**1 Тестирование программного продукта**

**1.1 Обоснование методов и средств тестирования**

Разработанный программный продукт имеет довольно широкий функционал и большой объем программного кода, однако, он не выполняет никаких математических вычислений, поэтому выполнить тестирование встроенными в MS Visual Studio 2019 утилитами не представляется возможным. По этой причине модульное тестирование (unit-test) придется выполнять вручную, применив в коде блоки обработки исключений try… catch.

Модульное тестирование предполагает полный доступ к коду, соответственно, будет применен метод белого ящика. Другими словами, тестировщику предоставлен весь код, более того, он должен понимать его логику и обладать знаниями по данному языку программирования.

**1.2 Модульное тестирование**

1.2.1 Тестируемый модуль: форма отображения списка блюд (Form2)

Исходный код:

using System;

using System.Windows.Forms;

namespace Menu

{

public partial class Form2 : Form

{

public Form2()

{

InitializeComponent();

}

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

{

ActiveForm.Hide();

userStartForm frm1 = new userStartForm();

frm1.Show();

}

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

{

Application.Exit();

}

private void Form2\_Load(object sender, EventArgs e)

{

this.FormClosing += MainForm\_Closing;

try

{

this.vegetablesTableAdapter.Fill(this.menuDataSet.vegetables);

this.soupTableAdapter.Fill(this.menuDataSet.soup);

this.pastaTableAdapter.Fill(this.menuDataSet.pasta);

this.snacksTableAdapter.Fill(this.menuDataSet.snacks);

this.meatTableAdapter.Fill(this.menuDataSet.meat);

this.garnishTableAdapter.Fill(this.menuDataSet.garnish);

this.fishTableAdapter.Fill(this.menuDataSet.fish);

this.desserts\_modernTableAdapter.Fill(this.menuDataSet.desserts\_modern);

this.desserts\_classicTableAdapter.Fill(this.menuDataSet.desserts\_classic);

this.breadTableAdapter.Fill(this.menuDataSet.bread);

}

catch (Exception)

{

MessageBox.Show("Ошибка подключения к базе данных. Попробуйте еще раз.");

}

}

private void MainForm\_Closing(object sender, System.ComponentModel.CancelEventArgs e)

//переназначение крестика на форме чтобы он закрывал все формы сразу

//на случай, если в ходе работы приложения было открыто более одной формы

//если не переназначать эту кнопку, то приложение не сможет полностью закрыться, если было открыто более одной формы, т.к. другие формы останутся висеть в процессах

{

Application.Exit();

}

private void dataGridView4\_CellContentClick(object sender, DataGridViewCellEventArgs e)

{

}

}

}

Цель тестирования: проверка работоспособности взаимодействия с базой данных

Условия: открыт файл Form2.cs

Ожидаемый результат: база данных отображается верно и полностью

Результат: база данных отображается верно и в полном объёме:

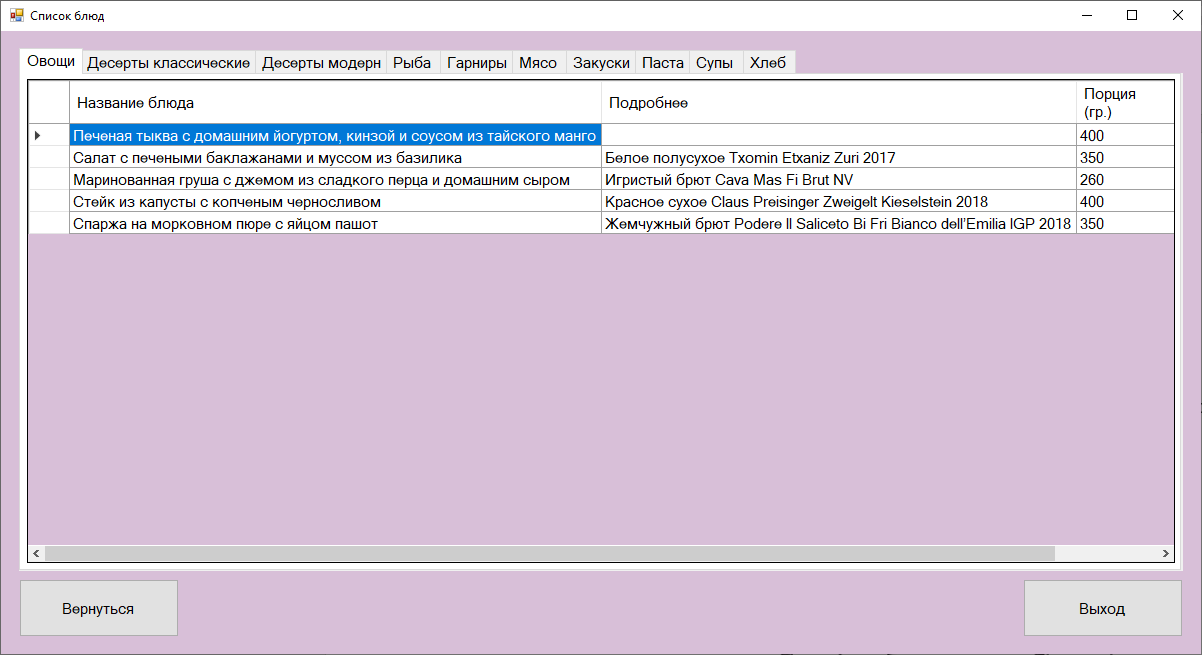


Рисунок 1 – Отображение базы данных в Form2

1.2.2 Тестируемый модуль: Форма отображения винной карты (Form3)

Исходный код:

using System;

using System.Windows.Forms;

namespace Menu

{

public partial class Form3 : Form

{

public Form3()

{

InitializeComponent();

}

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

{

ActiveForm.Hide();

userStartForm frm1 = new userStartForm();

frm1.Show();

}

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

{

Application.Exit();

}

private void MainForm\_Closing(object sender, System.ComponentModel.CancelEventArgs e)

//переназначение крестика на форме чтобы он закрывал все формы сразу

//на случай, если в ходе работы приложения было открыто более одной формы

//если не переназначать эту кнопку, то приложение не сможет полностью закрыться, если было открыто более одной формы, т.к. другие формы останутся висеть в процессах

{

Application.Exit();

}

private void Form3\_Load(object sender, EventArgs e)

{

this.FormClosing += MainForm\_Closing;

try

{

this.whiteTableAdapter.Fill(this.wine\_mapDataSet.white);

this.redTableAdapter.Fill(this.wine\_mapDataSet.red);

this.pinkTableAdapter.Fill(this.wine\_mapDataSet.pink);

this.sparklingTableAdapter.Fill(this.wine\_mapDataSet.sparkling);

}

catch (Exception)

{

MessageBox.Show("Ошибка подключения к базе данных. Попробуйте еще раз.");

}

}

}

}

Цель тестирования: проверка работоспособности взаимодействия с базой данных

Ожидаемый результат: база данных отображается верно и полностью (при)

Результат: база данных отображена верно и в полностью:

