**Код библиотеки:**

using System;

using System;

using System.Data;

using System.Data.SqlClient;

using System.Windows.Forms;

namespace Library

{

public class BD

{

private string connectionString = "Data Source=192.168.188.11; Initial Catalog=!!!AMetYP; Integrated Security=True;";

private SqlConnection con;

public BD()

{

//con = new SqlConnection("Data Source=DESKTOP-S2UBJ1D\\SQLEXPRESS; Initial Catalog=YP; Integrated Security=True;");

con = new SqlConnection("Data Source=192.168.188.11; Initial Catalog=!!!AMetYP; Integrated Security=True;");

}

public void Con()

{

con.Open();

}

public void closeCon()

{

con.Close();

}

// логин форма

public int LoginUser(string username, string password)

{

string query = "SELECT employeeID FROM Employees WHERE login = @userlogin AND password = @userpassword";

using (SqlCommand cmd = new SqlCommand(query, con))

{

cmd.Parameters.AddWithValue("@userlogin", username);

cmd.Parameters.AddWithValue("@userpassword", password);

con.Open();

object result = cmd.ExecuteScalar();

con.Close();

if (result != null)

return Convert.ToInt32(result);

}

query = "SELECT clientID FROM Clients WHERE login = @userlogin AND password = @userpassword";

using (SqlCommand cmd = new SqlCommand(query, con))

{

cmd.Parameters.AddWithValue("@userlogin", username);

cmd.Parameters.AddWithValue("@userpassword", password);

con.Open();

object result = cmd.ExecuteScalar();

con.Close();

return result != null ? Convert.ToInt32(result) : -1;

}

}

public string RoleUser(string username, string password)

{

string query = "SELECT type FROM Employees WHERE login = @userlogin AND password = @userpassword";

using (SqlCommand cmd = new SqlCommand(query, con))

{

cmd.Parameters.AddWithValue("@userlogin", username);

cmd.Parameters.AddWithValue("@userpassword", password);

con.Open();

object result = cmd.ExecuteScalar();

con.Close();

return result != null ? Convert.ToString(result) : "none";

}

}

public void AddFailurePlus(string login)

{

int? userId = null;

using (SqlCommand cmd = new SqlCommand("SELECT clientID FROM Clients WHERE [login] = @login", con))

{

cmd.Parameters.AddWithValue("@login", login);

con.Open();

var result = cmd.ExecuteScalar();

if (result != null)

{

userId = Convert.ToInt32(result);

}

con.Close();

}

if (userId.HasValue)

{

using (SqlCommand cmd = new SqlCommand("INSERT INTO login\_attmpts (id\_user, login\_date, failure) VALUES (@id\_user, @Date, 1);", con))

{

cmd.Parameters.AddWithValue("@id\_user", userId.Value);

cmd.Parameters.AddWithValue("@Date", DateTime.Now);

con.Open();

cmd.ExecuteNonQuery();

con.Close();

}

}

else

{

using (SqlCommand cmd = new SqlCommand("SELECT employeeID FROM Employees WHERE [login] = @login", con))

{

cmd.Parameters.AddWithValue("@login", login);

con.Open();

var result = cmd.ExecuteScalar();

if (result != null)

{

userId = Convert.ToInt32(result);

}

con.Close();

}

if (userId.HasValue)

{

using (SqlCommand cmd = new SqlCommand("INSERT INTO login\_attmpts (id\_user, login\_date, failure) VALUES (@id\_user, @Date, 1);", con))

{

cmd.Parameters.AddWithValue("@id\_user", userId.Value);

cmd.Parameters.AddWithValue("@Date", DateTime.Now);

con.Open();

cmd.ExecuteNonQuery();

con.Close();

}

}

}

}

public void AddFailureMinus(int userId)

{

using (SqlCommand cmd = new SqlCommand("INSERT INTO login\_attmpts (id\_user, login\_date, failure) VALUES (@id\_user, @Date, 0);", con))

{

cmd.Parameters.AddWithValue("@id\_user", userId);

cmd.Parameters.AddWithValue("@Date", DateTime.Now);

con.Open();

cmd.ExecuteNonQuery();

con.Close();

}

}

// клиент

public void LoadUserInfo(Label labelFIO, Label labelPhone, int userID)

{

Con();

string userInfoQuery = @"

SELECT fio, phone

FROM Clients

WHERE clientID = @userID";

try

{

using (SqlCommand command = new SqlCommand(userInfoQuery, con))

{

command.Parameters.AddWithValue("@userID", userID);

using (SqlDataReader reader = command.ExecuteReader())

{

if (reader.Read())

{

labelFIO.Text = reader["fio"].ToString();

labelPhone.Text = reader["phone"].ToString();

}

}

}

}

catch (SqlException ex)

