Примечания/комментарии** |  |

Тест кейс #2:

|  |  |
| --- | --- |
| **Тест кейс #** | 2 |
| **Приоритет тестирования** | Высокий |
| **Заголовок/название теста** | Негативная авторизация пользователя в системе |
| **Краткое изложение теста** | Запуск приложения, ввод неверных данных в поля авторизации |
| **Этапы теста** | 1. Запуск приложения 2. Ввод входных неверных данных: логин и пароля 3. Нажатие на «Войти» |
| **Тестовые данные** | login: login10  password: pass10 |
| **Ожидаемый результат** | Вывод сообщения «Неверные входные данные!» |
| **Фактический результат** | Вывод сообщения «Неверные входные данные!» |
| **Статус** | Зачет |
| **Предварительное условие** | Убедиться, что пользователь с такими входными данными не существует |
| **Постусловие** | Для последующей попытки входа необходимо ввести капчу |
| **Примечания/комментарии** |  |

Тест кейс #3:

|  |  |
| --- | --- |
| **Тест кейс #** | 3 |
| **Приоритет тестирования** | Высокий |
| **Заголовок/название теста** | Успешное изменение существующей заявки |
| **Краткое изложение теста** | Изменение выбранной заявки |
| **Этапы теста** | 1. Выбор строки нужной заявки 2. Нажатие на «Изменение заявки» 3. Заполнение всех полей в выведенном окне 4. Нажатие на «Изменение заявки» |
| **Тестовые данные** | Номер: 7  Тип машины: Легковая  Модель машины: Hyundai Avante (CN7)  Проблема: Поменять масло. |
| **Ожидаемый результат** | Сообщение об успешном сохранении данных |
| **Фактический результат** | Сообщение об успешном сохранении данных |
| **Статус** | Зачет |
| **Предварительное условие** | Пользователь вошел в систему, как заказчик, открыл форму «Заявки» |
| **Постусловие** | С измененной заявкой могут работать другие пользователи |
| **Примечания/комментарии** |  |

Тест кейс #4:

|  |  |
| --- | --- |
| **Тест кейс #** | 4 |
| **Приоритет тестирования** | Низкий |
| **Заголовок/название теста** | Переход с приветственного окна входа в систему на главное меню |
| **Краткое изложение теста** | открытие приветственного окна, переход на главное меню по истечению 2 минут |
| **Этапы теста** | 1. Вывод приветственного окна 2. Переход в главное меню по истечению 2 минут |
| **Тестовые данные** |  |
| **Ожидаемый результат** | Переход в главное меню по истечению 2 минут с открытия приветственного окна |
| **Фактический результат** | Переход в главное меню по истечению 2 минут с открытия приветственного окна |
| **Статус** | Зачет |
| **Предварительное условие** | Пользователь вошел в систему |
| **Постусловие** | Доступен функционал системы по роли пользователя |
| **Примечания/комментарии** |  |

Тест кейс #5:

|  |  |
| --- | --- |
| **Тест кейс #** | 5 |
| **Приоритет тестирования** | Высокий |
| **Заголовок/название теста** | Успешное создание новой заявки |
| **Краткое изложение теста** | Создание новой заявки |
| **Этапы теста** | 1. Нажатие на «Создание заявки» 2. Заполнение всех полей в выведенном окне 3. Нажатие на «Создание заявки» |
| **Тестовые данные** | Тип машины: Легковая  Модель машины: Hyundai Avante (CN7)  Проблема: Отказали тормоза! |
| **Ожидаемый результат** | Сообщение об успешном сохранении данных |
| **Фактический результат** | Сообщение об успешном сохранении данных |
| **Статус** | Зачет |
| **Предварительное условие** | Пользователь вошел в систему, как заказчик, открыл форму «Заявки» |
| **Постусловие** | С этой заявкой могут работать другие пользователи |
| **Примечания/комментарии** |  |

**Код Тестирования:**

using Microsoft.VisualStudio.TestTools.UnitTesting;

using System;

using ClassLibraryCarService;

using System.Data.SqlClient;

namespace UnitTest\_CarService\_DataInTable

{

[TestClass]

public class UnitTest1

{

[TestMethod]

public void TestMethod\_GetConnection()

{

DataBase dataBase = new DataBase();

SqlConnection result = dataBase.GetConection();

SqlConnection delive = new SqlConnection(@"Data Source = ADCLG1; Initial catalog = z111; Integrated Security=True;");

Assert.AreEqual(result.ToString(), delive.ToString());

}

[TestMethod]

public void TestMethod\_SelectInfoToQuery\_Positiv()

{

DataBase dataBase = new DataBase();

string login = "login1";

string result = dataBase.SelectInfoToQuery($" Select (Convert(Nvarchar(max), DecryptByPassphrase('MyPassword', password))) from [user] where login='{login}'");

string delive = "pass1";

Assert.AreEqual(result, delive);

}

[TestMethod]

public void TestMethod\_SelectInfoToQuery\_Negativ()

{

DataBase dataBase = new DataBase();

string login = "login10";

string result = dataBase.SelectInfoToQuery($" Select (Convert(Nvarchar(max), DecryptByPassphrase('MyPassword', password))) from [user] where login='{login}'");

string delive = string.Empty;

Assert.AreEqual(result, delive);

}

[TestMethod]

public void TestMethod\_InsertDataInTablecarType\_Positiv()

{

DataBase dataBase = new DataBase();

CarService\_DataInTadle dataInTadle = new CarService\_DataInTadle();

int countRowsInTable = int.Parse(dataBase.SelectInfoToQuery($"Select count(\*) from [carType]"));

string type = "Лендровер";

dataInTadle.InsertDataInTablecarType(type);

int result = int.Parse(dataBase.SelectInfoToQuery($"Select count(\*) from [carType]"));

int delive = countRowsInTable+1;

Assert.AreEqual(result, delive);

}

[TestMethod]

public void TestMethod\_UpDateDataInTablecarType\_Positiv()

{

DataBase dataBase = new DataBase();

CarService\_DataInTadle dataInTadle = new CarService\_DataInTadle();

int id = 4;

string type = "Сидан";

dataInTadle.UpDateDataInTablecarType(id, type);

string result = dataBase.SelectInfoToQuery($"Select carTypeName from [carType] where carTypeID = {id}");

string delive = type;

Assert.AreEqual(result, delive);

}

}

}