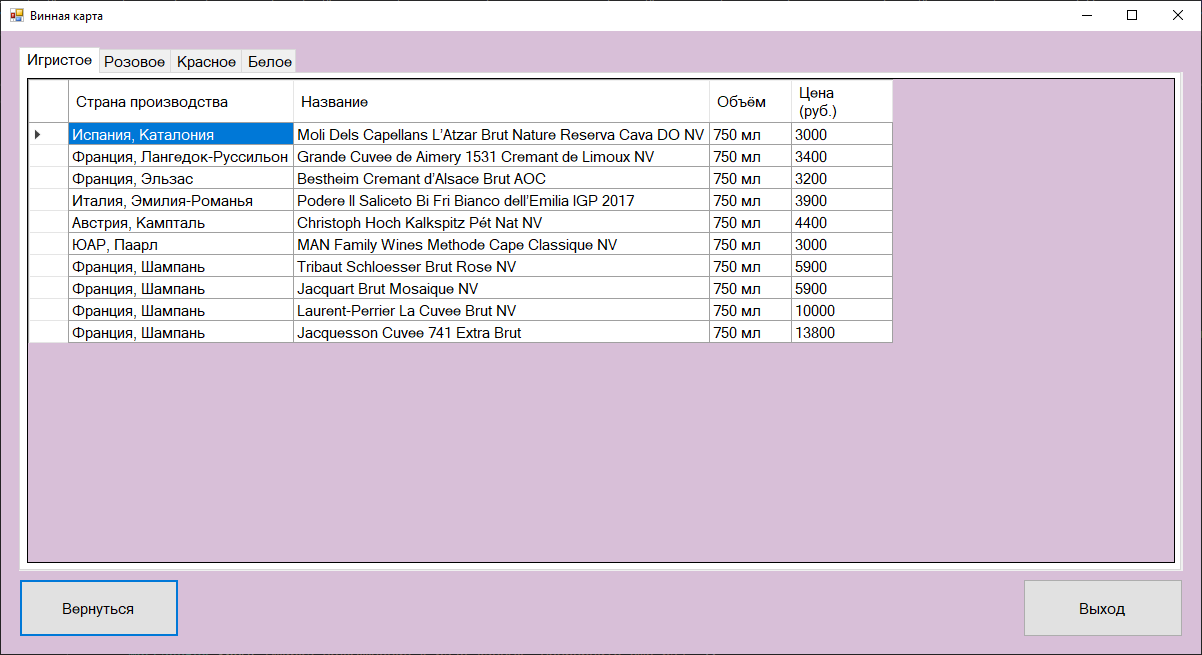


Рисунок 2 – Отображение винной карты в Form3

Так как формы Form2Cook и Form3Cook являются точными копиями уже протестированных форм с незначительными изменениями, они не будут подвергаться тестированию.

1.2.3 Тестируемый модуль: Форма регистрации посетителя ресторана (UserRegistration)

Исходный код:

using System;

using System.Windows.Forms;

using MySql.Data.MySqlClient;

namespace Menu

{

public partial class UserRegistration : Form

{

public UserRegistration()

{

InitializeComponent();

}

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

{

this.FormClosing += MainForm\_Closing;

}

private void MainForm\_Closing(object sender, System.ComponentModel.CancelEventArgs e)

//переназначение крестика на форме чтобы он закрывал все формы сразу

//на случай, если в ходе работы приложения было открыто более одной формы

//если не переназначать эту кнопку, то приложение не сможет полностью закрыться, если было открыто более одной формы, т.к. другие формы останутся висеть в процессах

{

Application.Exit();

}

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

{

try

{

ConnectDatabase db = new ConnectDatabase();

MySqlCommand command = new MySqlCommand("INSERT INTO `visitors` (`id`, `name`, `surname`, `thirdname`, `login`, `password`) VALUES (NULL, @name, @surname, @thirdname, @login, @password)", db.GetConnection());

command.Parameters.Add("@login", MySqlDbType.VarChar).Value = login.Text;

command.Parameters.Add("@password", MySqlDbType.VarChar).Value = password.Text;

command.Parameters.Add("@name", MySqlDbType.VarChar).Value = NName.Text;

command.Parameters.Add("@surname", MySqlDbType.VarChar).Value = Surname.Text;

command.Parameters.Add("@thirdname", MySqlDbType.VarChar).Value = Thirdname.Text;

db.OpenConnection();

if (command.ExecuteNonQuery() == 1)

{

MessageBox.Show("Учетная запись успешно создана!");

}

else

{

MessageBox.Show("Ошибка при создании учетной записи");

}

db.CloseConnection();

ActiveForm.Hide();

userAR user = new userAR();

user.Show();

}

catch (Exception)

{

MessageBox.Show("Ошибка при регистрации");

}

}

}

}

Цель тестирования: проверка работоспособности формы регистрации

Ожидаемый результат: при регистрации введенные данные будут занесены в базу данных

Результат: данные занесены в базу:

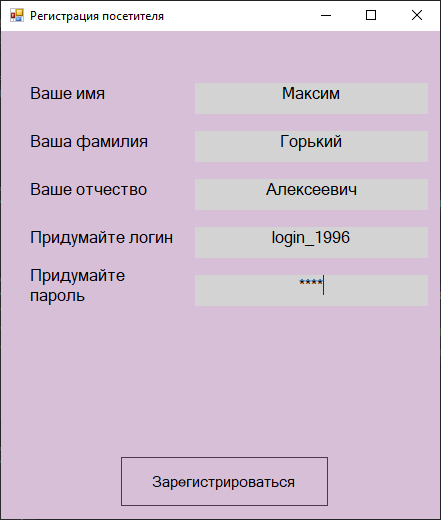


Рисунок 3 – Вводимые данные

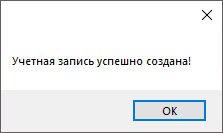


Рисунок 4 – Сообщение об успешной регистрации

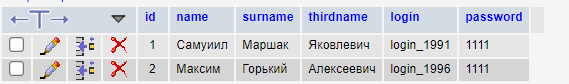
****

Рисунок 5 – Записи в базе данных

1.2.4 Тестируемый модуль: Форма авторизации пользователя (AuthorizationUser)

Исходный код:

using System;

using System.Windows.Forms;

using MySql.Data.MySqlClient;

using System.Data;

namespace Menu

{

public partial class AuthorizationUser : Form

{

public AuthorizationUser()

{

InitializeComponent();

}

static public string role;

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

{

try

{

string Login = log.Text;

string Password = pass.Text;

ConnectDatabase db = new ConnectDatabase();

DataTable table = new DataTable();

MySqlDataAdapter adapter = new MySqlDataAdapter();

MySqlCommand command = new MySqlCommand("SELECT \* FROM `visitors` WHERE `login` = @login AND `password` = @password", db.GetConnection());

command.Parameters.Add("@login", MySqlDbType.VarChar).Value = Login;

command.Parameters.Add("@password", MySqlDbType.VarChar).Value = Password;

adapter.SelectCommand = command;

adapter.Fill(table);

MySqlDataAdapter users = new MySqlDataAdapter($"Select role From visitors", db.GetConnection());

role = Convert.ToString(table.Rows[0][5]);

if (table.Rows.Count > 0)

{

ActiveForm.Hide();

cookStartFormAfterRegistrationOrAuthorization csForm = new cookStartFormAfterRegistrationOrAuthorization();

csForm.Show();

}

else

MessageBox.Show("No");

ActiveForm.Hide();

userAR usstfrm = new userAR();

usstfrm.Show();

}

catch (Exception)

{

MessageBox.Show("Ошибка при авторизации");

}

}

private void MainForm\_Closing(object sender, System.ComponentModel.CancelEventArgs e)

//переназначение крестика на форме чтобы он закрывал все формы сразу

//на случай, если в ходе работы приложения было открыто более одной формы

//если не переназначать эту кнопку, то приложение не сможет полностью закрыться, если было открыто более одной формы, т.к. другие формы останутся висеть в процессах

{

Application.Exit();

}

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

{

this.FormClosing += MainForm\_Closing;

}

}

}

Цель тестирования: проверка работоспособности формы авторизации ()

Ожидаемый результат: введенные данные сверяются с уже добавленными в базу и, если есть совпадение, то пользователь попадает на следующую форму. При вводе данных, которых нет в базе, выводится сообщение, что аккаунта не существует.

