Аннотация теста:

|  |  |
| --- | --- |
| **Название проекта** | Разработка программного модуля для учета заявок на бронирование номеров гостиницы |
| **Рабочая версия** | 1.0 |
| **Имя тестирующего** | Фрич А.М. |
| **Дата(ы) теста** | 02.10.2024 |

Тестовый пример #1:

|  |  |
| --- | --- |
| **Тестовый пример #** | TC\_UI\_1 |
| **Приоритет тестирования** | Высокий |
| **Заголовок/название теста** | Проверка авторизации с действительным именем пользователя и паролем. |
| **Краткое изложение теста** | Проверка возможности входа в систему с действительными учетными данными. |
| **Этапы теста** | 1. Открыть форму авторизации нажатием на кнопку «Войти» 2. Ввести действительное имя пользователя. 3. Ввести действительный пароль. 4. Нажать кнопку «Войти». |
| **Тестовые данные** | - Имя пользователя: admin  - Пароль: admin |
| **Ожидаемый результат** | Пользователь успешно авторизуется и перенаправляется на главную форму системы управления гостиницей |
| **Фактический результат** | Пользователь успешно авторизуется и перенаправляется на главную страницу системы управления гостиницей |
| **Статус** | Зачет |
| **Предварительное условие** | Пользователь должен быть зарегистрирован в системе |
| **Постусловие** | Пользователь должен быть авторизован и иметь доступ к функциональности системы. |
| **Примечания/комментарии** | Нет |

Тестовый пример #2:

|  |  |
| --- | --- |
| **Тестовый пример #** | TC\_UI\_2 |
| **Приоритет тестирования** | Высокий |
| **Заголовок/название теста** | Проверка процесса регистрации нового пользователя. |
| **Краткое изложение теста** | Проверка возможности регистрации нового пользователя в системе. |
| **Этапы теста** | 1. Открыть форму регистрации нажатием на кнопку «Зарегистрироваться» 2. Ввести Логин 3. Ввести пароль 4. Нажать кнопку «Зарегистрироваться» |
| **Тестовые данные** | - Имя пользователя: newuser123  - Пароль: newuser123 |
| **Ожидаемый результат** | Пользователь успешно зарегистрирован и перенаправлен на главную форму входа. |
| **Фактический результат** | Пользователь успешно зарегистрирован и перенаправлен на главную форму входа. |
| **Статус** | Зачет |
| **Предварительное условие** | Пользователь должен быть не зарегистрирован в системе (логин не должен быть занят) |
| **Постусловие** | Пользователь должен быть зарегистрирован и иметь возможность авторизоваться. |
| **Примечания/комментарии** | Нет |

Тестовый пример #3:

|  |  |
| --- | --- |
| **Тестовый пример #** | TC\_UI\_3 |
| **Приоритет тестирования** | Высокий |
| **Заголовок/название теста** | Проверка возможности изменения таблиц в БД администратором. |
| **Краткое изложение теста** | Проверка возможности администратора изменять таблицы в базе данных. |
| **Этапы теста** | 1. Авторизоваться как администратор. 2. Перейти в раздел управления базой данных. 3. Выбрать таблицу для изменения. 4. Внести изменения в таблицу. 5. Сохранить изменения нажатием кнопки «Обновить» |
| **Тестовые данные** | - Имя пользователя: admin  - Пароль: admin123  - Таблица: Дополнительные услуги  - Изменения: Добавить новую услугу. |
| **Ожидаемый результат** | Изменения успешно сохранены в БД и отображаются в таблице. |
| **Фактический результат** | Изменения успешно сохранены в БД и отображаются в таблице. |
| **Статус** | Зачет |
| **Предварительное условие** | Администратор должен быть авторизован. |
| **Постусловие** | Изменения должны быть сохранены в базе данных. |
| **Примечания/комментарии** | Нет |

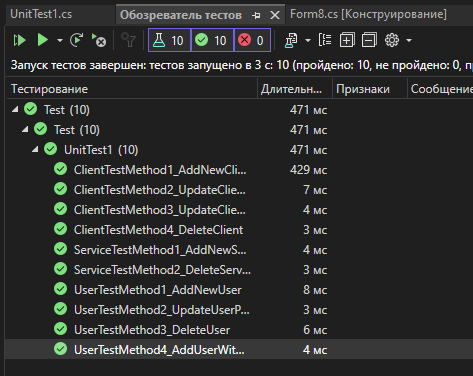
Тестовый пример #4:

|  |  |
| --- | --- |
| **Тестовый пример #** | TC\_UI\_4 |
| **Приоритет тестирования** | Средний |
| **Заголовок/название теста** | Проверка возможности ввода личных данных клиентом. |
| **Краткое изложение теста** | Проверка возможности клиента вводить свои личные данные. |
| **Этапы теста** | 1. Авторизоваться как клиент.  2. Перейти в раздел личных данных.  3. Ввести личные данные (ФИО, email, телефон).  4. Сохранить данные нажатием на кнопку «Изменить». |
| **Тестовые данные** | -- Имя пользователя: client  - Пароль: client123  - Личные данные:  - ФИО: Плотов Иван Анатольевич  - email: plotov@gmail.com  - Телефон: 5554634663 |
| **Ожидаемый результат** | Личные данные успешно сохранены и отображаются в профиле клиента. |
| **Фактический результат** | Личные данные успешно сохранены и отображаются в профиле клиента. |
| **Статус** | Зачет |
| **Предварительное условие** | Клиент должен быть авторизован. |
| **Постусловие** | Личные данные должны быть сохранены в БД. |
| **Примечания/комментарии** | Нет |

Тестовый пример #5:

|  |  |
| --- | --- |
| **Тестовый пример #** | TC\_UI\_5 |
| **Приоритет тестирования** | Высокий |
| **Заголовок/название теста** | Проверка возможности бронирования номера клиентом. |
| **Краткое изложение теста** | Проверка возможности клиента бронировать номер в гостинице |
| **Этапы теста** | 1. Авторизоваться как клиент.  2. Перейти в раздел бронирования.  3. Выбрать даты заезда и выезда.  4. Выбрать номер.  5. Выбрать дополнительные услуги.  5. Подтвердить бронирование. |
| **Тестовые данные** | - Имя пользователя: client  - Пароль: client123  - Даты:  - Заезд: 01.01.2023  - Выезд: 05.01.2023  - Номер: одноместный  - Дополнительные услуги: парковка |
| **Ожидаемый результат** | Бронирование успешно подтверждено и отображается в списке бронирований клиента. |
| **Фактический результат** | Бронирование успешно подтверждено и отображается в списке бронирований клиента. |
| **Статус** | Зачет |
| **Предварительное условие** | Клиент должен быть авторизован. |
| **Постусловие** | Бронирование должно быть сохранено в БД. |
| **Примечания/комментарии** | Нет |

Модульные тесты



Код:

using Microsoft.VisualStudio.TestTools.UnitTesting;

using System;

using System.Data.SqlClient;

namespace Test

{

[TestClass]

public class UnitTest1

{

private const string ConnectionString = "Data Source= ADCLG1; Initial catalog=Фрич\_УП; Integrated Security=True; TrustServerCertificate=true";

int clientID = 0;

int userID;

[TestMethod]

public void UserTestMethod1\_AddNewUser()

{

DataBase dataBase = new DataBase(ConnectionString);

dataBase.OpenConnection();

string comDel = "INSERT INTO Пользователи (Логин, Пароль, Роль) VALUES (@Login, @Password, @Role); SELECT SCOPE\_IDENTITY();";

SqlCommand cmd1 = new SqlCommand(comDel, dataBase.GetConnection());

SqlParameter pr1, pr2, pr3;

pr1 = new SqlParameter("@Login", "user11");

pr2 = new SqlParameter("@Password", "password1");

pr3 = new SqlParameter("@Role", 2);

cmd1.Parameters.Add(pr1);

cmd1.Parameters.Add(pr2);

cmd1.Parameters.Add(pr3);

userID = Convert.ToInt32(cmd1.ExecuteScalar());

comDel = "SELECT Логин FROM Пользователи WHERE [Id Пользователя] = @Id";

cmd1 = new SqlCommand(comDel, dataBase.GetConnection());

pr1 = new SqlParameter("@Id", userID);

cmd1.Parameters.Add(pr1);

string result = cmd1.ExecuteScalar().ToString();

string expected = "user11";

dataBase.CloseConnection();

Assert.AreEqual(expected, result, "Ожидаемый логин не был получен!");

}

[TestMethod]

public void UserTestMethod2\_UpdateUserPassword()

{

DataBase dataBase = new DataBase(ConnectionString);

dataBase.OpenConnection();

string comDel = "UPDATE Пользователи SET Пароль = @Password WHERE Логин= @Логин";

SqlCommand cmd1 = new SqlCommand(comDel, dataBase.GetConnection());

SqlParameter pr1, pr2;

pr1 = new SqlParameter("@Логин", "user11");

pr2 = new SqlParameter("@Password", "password1\_updated");

cmd1.Parameters.Add(pr1);

cmd1.Parameters.Add(pr2);

cmd1.ExecuteNonQuery();

comDel = "SELECT Пароль FROM Пользователи WHERE Логин= @Логин";

cmd1 = new SqlCommand(comDel, dataBase.GetConnection());

pr1 = new SqlParameter("@Логин", "user11");

cmd1.Parameters.Add(pr1);

string result = (cmd1.ExecuteScalar()).ToString();

string expected = "password1\_updated";

dataBase.CloseConnection();

