**ĐẠI HỌC LẠC HỒNG**

**KHOA CÔNG NGHỆ THÔNG TIN**



**ĐỒ ÁN CUỐI KỲ MÔN C#**

Đề tài:

QUẢN LÝ NHÀ HÀNG

Giảng viên:

**LÊ MẠNH HÙNG**

Sinh viên thực hiện:

**LÊ VĂN TUẤN 121001332**

**TRẦN HỮU LỘC 121001198**

**NGUYỄN THIỆN NHÂN 121001126**

**NGUYỄN MINH QUÂN 121001113**

**LÊ TẤN PHÁT 121001491**

**TP. Biên Hoà, tháng 10 năm 2023**

**LỜI CẢM ƠN**

Trong khoảng thời gian thực hiện đề tài này, chúng em đã nhận được sự giúp đỡ, chỉ bảo tận tình của thầy. Chúng em xin gửi lời cảm ơn sâu sắc đến thầy. Đặc biệt xin chân thành cảm ơn thầy **Lê Mạnh Hùng** là người hướng dẫn, cung cấp cho chúng em những tài liệu và tạo điều kiện thuận lợi giúp đỡ chúng em để hoàn thành đề tài một cách tốt nhất.

Do thời gian và kiến thức có hạn nên mặc dù đã cố gắng hết khả năng nhưng đề tài không thể nào tránh khỏi những thiếu sót. Rất mong được sự đóng góp của thầy để đề tài này hoàn chỉnh hơn.

Nhóm sinh viên thực hiện

**Lê Văn Tuấn**

**Trần Hữu Lộc**

**Nguyễn Thiện Nhân  
 Nguyễn Minh Quân  
 Lê Tấn Phát**

**NHẬN XÉT**

(Của giảng viên hướng dẫn)

**CHƯƠNG 1: GIỚI THIỆU ĐỀ TÀI**

# Giới thiệu đề tài:

Hiện nay, dịch vụ nhà hàng đang được phát triển quy mô lớn. Số lượng nhà hàng đang ngày càng tăng, dẫn đến các dịch vụ cho nhà hàng không thể đáp ứng được những nhu cầu trước đây. Nhà hàng của bạn đang kinh doanh phát đạt, bạn phải thuê thêm nhân viên để quản lý nhà hàng, nhưng điều đó vẫn chưa làm bạn hài lòng về việc quản lý. Nó làm bạn mất nhiều thời giờ hơn, nhân viên của bạn phải ghi chép sổ sách rất nhiều, đôi khi còn bị nhầm lẫn, chậm trễ trong quá trình liên lạc giữa các bộ phận nghiệp vụ trong nhà hàng, cùng với việc tra cứu thông tin chậm trễ hoặc không chính xác có thể làm các bạn bỏ lỡ các cơ hội cho việc tiếp khách. Bất kỳ một sai sót nhỏ nào cũng có thể khiến uy tín của nhà hàng của bạn bị giảm sút, một điều bạn không bao giờ mong đợi. Hàng hóa dịch vụ đôi khi bị thất thoát, bạn bận rộn với rất nhiều việc, nên bạn cần biết ngay tình hình kinh doanh nhà hàng khi bạn cần, nhưng thường thì bạn phải đợi đến cuối tháng nhân viên mới hoàn tất báo cáo.

Tất cả các lý do trên đã đặt ra yêu cầu phải có hệ thống phần mềm quản lý nhà hàng với nhiều ưu điểm để khắc phục các nhược điểm trên, và thực hiện tốt công việc nghiệp vụ của các bộ phận:

* Dễ dàng quản lý, truy cập, thay đổi chỉnh sửa.
* Việc tìm kiếm, liên hệ nhanh chóng và thực hiện dễ dàng.
* Quá trình lưu trữ nhanh và lưu trữ được nhiều thông tin.
* …

# Các chức năng chính của chương trình:

* Thêm thông tin tài khoản.
* Tra cứu thông tin tài khoản
* Xoá thông tin tài khoản.
* Sửa thông tin tài khoản.
* Thêm bàn.
* Xóa bàn.
* Sửa bàn.
* Thêm thông tin món ăn.
* Xoá thông tin món ăn.
* Sửa thông tin món ăn.
* Tra cứu thông tin món ăn.
* Nhận order.
* Kết thúc order.
* Lập hóa đơn thanh toán.
* Xóa hóa đơn thanh toán.
* Tra cứu hóa đơn thanh toán.