Результат: при вводе данных, которые уже есть в базе, пользователь попадает на следующую форму:

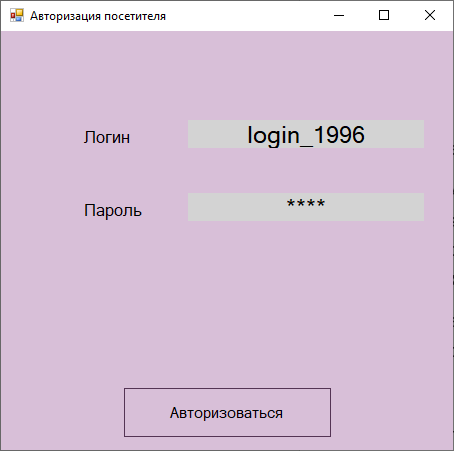


Рисунок 6 – Данные для авторизации

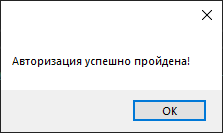


Рисунок 7 – Сообщение об успешной авторизации

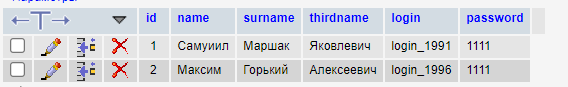


Рисунок 8 – Записи в базе данных

1.2.4 Тестируемый модуль: Форма регистрации работника кухни (cookRegistration)

Исходный код:

using System;

using System.Windows.Forms;

using MySql.Data.MySqlClient;

namespace Menu

{

public partial class cookRegistration : Form

{

public cookRegistration()

{

InitializeComponent();

}

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

{

try

{

ConnectDatabase db = new ConnectDatabase();

MySqlCommand command = new MySqlCommand("INSERT INTO `cooks` (`id`, `name`, `surname`, `thirdname`, `login`, `password`) VALUES (NULL, @name, @surname, @thirdname, @login, @password)", db.GetConnection());

command.Parameters.Add("@login", MySqlDbType.VarChar).Value = login.Text;

command.Parameters.Add("@password", MySqlDbType.VarChar).Value = password.Text;

command.Parameters.Add("@name", MySqlDbType.VarChar).Value = userName.Text;

command.Parameters.Add("@surname", MySqlDbType.VarChar).Value = userSurname.Text;

command.Parameters.Add("@thirdname", MySqlDbType.VarChar).Value = userThirdname.Text;

db.OpenConnection();

if (command.ExecuteNonQuery() == 1)

{

MessageBox.Show("Учетная запись успешно создана!");

}

else

{

MessageBox.Show("Ошибка при создании учетной записи");

}

db.CloseConnection();

ActiveForm.Hide();

cookStartFormAfterRegistrationOrAuthorization form = new cookStartFormAfterRegistrationOrAuthorization();

form.Show();

}

catch (Exception)

{

MessageBox.Show("Обшибка при регистрации");

}

}

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

{

this.FormClosing += MainForm\_Closing;

}

private void MainForm\_Closing(object sender, System.ComponentModel.CancelEventArgs e)

//переназначение крестика на форме чтобы он закрывал все формы сразу

//на случай, если в ходе работы приложения было открыто более одной формы

//если не переназначать эту кнопку, то приложение не сможет полностью закрыться, если было открыто более одной формы, т.к. другие формы останутся висеть в процессах

{

Application.Exit();

}

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

{

}

}

}

Цель тестирования: проверка работоспособности формы регистрации

Ожидаемый результат: введенные данные будут отправлены в базу данных

Результат: данные занесены в базу, выведено сообщение об успешной регистрации, появилась форма для работника кухни:

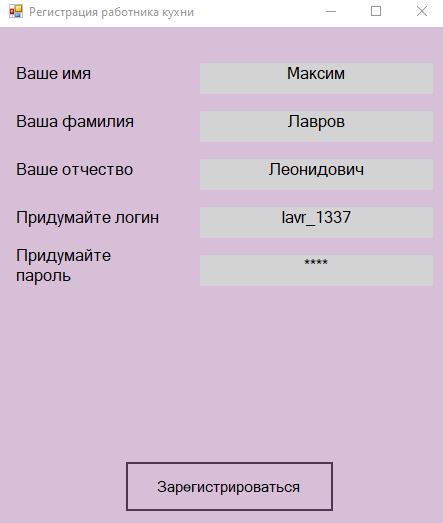


Рисунок 9 – Вводимые данные при регистрации

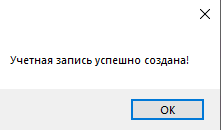


Рисунок 10 – Сообщение об успешной регистрации

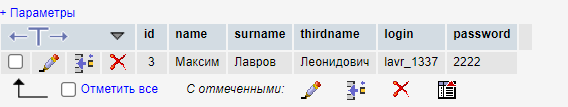


Рисунок 11 – Запись в базе данных

1.2.4 Тестируемый модуль: Форма авторизации работника кухни (cookAuthorization)

Исходный код:

using System;

using System.Windows.Forms;

using MySql.Data.MySqlClient;

using System.Data;

namespace Menu

{

public partial class cookAuthorization : Form

{

public cookAuthorization()

{

InitializeComponent();

}

static public string role;

public void button3\_Click(object sender, EventArgs e)//авторизация работника кухни

{

try

{

string Login = login.Text;

string Password = password.Text;

ConnectDatabase db = new ConnectDatabase();

DataTable table = new DataTable();

MySqlDataAdapter adapter = new MySqlDataAdapter();

MySqlCommand command = new MySqlCommand("SELECT \* FROM `cooks` WHERE `login` = @login AND `password` = @password", db.GetConnection());

command.Parameters.Add("@login", MySqlDbType.VarChar).Value = Login;

command.Parameters.Add("@password", MySqlDbType.VarChar).Value = Password;

adapter.SelectCommand = command;

adapter.Fill(table);

MySqlDataAdapter users = new MySqlDataAdapter($"Select role From cooks", db.GetConnection());

role = Convert.ToString(table.Rows[0][5]);

if (table.Rows.Count > 0)

{

ActiveForm.Hide();

cookStartFormAfterRegistrationOrAuthorization csForm = new cookStartFormAfterRegistrationOrAuthorization();

csForm.Show();

}

else

MessageBox.Show("No");

}

catch (Exception)

{

MessageBox.Show("Ошибка при авторизации");

}

}

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

{

this.FormClosing += MainForm\_Closing;

}

private void MainForm\_Closing(object sender, System.ComponentModel.CancelEventArgs e)

//переназначение крестика на форме чтобы он закрывал все формы сразу

//на случай, если в ходе работы приложения было открыто более одной формы

//если не переназначать эту кнопку, то приложение не сможет полностью закрыться, если было открыто более одной формы, т.к. другие формы останутся висеть в процессах

{

Application.Exit();

}

}

}

Цель тестирования: проверка работоспособности формы авторизации

Ожидаемый результат: если введенные данные уже есть в базе, то будет выведено сообщение об успешной авторизации и пользователь попадет на форму для зарегистрированных работников кухни, если же данные не совпадут, будет выведено соответствующее сообщение.