Assert.AreEqual(expected, result, "Ожидаемый пароль не был получен!");

}

[TestMethod]

public void UserTestMethod3\_DeleteUser()

{

DataBase dataBase = new DataBase(ConnectionString);

dataBase.OpenConnection();

string comDel = "DELETE FROM Пользователи WHERE Логин= @Логин";

SqlCommand cmd1 = new SqlCommand(comDel, dataBase.GetConnection());

SqlParameter pr1;

pr1 = new SqlParameter("@Логин", "user11");

cmd1.Parameters.Add(pr1);

cmd1.ExecuteNonQuery();

comDel = "SELECT COUNT(\*) FROM Пользователи WHERE Логин= @Логин";

cmd1 = new SqlCommand(comDel, dataBase.GetConnection());

pr1 = new SqlParameter("@Логин", "user11");

cmd1.Parameters.Add(pr1);

var result = Convert.ToInt32(cmd1.ExecuteScalar());

dataBase.CloseConnection();

Assert.AreEqual(0, result, "Ожидаемый пользователь не был удален!");

}

[TestMethod]

public void UserTestMethod4\_AddUserWithInvalidData()

{

DataBase dataBase = new DataBase(ConnectionString);

dataBase.OpenConnection();

string comDel = "INSERT INTO Пользователи (Логин, Пароль, Роль) VALUES (@Login, @Password, @Role)";

SqlCommand cmd1 = new SqlCommand(comDel, dataBase.GetConnection());

SqlParameter pr1, pr2, pr3;

pr1 = new SqlParameter("@Login", null); // Invalid login

pr2 = new SqlParameter("@Password", "password1");

pr3 = new SqlParameter("@Role", 2);

cmd1.Parameters.Add(pr1);

cmd1.Parameters.Add(pr2);

cmd1.Parameters.Add(pr3);

try

{

cmd1.ExecuteNonQuery();

dataBase.CloseConnection();

Assert.Fail("Ожидаемое исключение не было получено!");

}

catch (SqlException)

{

dataBase.CloseConnection();

}

}

[TestMethod]

public void ClientTestMethod1\_AddNewClient()

{

DataBase dataBase = new DataBase(ConnectionString);

dataBase.OpenConnection();

string comDel = "INSERT INTO Клиенты (ФИО, [Номер телефона], email, [Id пользователя]) VALUES (@Fio, @Phone, @Email, @UserId)";

SqlCommand cmd1 = new SqlCommand(comDel, dataBase.GetConnection());

SqlParameter pr1, pr2, pr3, pr4;

pr1 = new SqlParameter("@Fio", "Иванов Иван Иванович");

pr2 = new SqlParameter("@Phone", "1234567890");

pr3 = new SqlParameter("@Email", "ivanov@example.com");

pr4 = new SqlParameter("@UserId", 6);

cmd1.Parameters.Add(pr1);

cmd1.Parameters.Add(pr2);

cmd1.Parameters.Add(pr3);

cmd1.Parameters.Add(pr4);

cmd1.ExecuteNonQuery();

comDel = "SELECT ФИО FROM Клиенты WHERE [Номер телефона] = @Phone";

cmd1 = new SqlCommand(comDel, dataBase.GetConnection());

pr1 = new SqlParameter("@Phone", "1234567890");

cmd1.Parameters.Add(pr1);

var result = cmd1.ExecuteScalar();

var expected = "Иванов Иван Иванович";

dataBase.CloseConnection();

Assert.AreEqual(expected, result, "Ожидаемое ФИО не было получено!");

}

[TestMethod]

public void ClientTestMethod2\_UpdateClientFIO()

{

DataBase dataBase = new DataBase(ConnectionString);

dataBase.OpenConnection();

string comDel = "UPDATE Клиенты SET ФИО = @Fio WHERE [Номер телефона] = @Phone";

SqlCommand cmd1 = new SqlCommand(comDel, dataBase.GetConnection());

SqlParameter pr1, pr2;

pr1 = new SqlParameter("@Fio", "Иванов Иван Иванович\_updated");

pr2 = new SqlParameter("@Phone", "1234567890");

cmd1.Parameters.Add(pr1);

cmd1.Parameters.Add(pr2);

cmd1.ExecuteNonQuery();

comDel = "SELECT ФИО FROM Клиенты WHERE [Номер телефона] = @Phone";

cmd1 = new SqlCommand(comDel, dataBase.GetConnection());

pr1 = new SqlParameter("@Phone", "1234567890");

cmd1.Parameters.Add(pr1);

var result = cmd1.ExecuteScalar();

string expected = "Иванов Иван Иванович\_updated";

dataBase.CloseConnection();