# CHƯƠNG 2: PHÂN TÍCH THIẾT KẾ

* 1. **PHÂN TÍCH ĐỀ TÀI:**

Phần mềm quản lý nhà hàng của chúng tôi là một giải pháp toàn diện và dễ sử dụng để quản lý mọi khía cạnh của hoạt động nhà hàng. Với giao diện thân thiện và dễ sử dụng, phần mềm giúp bạn tối ưu hóa quy trình làm việc, tăng cường hiệu suất và cải thiện trải nghiệm của khách hàng.

Giao diện người dùng được thiết kế một cách trực quan và thân thiện, cho phép bạn nhanh chóng thao tác và tiếp cận các chức năng quản lý quan trọng.

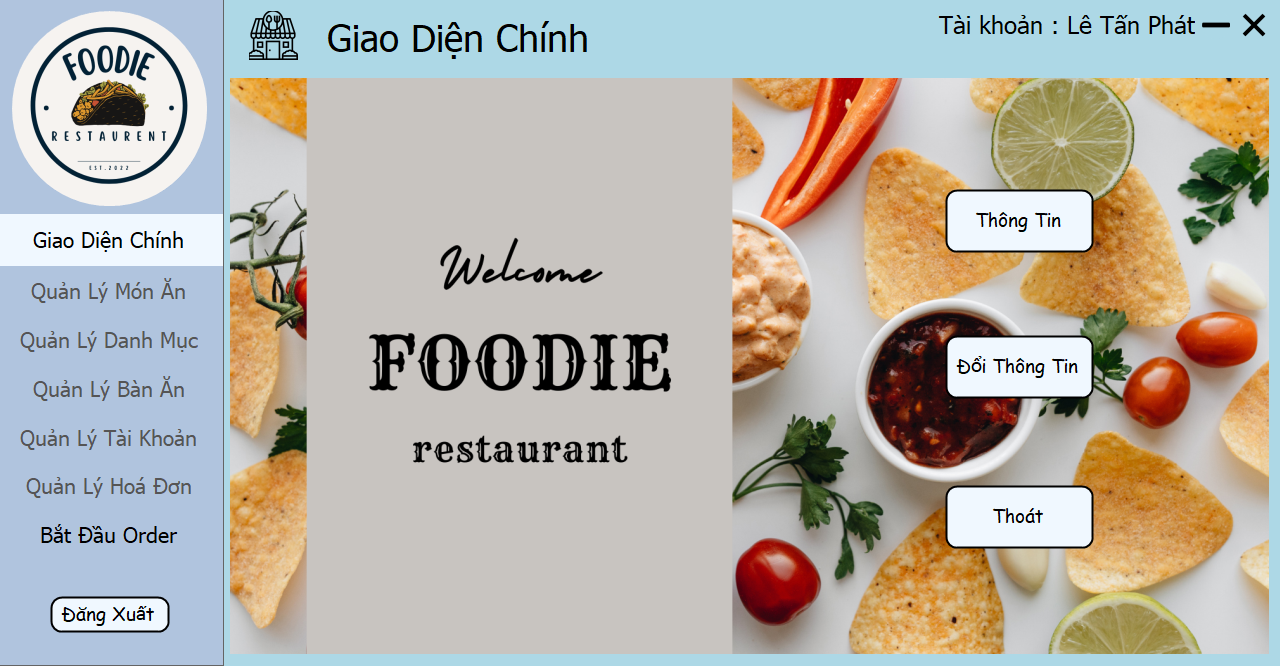
**Giao diện phần mềm:**

1. **Giao diện đăng nhập**



1. **Giao diện của nhân viên**

Nhân viên chỉ có thể thao tác với tuỳ chọn bắt đầu order



1. **Giao diện của quản lý gồm các tuỳ chọn sau**
2. Quản lý món ăn: thêm, xoá, sửa và tìm kiếm thông tin một món ăn.
3. Quản lý danh mục: thêm, xoá, sửa và tìm kiếm tên danh mục.
4. Quản lý bàn ăn: thêm, xoá, sửa và tìm kiếm bàn ăn.
5. Quản lý tài khoản: thêm, xoá, sửa và tìm kiếm tài khoản.
6. Quản lý hoá đơn: Thống kê hoá đơn và in hoá đơn.
7. Order: thêm hoặc xoá món khỏi bàn đã chọn, tính tiền và in ra hoá đơn.
8. Đăng xuất: thoát khỏi tài khoản.



**Mô tả:**Phần mềm dễ làm quen, tối ưu hóa quy trình làm việc, tăng cường hiệu suất và cải thiện trải nghiệm của khách hàng.