Результат: вывод сообщения об успешной авторизации, открыта форма для работников кухни:

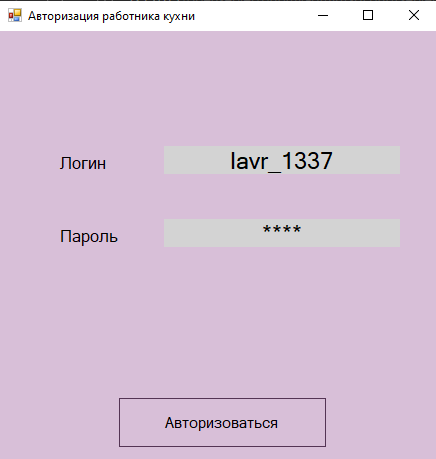


Рисунок 12 – Данные для авторизации

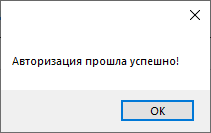


Рисунок 13 – Сообщение об успешной авторизации

1.2.5 Тестируемый модуль: Форма редактирования меню (menuEdit)

Исходный код:

using System;

using System.Windows.Forms;

namespace Menu

{

public partial class menuEditor : Form

{

public menuEditor()

{

InitializeComponent();

}

private void MainForm\_Closing(object sender, System.ComponentModel.CancelEventArgs e)

//переназначение крестика на форме чтобы он закрывал все формы сразу

//на случай, если в ходе работы приложения было открыто более одной формы

//если не переназначать эту кнопку, то приложение не сможет полностью закрыться, если было открыто более одной формы, т.к. другие формы останутся висеть в процессах

{

Application.Exit();

}

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

{

try

{

this.whiteTableAdapter.Fill(this.wine\_mapDataSet.white);

this.redTableAdapter.Fill(this.wine\_mapDataSet.red);

this.pinkTableAdapter.Fill(this.wine\_mapDataSet.pink);

this.sparklingTableAdapter.Fill(this.wine\_mapDataSet.sparkling);

this.breadTableAdapter.Fill(this.menuDataSet.bread);

this.soupTableAdapter.Fill(this.menuDataSet.soup);

this.pastaTableAdapter.Fill(this.menuDataSet.pasta);

this.snacksTableAdapter.Fill(this.menuDataSet.snacks);

this.meatTableAdapter.Fill(this.menuDataSet.meat);

this.garnishTableAdapter.Fill(this.menuDataSet.garnish);

this.fishTableAdapter.Fill(this.menuDataSet.fish);

this.desserts\_modernTableAdapter.Fill(this.menuDataSet.desserts\_modern);

this.desserts\_classicTableAdapter.Fill(this.menuDataSet.desserts\_classic);

this.vegetablesTableAdapter.Fill(this.menuDataSet.vegetables);

}

catch (Exception)

{

MessageBox.Show("Ошибка соединения с базой данных");

}

this.FormClosing += MainForm\_Closing;

}

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

{

try

{

this.vegetablesTableAdapter.Update(this.menuDataSet.vegetables);

this.soupTableAdapter.Update(this.menuDataSet.soup);

this.pastaTableAdapter.Update(this.menuDataSet.pasta);

this.snacksTableAdapter.Update(this.menuDataSet.snacks);

this.meatTableAdapter.Update(this.menuDataSet.meat);

this.garnishTableAdapter.Update(this.menuDataSet.garnish);

this.fishTableAdapter.Update(this.menuDataSet.fish);

this.desserts\_modernTableAdapter.Update(this.menuDataSet.desserts\_modern);

this.desserts\_classicTableAdapter.Update(this.menuDataSet.desserts\_classic);

this.breadTableAdapter.Update(this.menuDataSet.bread);

}

catch (Exception)

{

MessageBox.Show("Ошибка соединения с базой данных");

}

}

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

{

try

{

int delet = dataGridView1.SelectedCells[0].RowIndex;

dataGridView1.Rows.RemoveAt(delet);

}

catch (Exception)

{

MessageBox.Show("Ошибка подключения к базе данных");

}

}

private void dataGridView7\_CellContentClick(object sender, DataGridViewCellEventArgs e)

{

}

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

{

try

{

int delet = dataGridView11.SelectedCells[0].RowIndex;

dataGridView11.Rows.RemoveAt(delet);

}

catch(Exception)

{

MessageBox.Show("Ошибка подключения к базе данных");

}

}

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

{

try

{

this.whiteTableAdapter.Update(this.wine\_mapDataSet.white);

this.redTableAdapter.Update(this.wine\_mapDataSet.red);

this.pinkTableAdapter.Update(this.wine\_mapDataSet.pink);

this.sparklingTableAdapter.Update(this.wine\_mapDataSet.sparkling);

}

catch (Exception)

{

MessageBox.Show("Ошибка подключения к базе данных");

}

}

private void tabPage3\_Click(object sender, EventArgs e)

{

}

private void button6\_Click(object sender, EventArgs e)

{

try

{

this.vegetablesTableAdapter.Update(this.menuDataSet.vegetables);

this.soupTableAdapter.Update(this.menuDataSet.soup);

this.pastaTableAdapter.Update(this.menuDataSet.pasta);

this.snacksTableAdapter.Update(this.menuDataSet.snacks);

this.meatTableAdapter.Update(this.menuDataSet.meat);

this.garnishTableAdapter.Update(this.menuDataSet.garnish);

this.fishTableAdapter.Update(this.menuDataSet.fish);

this.desserts\_modernTableAdapter.Update(this.menuDataSet.desserts\_modern);

this.desserts\_classicTableAdapter.Update(this.menuDataSet.desserts\_classic);

this.breadTableAdapter.Update(this.menuDataSet.bread);

}

catch (Exception)

{

MessageBox.Show("Ошибка соединения с базой данных");

}

}

private void button8\_Click(object sender, EventArgs e)

{

try

{

this.vegetablesTableAdapter.Update(this.menuDataSet.vegetables);

this.soupTableAdapter.Update(this.menuDataSet.soup);

this.pastaTableAdapter.Update(this.menuDataSet.pasta);

this.snacksTableAdapter.Update(this.menuDataSet.snacks);

this.meatTableAdapter.Update(this.menuDataSet.meat);

this.garnishTableAdapter.Update(this.menuDataSet.garnish);

this.fishTableAdapter.Update(this.menuDataSet.fish);

this.desserts\_modernTableAdapter.Update(this.menuDataSet.desserts\_modern);

this.desserts\_classicTableAdapter.Update(this.menuDataSet.desserts\_classic);

this.breadTableAdapter.Update(this.menuDataSet.bread);