Assert.AreEqual(expected, result, "Ожидаемое ФИО не было получено!");

}

[TestMethod]

public void ClientTestMethod4\_DeleteClient()

{

DataBase dataBase = new DataBase(ConnectionString);

dataBase.OpenConnection();

string comDel = "DELETE FROM Клиенты WHERE [Номер телефона] = @Phone";

SqlCommand cmd1 = new SqlCommand(comDel, dataBase.GetConnection());

SqlParameter pr1;

pr1 = new SqlParameter("@Phone", "1234567890");

cmd1.Parameters.Add(pr1);

cmd1.ExecuteNonQuery();

comDel = "SELECT ФИО FROM Клиенты WHERE [Номер телефона] = @Phone";

cmd1 = new SqlCommand(comDel, dataBase.GetConnection());

pr1 = new SqlParameter("@Phone", "1234567890");

cmd1.Parameters.Add(pr1);

var result = cmd1.ExecuteScalar();

dataBase.CloseConnection();

Assert.AreEqual(null, result, "Ожидаемый клиент не был удален!");

}

[TestMethod]

public void ClientTestMethod3\_UpdateClientEmail()

{

DataBase dataBase = new DataBase(ConnectionString);

dataBase.OpenConnection();

string comDel = "UPDATE Клиенты SET email = @Email WHERE [Номер телефона] = @Phone";

SqlCommand cmd1 = new SqlCommand(comDel, dataBase.GetConnection());

SqlParameter pr1, pr2;

pr1 = new SqlParameter("@Email", "newemail@example.com");

pr2 = new SqlParameter("@Phone", "1234567890");

cmd1.Parameters.Add(pr1);

cmd1.Parameters.Add(pr2);

cmd1.ExecuteNonQuery();

comDel = "SELECT email FROM Клиенты WHERE [Номер телефона] = @Phone";

cmd1 = new SqlCommand(comDel, dataBase.GetConnection());

pr1 = new SqlParameter("@Phone", "1234567890");

cmd1.Parameters.Add(pr1);

var result = cmd1.ExecuteScalar();

string expected = "newemail@example.com";

dataBase.CloseConnection();

Assert.AreEqual(expected, result, "Ожидаемый email не был получен!");

}

[TestMethod]

public void ServiceTestMethod1\_AddNewService()

{

DataBase dataBase = new DataBase(ConnectionString);

dataBase.OpenConnection();

string comDel = "INSERT INTO Дополнительные\_услуги (Название, Описание, Стоимость) VALUES (@Name, @Description, @Cost)";

SqlCommand cmd1 = new SqlCommand(comDel, dataBase.GetConnection());

SqlParameter pr1, pr2, pr3;

pr1 = new SqlParameter("@Name", "Завтрак");

pr2 = new SqlParameter("@Description", "Континентальный завтрак");

pr3 = new SqlParameter("@Cost", 500.00);

cmd1.Parameters.Add(pr1);

cmd1.Parameters.Add(pr2);

cmd1.Parameters.Add(pr3);

cmd1.ExecuteNonQuery();

comDel = "SELECT Название FROM Дополнительные\_услуги WHERE Описание = @Description";

cmd1 = new SqlCommand(comDel, dataBase.GetConnection());

pr1 = new SqlParameter("@Description", "Континентальный завтрак");

cmd1.Parameters.Add(pr1);

var result = cmd1.ExecuteScalar();

var expected = "Завтрак";

dataBase.CloseConnection();

Assert.AreEqual(expected, result, "Ожидаемое название услуги не было получено!");

}

[TestMethod]

public void ServiceTestMethod2\_DeleteService()

{

DataBase dataBase = new DataBase(ConnectionString);

dataBase.OpenConnection();

string comDel = "DELETE FROM Дополнительные\_услуги WHERE Название = @Name";

SqlCommand cmd1 = new SqlCommand(comDel, dataBase.GetConnection());

SqlParameter pr1;

pr1 = new SqlParameter("@Name", "Завтрак");

cmd1.Parameters.Add(pr1);

cmd1.ExecuteNonQuery();

comDel = "SELECT Название FROM Дополнительные\_услуги WHERE Название = @Name";

cmd1 = new SqlCommand(comDel, dataBase.GetConnection());

pr1 = new SqlParameter("@Name", "Завтрак");

cmd1.Parameters.Add(pr1);

var result = cmd1.ExecuteScalar();

dataBase.CloseConnection();

Assert.AreEqual(null, result, "Ожидаемая услуга не была удалена!");

}

}

}