**TRIỂN KHAI LẬP TRÌNH**

# Code giao diện:

1. **Login.cs**

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Security.Cryptography.X509Certificates;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

using ui\_qlnhahang.DAo;

using ui\_qlnhahang.Properties;

namespace ui\_qlnhahang

{

public partial class Login : BorderForm

{

public Login()

{

InitializeComponent();

CenterToScreen();

}

public string tk ;

public string mk ;

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

{

tk = txttk.Text;

mk = txtmk.Text;

if (checkLogin(tk, mk))

{

this.Close();

}

else {

MessBox mb = new MessBox("Sai tài khoản mật khẩu!");

mb.ShowDialog();

}

}

bool checkLogin(string tk, string mk)

{

return AccountDAO.Instance.Login(tk, mk);

}

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

{

Application.Exit();

}

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

{

txtmk.PasswordChar = '\*';

pbHide.BringToFront();

this.KeyPreview = true;

}

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

{

if (txtmk.PasswordChar == '\0')

{

txtmk.PasswordChar = '\*';

pbHide.BringToFront();

}

}

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

{

if (txtmk.PasswordChar == '\*')

{

txtmk.PasswordChar = '\0';

pbShow.BringToFront();

}

}

private void Login\_KeyDown(object sender, KeyEventArgs e)

{

if (e.KeyCode == Keys.Enter)

{

btnDangNhap\_Click(sender, e);

}

}

}

}

1. **Loading.cs**

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Drawing.Imaging;

using System.IO.Ports;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace ui\_qlnhahang

{

public partial class Loading : Form

{

public Loading()

{

InitializeComponent();

CenterToParent();

handleLoadingBounds();

}

private int rotationAngle = 0;

private void timer1\_Tick(object sender, EventArgs e)

{

rotationAngle += 5;

logo.Image.RotateFlip(RotateFlipType.Rotate90FlipNone);

logo.Refresh();

if (rotationAngle >= 25) { this.Close(); }

}

void handleLoadingBounds()

{

logo.Anchor = AnchorStyles.None;

int labelWidth = logo.Width;

int labelHeight = logo.Height;

int labelX = (this.Height - labelWidth) / 2;

int labelY = (this.Height - labelHeight) / 2;

logo.SetBounds(labelX, labelY, labelWidth, labelHeight);

}

}

}

1. **Dashboard.cs**

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

using ui\_qlnhahang.Properties;

using System.Drawing;

using Bunifu.UI.WinForms.BunifuButton;

using static System.Windows.Forms.VisualStyles.VisualStyleElement;

using ui\_qlnhahang.DAo;

using static ui\_qlnhahang.FormUltility;

namespace ui\_qlnhahang

{

public partial class Dashboard : Form

{

string accountquery = "select \* from [Account]";

Form subForm = null;

public Dashboard()

{

InitializeComponent();

CenterToScreen();

}

private string tk;

private string mk;

private void CenterFormInPanel(Form form, Panel panel)

{

int panelWidth = panel.Width;

int panelHeight = panel.Height;

int formWidth = form.Width;

int formHeight = form.Height;

int left = (panelWidth - formWidth) / 2;

int top = (panelHeight - formHeight) / 2;

form.Location = new Point(left, top);

}

private void navigation(Form form, string title)

{

if (subForm != null)

{

subForm.Dispose();

panel.Controls.Remove(subForm);

}

activeButton(title);

Loading fl = new Loading();

fl.ShowDialog();

form.TopLevel = false;

form.FormBorderStyle = FormBorderStyle.None;

form.Dock = DockStyle.Fill;

subForm = form;

panel.Controls.Add(subForm);

form.Show();

changeTitle(title);

}

void activeButton(string title)

{

foreach (Control control in sidebar.Controls)

{

if (control is System.Windows.Forms.Button)

{

System.Windows.Forms.Button button = (System.Windows.Forms.Button)control;

if(button.Text == title)

{

button.BackColor = Color.AliceBlue;

} else

{

button.BackColor = Color.Transparent;

}

}

}

}

private void changeTitle(String title)

{

lblTitle.Text = title;

}

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

{

//if (MessageBox.Show("Bạn có muốn rời nhà hàng", "NHÀ HÀNG FOODIE", MessageBoxButtons.OKCancel, MessageBoxIcon.Question) == DialogResult.OK)

//{

// Application.Exit();

//}

CustomMessBox messbox = new CustomMessBox("Bạn có muốn rời nhà hàng FOODIE!");

messbox.ShowDialog();

}

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

{

FoodManage form = new FoodManage();

navigation(form, btnFoodManage.Text);

pbHeader.Image = Resources.food;

}

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

{

CateManage form = new CateManage();

navigation(form, btnCateManage.Text);

pbHeader.Image = Resources.food\_delivery;