}

catch (Exception)

{

MessageBox.Show("Ошибка соединения с базой данных");

}

}

private void button10\_Click(object sender, EventArgs e)

{

try

{

this.vegetablesTableAdapter.Update(this.menuDataSet.vegetables);

this.soupTableAdapter.Update(this.menuDataSet.soup);

this.pastaTableAdapter.Update(this.menuDataSet.pasta);

this.snacksTableAdapter.Update(this.menuDataSet.snacks);

this.meatTableAdapter.Update(this.menuDataSet.meat);

this.garnishTableAdapter.Update(this.menuDataSet.garnish);

this.fishTableAdapter.Update(this.menuDataSet.fish);

this.desserts\_modernTableAdapter.Update(this.menuDataSet.desserts\_modern);

this.desserts\_classicTableAdapter.Update(this.menuDataSet.desserts\_classic);

this.breadTableAdapter.Update(this.menuDataSet.bread);

}

catch (Exception)

{

MessageBox.Show("Ошибка соединения с базой данных");

}

}

private void button12\_Click(object sender, EventArgs e)

{

try

{

this.vegetablesTableAdapter.Update(this.menuDataSet.vegetables);

this.soupTableAdapter.Update(this.menuDataSet.soup);

this.pastaTableAdapter.Update(this.menuDataSet.pasta);

this.snacksTableAdapter.Update(this.menuDataSet.snacks);

this.meatTableAdapter.Update(this.menuDataSet.meat);

this.garnishTableAdapter.Update(this.menuDataSet.garnish);

this.fishTableAdapter.Update(this.menuDataSet.fish);

this.desserts\_modernTableAdapter.Update(this.menuDataSet.desserts\_modern);

this.desserts\_classicTableAdapter.Update(this.menuDataSet.desserts\_classic);

this.breadTableAdapter.Update(this.menuDataSet.bread);

}

catch (Exception)

{

MessageBox.Show("Ошибка соединения с базой данных");

}

}

private void button14\_Click(object sender, EventArgs e)

{

try

{

this.vegetablesTableAdapter.Update(this.menuDataSet.vegetables);

this.soupTableAdapter.Update(this.menuDataSet.soup);

this.pastaTableAdapter.Update(this.menuDataSet.pasta);

this.snacksTableAdapter.Update(this.menuDataSet.snacks);

this.meatTableAdapter.Update(this.menuDataSet.meat);

this.garnishTableAdapter.Update(this.menuDataSet.garnish);

this.fishTableAdapter.Update(this.menuDataSet.fish);

this.desserts\_modernTableAdapter.Update(this.menuDataSet.desserts\_modern);

this.desserts\_classicTableAdapter.Update(this.menuDataSet.desserts\_classic);

this.breadTableAdapter.Update(this.menuDataSet.bread);

}

catch (Exception)

{

MessageBox.Show("Ошибка соединения с базой данных");

}

}

private void button16\_Click(object sender, EventArgs e)

{

try

{

this.vegetablesTableAdapter.Update(this.menuDataSet.vegetables);

this.soupTableAdapter.Update(this.menuDataSet.soup);

this.pastaTableAdapter.Update(this.menuDataSet.pasta);

this.snacksTableAdapter.Update(this.menuDataSet.snacks);

this.meatTableAdapter.Update(this.menuDataSet.meat);

this.garnishTableAdapter.Update(this.menuDataSet.garnish);

this.fishTableAdapter.Update(this.menuDataSet.fish);

this.desserts\_modernTableAdapter.Update(this.menuDataSet.desserts\_modern);

this.desserts\_classicTableAdapter.Update(this.menuDataSet.desserts\_classic);

this.breadTableAdapter.Update(this.menuDataSet.bread);

}

catch (Exception)

{

MessageBox.Show("Ошибка соединения с базой данных");

}

}

private void button18\_Click(object sender, EventArgs e)

{

try

{

this.vegetablesTableAdapter.Update(this.menuDataSet.vegetables);

this.soupTableAdapter.Update(this.menuDataSet.soup);

this.pastaTableAdapter.Update(this.menuDataSet.pasta);

this.snacksTableAdapter.Update(this.menuDataSet.snacks);

this.meatTableAdapter.Update(this.menuDataSet.meat);

this.garnishTableAdapter.Update(this.menuDataSet.garnish);

this.fishTableAdapter.Update(this.menuDataSet.fish);

this.desserts\_modernTableAdapter.Update(this.menuDataSet.desserts\_modern);

this.desserts\_classicTableAdapter.Update(this.menuDataSet.desserts\_classic);

this.breadTableAdapter.Update(this.menuDataSet.bread);

}

catch (Exception)

{

MessageBox.Show("Ошибка соединения с базой данных");

}

}

private void button20\_Click(object sender, EventArgs e)

{

try

{

this.vegetablesTableAdapter.Update(this.menuDataSet.vegetables);

this.soupTableAdapter.Update(this.menuDataSet.soup);

this.pastaTableAdapter.Update(this.menuDataSet.pasta);

this.snacksTableAdapter.Update(this.menuDataSet.snacks);

this.meatTableAdapter.Update(this.menuDataSet.meat);

this.garnishTableAdapter.Update(this.menuDataSet.garnish);

this.fishTableAdapter.Update(this.menuDataSet.fish);

this.desserts\_modernTableAdapter.Update(this.menuDataSet.desserts\_modern);

this.desserts\_classicTableAdapter.Update(this.menuDataSet.desserts\_classic);

this.breadTableAdapter.Update(this.menuDataSet.bread);

}

catch (Exception)

{

MessageBox.Show("Ошибка соединения с базой данных");

}

}

private void button22\_Click(object sender, EventArgs e)

{

try

{

this.vegetablesTableAdapter.Update(this.menuDataSet.vegetables);

this.soupTableAdapter.Update(this.menuDataSet.soup);

this.pastaTableAdapter.Update(this.menuDataSet.pasta);

this.snacksTableAdapter.Update(this.menuDataSet.snacks);

this.meatTableAdapter.Update(this.menuDataSet.meat);

this.garnishTableAdapter.Update(this.menuDataSet.garnish);

this.fishTableAdapter.Update(this.menuDataSet.fish);

this.desserts\_modernTableAdapter.Update(this.menuDataSet.desserts\_modern);

this.desserts\_classicTableAdapter.Update(this.menuDataSet.desserts\_classic);

this.breadTableAdapter.Update(this.menuDataSet.bread);

}

catch (Exception)

{

MessageBox.Show("Ошибка соединения с базой данных");

}

}

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

{

try

{

int delet1 = dataGridView2.SelectedCells[0].RowIndex;

dataGridView2.Rows.RemoveAt(delet1);

}

catch (Exception)

{

MessageBox.Show("Ошибка подключения к базе данных");

}

}

private void button7\_Click(object sender, EventArgs e)

{

try

{

int delet2 = dataGridView3.SelectedCells[0].RowIndex;

dataGridView3.Rows.RemoveAt(delet2);

}

catch (Exception)

{

MessageBox.Show("Ошибка подключения к базе данных");

}

}

private void button9\_Click(object sender, EventArgs e)