{

Console.WriteLine($"Ошибка получения данных о пользователе: {ex.Message}");

}

finally

{

if (con.State == ConnectionState.Open)

{

closeCon();

}

}

}

// редактирование заявки для клиента

public bool EditRequest(int requestID, string model, string problem)

{

if (string.IsNullOrWhiteSpace(model))

{

throw new ArgumentException("Модель не может быть пустой.");

}

if (string.IsNullOrWhiteSpace(problem))

{

throw new ArgumentException("Проблема не может быть пустой.");

}

try

{

con.Open();

// Получаем techID по модели

using (SqlCommand command = new SqlCommand("SELECT techID FROM Tech WHERE model = @model", con))

{

command.Parameters.AddWithValue("@model", model);

var result = command.ExecuteScalar();

if (result == null)

{

throw new Exception("Указанная модель не найдена в базе данных!");

}

int techID = (int)result;

// Обновляем заявку

command.CommandText = "UPDATE Requests SET techId = @techId, problem = @problem WHERE requestID = @requestID";

command.Parameters.Clear(); // Очищаем параметры перед повторным использованием

command.Parameters.AddWithValue("@techId", techID);

command.Parameters.AddWithValue("@problem", problem);

command.Parameters.AddWithValue("@requestID", requestID);

int rowsAffected = command.ExecuteNonQuery();

return rowsAffected > 0;

}

}

catch (Exception ex)

{

throw new Exception($"Произошла ошибка при обновлении заявки: {ex.Message}");

}

finally

{

closeCon();

}

}

}

}

**Код тестов:**

using Microsoft.VisualStudio.TestTools.UnitTesting;

using System.Windows.Forms;

using Library;

namespace UnitTests

{

[TestClass]

public class UnitTest1

{

[TestMethod]

public void TestMethod1()

{

string login = "login2";

string pass = "pass2";

int userid = 1;

BD sql = new BD();

int userid\_query = sql.LoginUser(login, pass);

Assert.AreEqual(userid\_query, userid);

}

[TestMethod]

public void TestMethod2()

{

string login = "perinaAD";

string pass = "250519";

BD sql = new BD();

string role = "Оператор";

string role\_query = sql.RoleUser(login, pass);

Assert.AreEqual(role\_query, role);

}

[TestMethod]

public void TestMethod3()

{

string login = "perinaAD";

string pass = "250519";

BD sql = new BD();

string role = "Менеджер";

string role\_query = sql.RoleUser(login, pass);

Assert.AreNotEqual(role\_query, role);

}

[TestMethod]

public void TestMethod4()

{

string login = "login2";

string pass = "pass2";

int userid = 2;

BD sql = new BD();

int userid\_query = sql.LoginUser(login, pass);

Assert.AreNotEqual(userid\_query, userid);

}

[TestMethod]

public void TestMethod5()

{

BD sql = new BD();

int requestID = 1; // ID заявки, которую мы будем редактировать

string newModel = "Ладомир ТА112 белый"; // Новая модель

string newProblem = "Не включается"; // Новая проблема

// Act

bool isUpdated = sql.EditRequest(requestID, newModel, newProblem);

// Assert

Assert.IsTrue(isUpdated, "Заявка не была обновлена.");

// Дополнительно проверим, что данные действительно обновились

string query = "SELECT problem FROM Requests WHERE requestID = @requestID";

using (SqlConnection con = new SqlConnection("Data Source=192.168.188.11; Initial Catalog=!!!AMetYP; Integrated Security=True;"))

{

con.Open();

using (SqlCommand cmd = new SqlCommand(query, con))

{

cmd.Parameters.AddWithValue("@requestID", requestID);

var result = cmd.ExecuteScalar();

Assert.AreEqual(newProblem, result.ToString(), "Проблема не была обновлена в базе данных.");

}

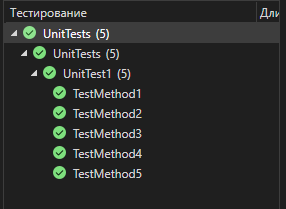
}

}

}

}

**Результат:**



Аннотация теста:

|  |  |
| --- | --- |
| **Название проекта** | Программа для системы учета заявок на ремонт бытовой техники |
| **Рабочая версия** | 1.0 |
| **Имя тестирующего** | Метель Ксения Игоревна |
| **Дата(ы) теста** | 19.12.2024 |