}

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

{

BillManage form = new BillManage();

navigation(form, btnBillManage.Text);

pbHeader.Image = Resources.bill;

}

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

{

TableManage form = new TableManage();

navigation(form, btnTableManage.Text);

pbHeader.Image = Resources.dining\_table;

}

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

{

AccountManage form = new AccountManage();

navigation(form, btnAccManage.Text);

pbHeader.Image = Resources.user;

}

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

{

this.WindowState = FormWindowState.Minimized;

}

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

{

if (btnOpenOrder.Text.Equals("Bắt Đầu Order"))

{

handleStateBtn(false);

Order form = new Order(tk);

navigation(form, btnOpenOrder.Text);

btnOpenOrder.Text = "Kết Thúc Order";

pbHeader.Image = Resources.order\_food;

} else

{

handleStateBtn(true);

btnOpenOrder.Text = "Bắt Đầu Order";

Home formHome = new Home();

navigation(formHome, btnHome.Text);

pbHeader.Image = Resources.restaurant;

}

}

void handleStateBtn(bool state)

{

if (checkStaff(tk))

{

btnHome.Enabled = state;

btnAccManage.Enabled = false;

btnBillManage.Enabled = false;

btnCateManage.Enabled = false;

btnTableManage.Enabled = false;

btnFoodManage.Enabled = false;

}

else

{

btnHome.Enabled = state;

btnAccManage.Enabled = state;

btnBillManage.Enabled = state;

btnCateManage.Enabled = state;

btnTableManage.Enabled = state;

btnFoodManage.Enabled = state;

}

}

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

{

Home form = new Home(tk, mk);

navigation(form, btnHome.Text);

pbHeader.Image = Resources.restaurant;

}

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

{

Login fn = new Login();

fn.ShowDialog();

tk = fn.tk;

mk = fn.mk;

Home form = new Home(tk, mk);

navigation(form, btnHome.Text);

activeButton(btnHome.Text);

btnAccManage.Enabled = !checkStaff(tk);

btnBillManage.Enabled = !checkStaff(tk);

btnCateManage.Enabled = !checkStaff(tk);

btnTableManage.Enabled = !checkStaff(tk);

btnFoodManage.Enabled = !checkStaff(tk);

string displayName = "";

foreach (DataRow account in GetAllDataNew(accountquery).Rows)

{

if (account["AccountName"].ToString().Equals(tk))

{

displayName = account["DisplayName"].ToString();

break;

}

}

lblaccountactive.Text = "Tài khoản : " + displayName;

}

bool checkStaff(string tk)

{

return AccountDAO.Instance.checkStaff(tk);

}

private void btnDangXuat\_Click\_1(object sender, EventArgs e)

{

tk = "";

mk = "";

Login f = new Login();

f.ShowDialog();

tk = f.tk;

mk = f.mk;

Home form = new Home(tk, mk);

navigation(form, btnHome.Text);

activeButton(btnHome.Text);

btnAccManage.Enabled = !checkStaff(tk);

btnBillManage.Enabled = !checkStaff(tk);

btnCateManage.Enabled = !checkStaff(tk);

btnTableManage.Enabled = !checkStaff(tk);

btnFoodManage.Enabled = !checkStaff(tk);

string displayName = "";

foreach (DataRow account in GetAllDataNew(accountquery).Rows)

{

if (account["AccountName"].ToString().Equals(tk))

{

displayName = account["DisplayName"].ToString();

break;

}

}

lblaccountactive.Text = "Tài khoản : " + displayName;

}

}

}

1. **FoodManage.cs**

using Bunifu.UI.WinForms;

using System;

using System.Collections;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Data.SqlClient;

using System.Drawing;

using System.Linq;

using System.Runtime.InteropServices.ComTypes;

using System.Security.Cryptography;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

using System.Xml.Linq;

using ui\_qlnhahang.DAo;

using static System.Windows.Forms.VisualStyles.VisualStyleElement;

using static ui\_qlnhahang.FormUltility;

namespace ui\_qlnhahang

{

public partial class FoodManage : Form

{

public string mainquery = "SELECT F.ID, F.Name AS Name, C.Name AS CategoryName, F.Price\r\nFROM Food F\r\nJOIN Category C ON F.FoodCategoryID = C.ID";

public string queryNameOfFood = "select name from [Category]";

public string queryCategory = "select \* from [Category]";

public DataTable categoryList;

BunifuTextBox[] myTextBoxes;

public FoodManage()

{

InitializeComponent();