{

try

{

int delet3 = dataGridView4.SelectedCells[0].RowIndex;

dataGridView4.Rows.RemoveAt(delet3);

}

catch (Exception)

{

MessageBox.Show("Ошибка подключения к базе данных");

}

}

private void button11\_Click(object sender, EventArgs e)

{

try

{

int delet4 = dataGridView5.SelectedCells[0].RowIndex;

dataGridView5.Rows.RemoveAt(delet4);

}

catch (Exception)

{

MessageBox.Show("Ошибка подключения к базе данных");

}

}

private void button13\_Click(object sender, EventArgs e)

{

try

{

int delet5 = dataGridView6.SelectedCells[0].RowIndex;

dataGridView6.Rows.RemoveAt(delet5);

}

catch (Exception)

{

MessageBox.Show("Ошибка подключения к базе данных");

}

}

private void tabPage9\_Click(object sender, EventArgs e)

{

}

private void button15\_Click(object sender, EventArgs e)

{

try

{

int delet6 = dataGridView7.SelectedCells[0].RowIndex;

dataGridView7.Rows.RemoveAt(delet6);

}

catch (Exception)

{

MessageBox.Show("Ошибка подключения к базе данных");

}

}

private void button17\_Click(object sender, EventArgs e)

{

try

{

int delet = dataGridView8.SelectedCells[0].RowIndex;

dataGridView8.Rows.RemoveAt(delet);

}

catch (Exception)

{

MessageBox.Show("Ошибка подключения к базе данных");

}

}

private void button19\_Click(object sender, EventArgs e)

{

try

{

int delet = dataGridView9.SelectedCells[0].RowIndex;

dataGridView9.Rows.RemoveAt(delet);

}

catch (Exception)

{

MessageBox.Show("Ошибка подключения к базе данных");

}

}

private void button21\_Click(object sender, EventArgs e)

{

try

{

int delet = dataGridView10.SelectedCells[0].RowIndex;

dataGridView10.Rows.RemoveAt(delet);

}

catch (Exception)

{

MessageBox.Show("Ошибка подключения к базе данных");

}

}

private void button24\_Click(object sender, EventArgs e)

{

try

{

this.whiteTableAdapter.Update(this.wine\_mapDataSet.white);

this.redTableAdapter.Update(this.wine\_mapDataSet.red);

this.pinkTableAdapter.Update(this.wine\_mapDataSet.pink);

this.sparklingTableAdapter.Update(this.wine\_mapDataSet.sparkling);

}

catch (Exception)

{

MessageBox.Show("Ошибка подключения к базе данных");

}

}

private void button26\_Click(object sender, EventArgs e)

{

try

{

this.whiteTableAdapter.Update(this.wine\_mapDataSet.white);

this.redTableAdapter.Update(this.wine\_mapDataSet.red);

this.pinkTableAdapter.Update(this.wine\_mapDataSet.pink);

this.sparklingTableAdapter.Update(this.wine\_mapDataSet.sparkling);

}

catch (Exception)

{

MessageBox.Show("Ошибка подключения к базе данных");

}

}

private void button28\_Click(object sender, EventArgs e)

{

try

{

this.whiteTableAdapter.Update(this.wine\_mapDataSet.white);

this.redTableAdapter.Update(this.wine\_mapDataSet.red);

this.pinkTableAdapter.Update(this.wine\_mapDataSet.pink);

this.sparklingTableAdapter.Update(this.wine\_mapDataSet.sparkling);

}

catch (Exception)

{

MessageBox.Show("Ошибка подключения к базе данных");

}

}

private void button23\_Click(object sender, EventArgs e)

{

try

{

int delet = dataGridView12.SelectedCells[0].RowIndex;

dataGridView12.Rows.RemoveAt(delet);

}

catch (Exception)

{

MessageBox.Show("Ошибка подключения к базе данных");

}

}

private void button25\_Click(object sender, EventArgs e)

{

try

{

int delet = dataGridView13.SelectedCells[0].RowIndex;

dataGridView13.Rows.RemoveAt(delet);

}

catch (Exception)

{

MessageBox.Show("Ошибка подключения к базе данных");

}

}

private void button27\_Click(object sender, EventArgs e)

{

try

{

int delet = dataGridView14.SelectedCells[0].RowIndex;

dataGridView14.Rows.RemoveAt(delet);

}

catch (Exception)

{

MessageBox.Show("Ошибка подключения к базе данных");

}

}

}

}

Цель тестирования: проверить работоспособность формы для редактирования меню, проверить работоспособность добавления данных в базу, проверить работоспособность удаления данных из базы

Ожидаемый результат: при вводе данных и сохранении, данные должны быть добавлены в базу и отображаться в ней; при удалении строки, данные должны исчезать из базы

Результат: при добавлении данных в таблицу и сохранении, данные отправляются в базу данных и отображаются в ней; при удалении строки и сохранении результата данные удаляются из базы:

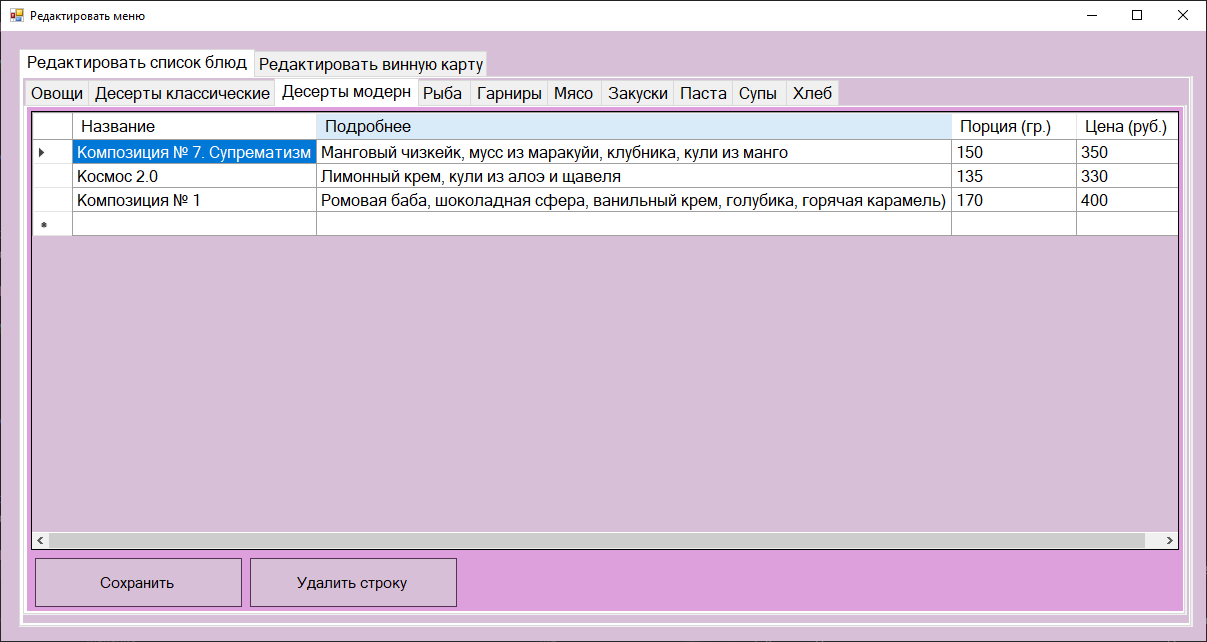


Рисунок 14 – Имеющиеся данные

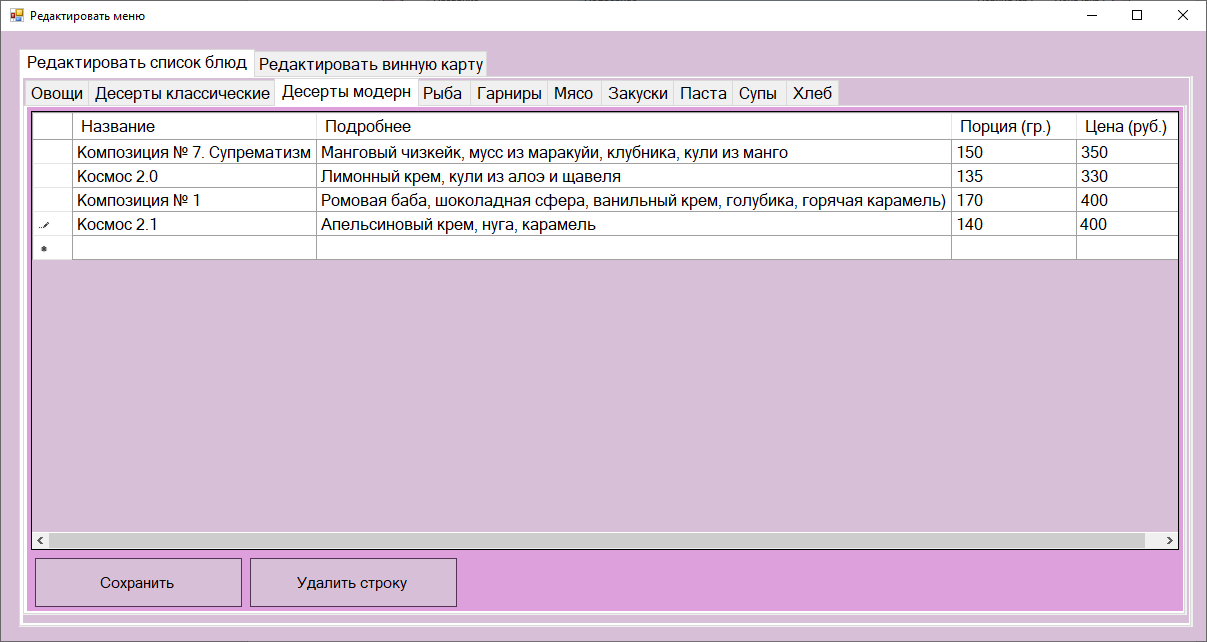


Рисунок 15 – Добавление данных

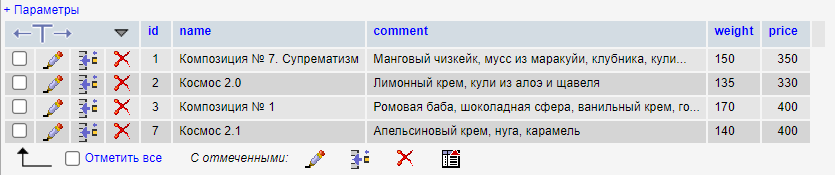


Рисунок 16 – Запись добавлена в базу

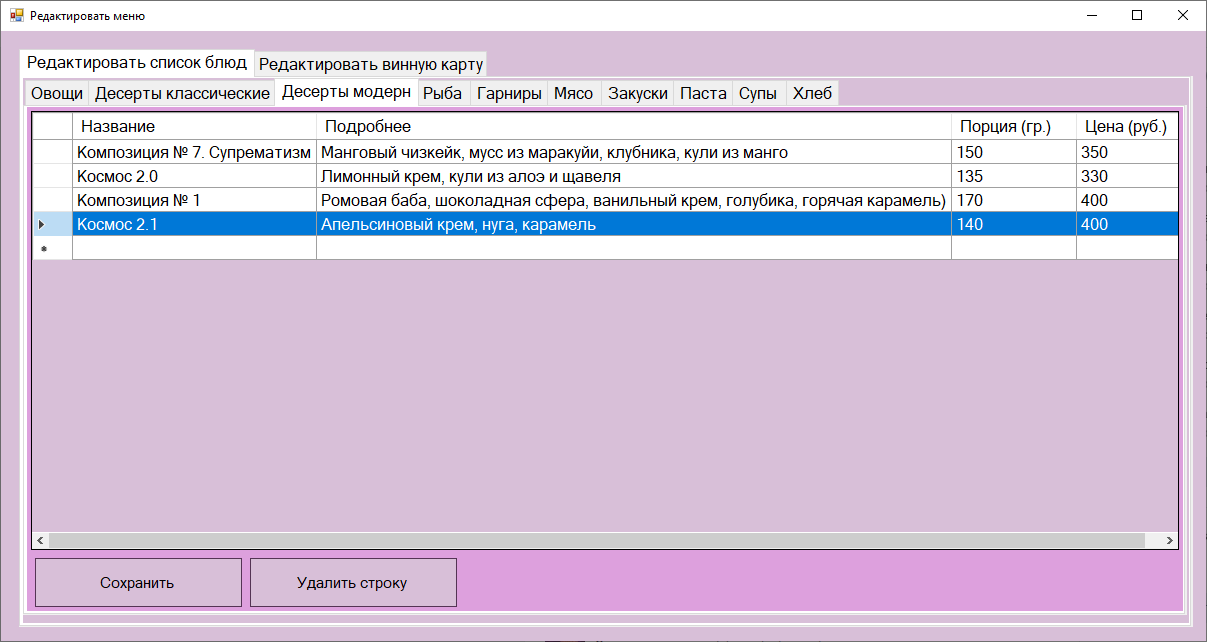


Рисунок 17 – Удаление строки

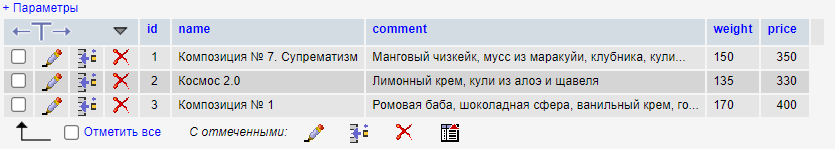


Рисунок 18 – Запись удалена из базы

1.2.6 Тестируемый модуль: Форма бронирования мест в ресторане (reservation)

Исходный код:

using System;

using System.Windows.Forms;

using MySql.Data.MySqlClient;

namespace Menu

{

public partial class reservation : Form

{

public reservation()

{

InitializeComponent();

}

private string connect = "server=127.0.0.1;userid=root;password=root;database=menu\_users;port=3306";

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

{

this.FormClosing += MainForm\_Closing;

}

private void MainForm\_Closing(object sender, System.ComponentModel.CancelEventArgs e)

//переназначение крестика на форме чтобы он закрывал все формы сразу

//на случай, если в ходе работы приложения было открыто более одной формы

//если не переназначать эту кнопку, то приложение не сможет полностью закрыться, если было открыто более одной формы, т.к. другие формы останутся висеть в процессах

{

Application.Exit();

}

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

{

try

{

MySqlConnection connection = new MySqlConnection(connect);

connection.Open();