Тест кейс #1

|  |  |
| --- | --- |
| **Тест кейс #** | 1 |
| **Приоритет тестирования** | Высокий |
| **Заголовок/название теста** | Проверка корректной авторизации под ролью «Клиент» |
| **Краткое изложение теста** | Тест проверяет, что пользователь с заданными логином и паролем имеет роль «Клиент» |
| **Этапы теста** | 1. Запустить приложение 2. Ввести логин и пароль клиента 3. Нажать кнопку «Войти» |
| **Тестовые данные** | Логин: login2  Пароль: pass2 |
| **Ожидаемый результат** | После успешной авторизации открывается окно профиля клиента |
| **Фактический результат** | После успешной авторизации открывается окно профиля клиента |
| **Статус** | Зачет |
| **Предварительное условие** | Корректные логин и пароль клиента внесены в БД |
| **Постусловие** | Пользователю доступен функционал роли «Клиент» |
| **Примечания/комментарии** | Нет |

Тест кейс #2

|  |  |
| --- | --- |
| **Тест кейс #** | 2 |
| **Приоритет тестирования** | Высокий |
| **Заголовок/название теста** | Проверка корректной авторизации под ролью «Оператор» |
| **Краткое изложение теста** | Тест проверяет, что пользователь с заданными логином и паролем имеет роль «Оператор» |
| **Этапы теста** | 1. Запустить приложение 2. Ввести логин и пароль оператора 3. Нажать кнопку «Войти» |
| **Тестовые данные** | Логин: perinaAD  Пароль: 250519 |
| **Ожидаемый результат** | После успешной авторизации открывается окно профиля оператора |
| **Фактический результат** | После успешной авторизации открывается окно профиля оператора |
| **Статус** | Зачет |
| **Предварительное условие** | Корректные логин и пароль оператора внесены в БД |
| **Постусловие** | Пользователю доступен функционал роли «Оператор» |
| **Примечания/комментарии** | Нет |

Тест кейс #3

|  |  |
| --- | --- |
| **Тест кейс #** | 3 |
| **Приоритет тестирования** | Высокий |
| **Заголовок/название теста** | Проверка некорректной авторизации под ролью «Менеджер» |
| **Краткое изложение теста** | Тест проверяет, что у пользователя с заданными логином и паролем отсутствует роль «Менеджер» |
| **Этапы теста** | 1. Запустить приложение 2. Ввести некорректные логин и пароль менеджера 3. Нажать кнопку «Войти» |
| **Тестовые данные** | Логин: perinaAD  Пароль: 250519 |
| **Ожидаемый результат** | После неудачной авторизации появляется уведомление о вводе некорректных данных, роль «Менеджер» не присваивается |
| **Фактический результат** | После неудачной авторизации появляется уведомление о вводе некорректных данных, роль «Менеджер» не присваивается |
| **Статус** | Зачет |
| **Предварительное условие** | Данного пользователя с введенными логином и паролем не существует в БД |
| **Постусловие** | Пользователю не доступен функционал роли «Менеджер» |
| **Примечания/комментарии** | Нет |

Тест кейс #4

|  |  |
| --- | --- |
| **Тест кейс #** | 4 |
| **Приоритет тестирования** | Средний |
| **Заголовок/название теста** | Проверка некорректной авторизации под ролью «Клиент» |
| **Краткое изложение теста** | Тест проверяет, что у пользователя с заданными логином и паролем отсутствует роль «Клиент» |
| **Этапы теста** | 1. Запустить приложение 2. Ввести некорректные логин и пароль клиента 3. Нажать кнопку «Войти» |
| **Тестовые данные** | Логин: login2  Пароль: 250519 |
| **Ожидаемый результат** | После неудачной авторизации появляется уведомление о вводе некорректных данных, роль «Клиент» не присваивается |
| **Фактический результат** | После неудачной авторизации появляется уведомление о вводе некорректных данных, роль «Клиент» не присваивается |
| **Статус** | Зачет |
| **Предварительное условие** | Данного пользователя с введенными логином и паролем не существует в БД |
| **Постусловие** | Корректные логин и пароль клиента внесены в БД |
| **Примечания/комментарии** | Нет |

Тест кейс #5

|  |  |
| --- | --- |
| **Тест кейс #** | 5 |
| **Приоритет тестирования** | Высокий |
| **Заголовок/название теста** | Проверка корректного редактирования заявки |
| **Краткое изложение теста** | Тест проверяет, что авторизованный клиент может корректно отредактировать заявку |
| **Этапы теста** | 1. Открыть форму редактирования заявки 2. внести новые данные |
| **Тестовые данные** | Id: 1  Новая модель: Ладомир ТА112 белый  Новая проблема: Не включается |
| **Ожидаемый результат** | Заявка успешно обновлена и отображает новые данные |
| **Фактический результат** | Заявка успешно обновлена и отображает новые данные |
| **Статус** | Зачет |
| **Предварительное условие** | Редактируемая заявка с Id 1 существует в БД, корректные логин и пароль клиента внесены в БД |
| **Постусловие** | Система возвращает к списку, где доступна обновленная информация |
| **Примечания/комментарии** | Нет |