}

private void bunifuTextBox3\_KeyPress(object sender, KeyPressEventArgs e)

{

if (!Char.IsDigit(e.KeyChar) && !Char.IsControl(e.KeyChar))

{

e.Handled = true;

}

}

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

{

categoryList = GetTableData(queryCategory);

GetAllData(mainquery, gvFood);

GetAllData(queryNameOfFood, dpdCate);

dpdCate.Text = dpdCate.Items[0].ToString();

myTextBoxes = new BunifuTextBox[] { txtFoodName, txtPrice };

handleResetTextbox(gvFood,txtFoodName, myTextBoxes);

}

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

{

string name = txtFoodName.Text.Trim();

string foodCateText = dpdCate.Text.Trim();

string price = txtPrice.Text.Trim();

if (String.IsNullOrEmpty(name))

{

MessBox mb = new MessBox("Vui lòng nhập tên món ăn!");

mb.ShowDialog();

txtFoodName.Clear();

return;

}

if (String.IsNullOrEmpty(price))

{

MessBox mb = new MessBox("Vui lòng nhập giá món ăn!");

mb.ShowDialog();

txtPrice.Clear();

return;

}

int foodPrice = Convert.ToInt32(price);

int foodCateID = 0;

foreach (DataRow row in categoryList.Rows)

{

if (row["Name"].ToString() == foodCateText)

{

foodCateID = Convert.ToInt32(row["ID"]);

break;

}

}

foreach (DataRow item in GetAllData(mainquery, gvFood).Rows)

{

if (item["Name"].ToString().Equals(name))

{

MessBox mb = new MessBox("Tên món ăn đã có trong cơ sở dữ liệu!");

mb.ShowDialog();

handleResetTextbox(gvFood, txtFoodName, myTextBoxes);

return;

}

}

string nameProcedure = "[InsertFood]";

string procedureParams = "@Name @FoodCategoryID @Price";

string desc = "Thêm món ăn thành công";

handleProcedure(mainquery, nameProcedure, procedureParams, gvFood, desc, new object[] { name, foodCateID, foodPrice });

handleResetTextbox(gvFood, txtFoodName, myTextBoxes);

}

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

{

if (gvFood.SelectedRows.Count > 0)

{

DataGridViewRow selectedRow = gvFood.SelectedRows[0];

string name = txtFoodName.Text.Trim();

string foodCateText = dpdCate.Text.Trim();

string price = txtPrice.Text.Trim();

int foodPrice = Convert.ToInt32(price);

if (String.IsNullOrEmpty(name))

{

MessBox mb = new MessBox("Vui lòng nhập tên món ăn!");

mb.ShowDialog();

txtFoodName.Clear();

return;

}

if (String.IsNullOrEmpty(price))

{

MessBox mb = new MessBox("Vui lòng nhập giá món ăn!");

mb.ShowDialog();

txtPrice.Clear();

return;

}

object id = selectedRow.Cells[0].Value;

int foodCateID = 0;

foreach (DataRow row in categoryList.Rows)

{

if (row["Name"].ToString() == foodCateText)

{

foodCateID = Convert.ToInt32(row["ID"]);

break;

}

}

string nameProcedure = "[UpdateFood]";

string procedureParams = "@ID @Name @FoodCategoryID @Price";

string desc = "Cập nhật món ăn thành công!";

handleProcedure(mainquery, nameProcedure, procedureParams,gvFood, desc, new object[] { id, name, foodCateID, foodPrice });

handleResetTextbox(gvFood, txtFoodName, myTextBoxes);

}

else

{

MessBox mb = new MessBox("Vui lòng chọn món ăn muốn sửa!");

mb.ShowDialog();

}

}

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

{

if (gvFood.SelectedRows.Count > 0)

{

DataGridViewRow selectedRow = gvFood.SelectedRows[0];

object id = selectedRow.Cells[0].Value;

string nameProcedure = "[DeleteFood]";

string procedureParams = "@FoodID";

string desc = "Xoá món ăn thành công!";

handleProcedure(mainquery ,nameProcedure, procedureParams, gvFood, desc, new object[] { id });

handleResetTextbox(gvFood, txtFoodName, myTextBoxes);

}

else

{

MessBox mb = new MessBox("Chọn món ăn để xoá!");

mb.ShowDialog();

}

}

private void txtPrice\_KeyPress(object sender, KeyPressEventArgs e)

{

if (!char.IsControl(e.KeyChar) && !char.IsDigit(e.KeyChar))

{

e.Handled = true;

}

}

private void gvFood\_SelectionChanged(object sender, EventArgs e)

{

if (gvFood.SelectedRows.Count > 0)