MySqlCommand command = new MySqlCommand($"INSERT INTO `reserve` (`id`, `place`, `place\_number`, `date`, `time`, `name`) VALUES (NULL, '{place.Text}', '{place\_num.Text}', '{date.Text}', '{time.Text}', '{name.Text}');")

{

Connection = connection

};

command.ExecuteNonQuery();

MessageBox.Show("Место успешно забронировано!");

}

catch (Exception)

{

MessageBox.Show("");

}

}

}

}

Цель тестирования: проверка работоспособности формы бронирования мест в ресторане

Ожидаемый результат: данные, введенные пользователем, будут добавлены в базу; при потере соединения с базой данных, вывод соответствующего сообщения

Результат: данные добавлены в базу:

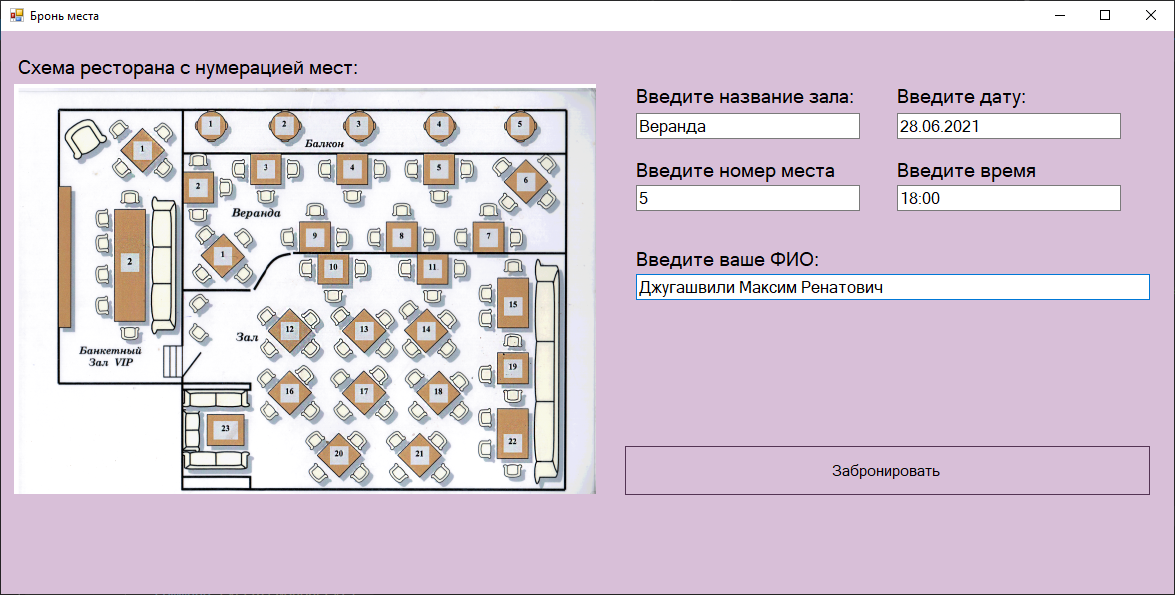


Рисунок 19 – Вводимые данные

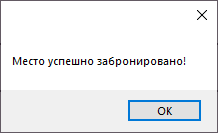


Рисунок 20 – Сообщение об успешном бронировании

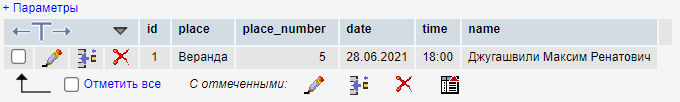


Рисунок 21 – Запись в базе

1.2.7 Тестируемый модуль: Форма заказа блюд (orders)

Исходный код:

using System;

using System.Windows.Forms;

using MySql.Data.MySqlClient;

namespace Menu

{

public partial class orders : Form

{

public orders()

{

InitializeComponent();

}

private string connect = "server=127.0.0.1;userid=root;password=root;database=menu\_users;port=3306";

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

{

this.FormClosing += MainForm\_Closing;

}

private void MainForm\_Closing(object sender, System.ComponentModel.CancelEventArgs e)

//переназначение крестика на форме чтобы он закрывал все формы сразу

//на случай, если в ходе работы приложения было открыто более одной формы

//если не переназначать эту кнопку, то приложение не сможет полностью закрыться, если было открыто более одной формы, т.к. другие формы останутся висеть в процессах

{

Application.Exit();

}

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

{

try

{

MySqlConnection connection = new MySqlConnection(connect);

connection.Open();//соединение с бд

MySqlCommand command = new MySqlCommand($"INSERT INTO `orders` (`id`, `name\_user`, `dish`, `drink`, `place`) VALUES (NULL, '{name.Text}', '{dish.Text}', '{drink.Text}', '{place\_num.Text}');")

{

Connection = connection//команда для заполнения

};

command.ExecuteNonQuery();//выполнение команды

MessageBox.Show("Заказ сделан! Время готовки - от 15 до 30 минут");

}

catch (Exception)

{

MessageBox.Show("Ошибка при формировании заказа");

}

}

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

{

ComboBox dish = (ComboBox)sender;

}

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

{

ComboBox drink = (ComboBox)sender;

}

}

}

Цель тестирования: проверка работоспособности формы для заказа блюд

Ожидаемый результат: введенные данные должны быть добавлены в базу данных; при потере соединения с базой данных должно быть выведено соответствующее сообщение

Результат: данные добавлены в базу:

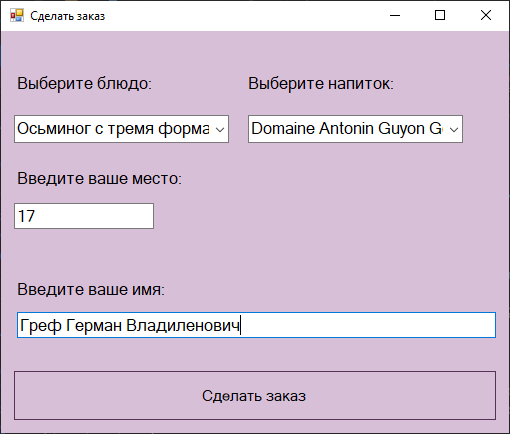


Рисунок 22 – Вводимые данные

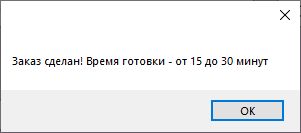


Рисунок 23 – Сообщение об успехе

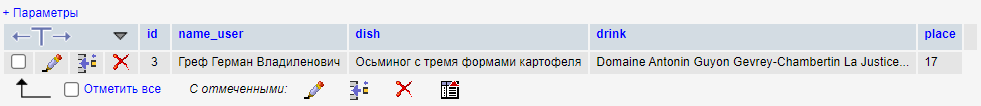


Рисунок 24 – Данные занесены в базу

**2 Результаты тестирования**

Система прошла все этапы тестирования и успешно показала свою работоспособность. Багов и недочетов обнаружено не было.

Модульное тестирование производилось вручную тестировщиком с применением блоков обработки исключений в коде. Подобные блоки были добавлены во многие модули проекта, что поможет в будущем избежать критических ситуаций в работе программы.

Из этого следует, что программный продукт полностью готов к работе в реальных условиях.