{

DataGridViewRow selectedRow = gvFood.SelectedRows[0];

txtFoodName.Text = selectedRow.Cells[1].Value.ToString();

dpdCate.Text = selectedRow.Cells[2].Value.ToString();

txtPrice.Text = selectedRow.Cells[3].Value.ToString();

}

}

private void txtSearch\_TextChange(object sender, EventArgs e)

{

string columnName = "Name";

handleFilter(gvFood, txtSearch, mainquery, columnName);

if (String.IsNullOrEmpty(txtSearch.Text))

{

handleResetTextbox(gvFood, txtSearch, myTextBoxes);

}

}

}

}

1. **Cate manage.cs**

﻿ using Bunifu.UI.WinForms;

using System;

using System.Collections;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Data.SqlClient;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

using System.Xml.Linq;

using ui\_qlnhahang.DAo;

using static ui\_qlnhahang.FormUltility;

using static ui\_qlnhahang.Order;

namespace ui\_qlnhahang

{

public partial class CateManage : Form

{

string mainquery = "select \* from [Category]";

string foodCateIDQuery = "select FoodCategoryID from [Food]";

DataTable foodCateList;

BunifuTextBox[] myTextBoxes;

public CateManage()

{

InitializeComponent();

}

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

{

GetAllData(mainquery, gvCate);

foodCateList = GetAllDataNew(foodCateIDQuery);

myTextBoxes = new BunifuTextBox[] { txtCate };

handleResetTextbox(gvCate, txtCate, myTextBoxes);

}

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

{

string cateName = txtCate.Text.Trim();

if (string.IsNullOrEmpty(cateName))

{

MessBox mb = new MessBox("Vui lòng nhập tên danh mục món ăn!");

mb.ShowDialog();

txtCate.Clear();

return;

}

foreach (DataRow item in GetAllData(mainquery, gvCate).Rows)

{

if (item["Name"].ToString().Equals(cateName))

{

MessBox mb = new MessBox("Tên danh mục đã có trong cơ sở dữ liệu!");

mb.ShowDialog();

handleResetTextbox(gvCate, txtCate, myTextBoxes);

return;

}

}

string name = "Category\_Insert";

string procedureParams = "@Name @Type";

string desc = "Thêm danh mục món ăn thành công!";

handleProcedure(mainquery, name, procedureParams,gvCate, desc, new object[] { cateName, 1 });

handleResetTextbox(gvCate, txtCate, myTextBoxes);

}

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

{

if (gvCate.SelectedRows.Count > 0)

{

DataGridViewRow selectedRow = gvCate.SelectedRows[0];

string cateName = txtCate.Text.Trim();

if (string.IsNullOrEmpty(cateName))

{

MessBox mb = new MessBox("Vui lòng nhập tên danh mục món ăn!");

mb.ShowDialog();

txtCate.Clear();

return;

}

object id = selectedRow.Cells[0].Value;

string name = "Category\_Update";

string procedureParams = "@ID @Name @Type";

string desc = "Cập nhật danh mục món ăn thành công!";

handleProcedure(mainquery, name, procedureParams, gvCate, desc, new object[] { id, cateName, 1 });

handleResetTextbox(gvCate, txtCate, myTextBoxes);

}

else

{

MessBox mb = new MessBox("Vui lòng chọn tên danh mục món ăn muốn sửa!");

mb.ShowDialog();

}

}

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

{

if (gvCate.SelectedRows.Count > 0)

{

DataGridViewRow selectedRow = gvCate.SelectedRows[0];

string name = "Category\_Delete";

string procedureParams = "@ID";

string desc = "Xoá danh mục món ăn thành công!";

object id = selectedRow.Cells[0].Value;

foreach (DataRow item in foodCateList.Rows)

{

if (item["FoodCategoryID"].ToString().Equals(id))

{

MessBox mb = new MessBox("Hiện đang có món ăn sử dụng danh mục món ăn này!");

mb.ShowDialog();

return;

} else

{

handleProcedure(mainquery, name, procedureParams, gvCate, desc, new object[] { id });

}

}

handleResetTextbox(gvCate, txtCate, myTextBoxes);

}

else

{

MessBox mb = new MessBox("Vui lòng chọn tên danh mục món ăn muốn xoá!");

mb.ShowDialog();

}

}

private void gvCate\_SelectionChanged(object sender, EventArgs e)

{

if (gvCate.SelectedRows.Count > 0)

{

DataGridViewRow selectedRow = gvCate.SelectedRows[0];

txtCate.Text = selectedRow.Cells[1].Value.ToString();

}

}

private void txtSearch\_TextChange(object sender, EventArgs e)

{

string columnName = "Name";

handleFilter(gvCate, txtSearch, mainquery, columnName);

if(String.IsNullOrEmpty(txtSearch.Text))

{

handleResetTextbox(gvCate, txtSearch, myTextBoxes);

}

}

}

}

1. **Table manage.cs**

using Bunifu.UI.WinForms;

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Security.Principal;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

using ui\_qlnhahang.DAo;

using static ui\_qlnhahang.FormUltility;

using static ui\_qlnhahang.Order;

namespace ui\_qlnhahang

{

public partial class TableManage : Form

{

string mainquery = "select ID, Name from [Table]";

BunifuTextBox[] myTextBoxes;

public TableManage()

{

InitializeComponent();

}

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

{

GetAllData(mainquery, gvTable);

txtNameTable.Clear();

myTextBoxes = new BunifuTextBox[] { txtNameTable };

handleResetTextbox(gvTable, txtNameTable, myTextBoxes);

}

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

{

string tableName = txtNameTable.Text.Trim();

if (string.IsNullOrEmpty(tableName))

{

MessBox mb = new MessBox("Vui lòng nhập tên bàn!");

mb.ShowDialog();

txtNameTable.Clear();

return;

}

foreach (DataRow item in GetAllData(mainquery, gvTable).Rows)

{

if (item["Name"].ToString().Equals(tableName))

{

MessBox mb = new MessBox("Tên bàn đã có trong cơ sở dữ liệu!");

mb.ShowDialog();

handleResetTextbox(gvTable, txtNameTable, myTextBoxes);

return;

}

}

string name = "Table\_Insert";

string procedureParams = "@Name @Status";

string desc = "Thêm bàn thành công!";

handleProcedure(mainquery, name, procedureParams, gvTable, desc, new object[] { tableName, 0 });

handleResetTextbox(gvTable, txtNameTable, myTextBoxes);

}

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

{

if (gvTable.SelectedRows.Count > 0)

{

DataGridViewRow selectedRow = gvTable.SelectedRows[0];

string tableName = txtNameTable.Text.Trim();

if (string.IsNullOrEmpty(tableName))

{

MessBox mb = new MessBox("Vui lòng nhập tên bàn!");

mb.ShowDialog();

handleResetTextbox(gvTable, txtNameTable, myTextBoxes);

return;

}

object id = selectedRow.Cells[0].Value;

string name = "Table\_Update";

string procedureParams = "@ID @Name @Status";

string desc = "Cập nhật bàn thành công!";

handleProcedure(mainquery, name, procedureParams, gvTable, desc, new object[] { id, tableName, 0 });

handleResetTextbox(gvTable, txtNameTable, myTextBoxes);

}

else

{

MessBox mb = new MessBox("Vui lòng chọn bàn để sửa!");

mb.ShowDialog();

}

}

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

{

if (gvTable.SelectedRows.Count > 0)

{

DataGridViewRow selectedRow = gvTable.SelectedRows[0];

string name = "Table\_Delete";

string procedureParams = "@ID";

string desc = "Xoá bàn thành công!";

object id = selectedRow.Cells[0].Value;

handleProcedure(mainquery, name, procedureParams,gvTable, desc, new object[] { id });

handleResetTextbox(gvTable, txtNameTable, myTextBoxes);

}

else

{

MessBox mb = new MessBox("Vui lòng chọn bàn để xoá!");

mb.ShowDialog();

}

}

private void gvTable\_SelectionChanged(object sender, EventArgs e)

{

if (gvTable.SelectedRows.Count > 0)

{

DataGridViewRow selectedRow = gvTable.SelectedRows[0];

txtNameTable.Text = selectedRow.Cells[1].Value.ToString();

}

}

private void txtSearch\_TextChange(object sender, EventArgs e)

{

string columnName = "Name";

handleFilter(gvTable, txtSearch, mainquery, columnName);

if (String.IsNullOrEmpty(txtSearch.Text))

{

handleResetTextbox(gvTable, txtSearch, myTextBoxes);

}

}

}

}

1. **Account Manage.cs**

using Bunifu.UI.WinForms;

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

using System.Xml.Linq;

using static ui\_qlnhahang.FormUltility;

using static ui\_qlnhahang.Order;

namespace ui\_qlnhahang

{

public partial class AccountManage : Form

{

string mainquery = "SELECT A.AccountName as Name, A.DisplayName as DisplayName, A.Password, R.RoleName\r\nFROM dbo.Account A\r\nJOIN dbo.RoleAccount RA ON A.AccountName = RA.AccountName\r\nJOIN dbo.Role R ON RA.RoleID = R.ID;";

string queryRoleName = "select [RoleName] from [Role]";

string queryRole = "select \* from [Role]";

DataTable roles;

DataTable list;

BunifuTextBox[] myTextBoxes;

public AccountManage()

{

InitializeComponent();

}

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

{

roles = GetTableData(queryRole);

GetAllData(mainquery, gvAccount);

GetAllData(queryRoleName, dpdType);

dpdType.Text = dpdType.Items[0].ToString();

myTextBoxes = new BunifuTextBox[] { txtUserName, txtUserNameDisplay, txtPass };

handleResetTextbox(gvAccount, txtUserName, myTextBoxes);

}

private void gvAccount\_ColumnHeadersHeightChanged(object sender, EventArgs e)

{

gvAccount.Rows[0].HeaderCell.Value = new Size(gvAccount.Rows[0].HeaderCell.Size.Width, 50);

}

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

{

string name = txtUserName.Text.Trim();

string displayName = txtUserNameDisplay.Text.Trim();

string pass = txtPass.Text.Trim();

if (String.IsNullOrEmpty(name))

{

MessBox mb = new MessBox("Vui lòng nhập tên tài khoản!");

mb.ShowDialog();

txtUserName.Clear();

return;

}

if (String.IsNullOrEmpty(displayName))

{

MessBox mb = new MessBox("Vui lòng nhập tên hiển thị!");

mb.ShowDialog();

txtUserNameDisplay.Clear();

return;

}

if (String.IsNullOrEmpty(pass))

{

MessBox mb = new MessBox("Vui lòng nhập mật khẩu!");

mb.ShowDialog();

txtPass.Clear();

return;

}

foreach (DataRow item in GetAllData(mainquery, gvAccount).Rows)

{

if (item["Name"].ToString().Equals(name))

{

MessBox mb = new MessBox("Tên tài khoản đã có trong cơ sỡ dữ liệu!");

mb.ShowDialog();

handleResetTextbox(gvAccount, txtUserName, myTextBoxes);

return;

}

}

string role = dpdType.Text;

string procedureName = "InsertAccount";

string procedureParams = "@AccountName @DisplayName @Password @RoleName";

string desc = "Thêm tài khoản thành công";

handleProcedure(mainquery, procedureName, procedureParams, gvAccount, desc, new object[] {name, displayName, pass, role});

handleResetTextbox(gvAccount, txtUserName, myTextBoxes);

}

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

{

if (gvAccount.SelectedRows.Count > 0)

{

DataGridViewRow selectedRow = gvAccount.SelectedRows[0];

string name = txtUserName.Text.Trim();

string displayName = txtUserNameDisplay.Text.Trim();

string pass = txtPass.Text.Trim();

if (String.IsNullOrEmpty(name))

{

MessBox mb = new MessBox("Vui lòng nhập tên tài khoản!");

mb.ShowDialog();

txtUserName.Clear();

return;

}

if (String.IsNullOrEmpty(displayName))

{

MessBox mb = new MessBox("Vui lòng nhập tên hiển thị!");

mb.ShowDialog();

txtUserNameDisplay.Clear();

return;

}

if (String.IsNullOrEmpty(pass))

{

MessBox mb = new MessBox("Vui lòng nhập mật khẩu!");

mb.ShowDialog();

txtPass.Clear();

return;

}

int roleId = 0;

string role = dpdType.Text;

foreach (DataRow row in roles.Rows)

{

if (row["RoleName"].ToString() == role)

{

roleId = Convert.ToInt32(row["ID"]);

break;

}

}

if (selectedRow.Cells[0].Value.ToString() != name)

{

MessBox mb = new MessBox("Không thể đổi tên đăng nhập tài khoản!");

mb.ShowDialog();

txtUserName.Clear();

return;

}

string nameProcedure = "[UpdateAccountWithRoleID]";

string procedureParams = "@AccountName @DisplayName @Password @NewRoleID";

string desc = "Cập nhật tài khoản thành công!";

handleProcedure(mainquery, nameProcedure, procedureParams, gvAccount, desc, new object[] { name, displayName, pass, roleId });

handleResetTextbox(gvAccount, txtUserName, myTextBoxes);

} else

{

MessBox mb = new MessBox("Vui lòng chọn tài khoản muốn sửa!");

mb.ShowDialog();

}

}

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

{

if (gvAccount.SelectedRows.Count > 0)

{

DataGridViewRow selectedRow = gvAccount.SelectedRows[0];

object name = selectedRow.Cells[0].Value;

string nameProcedure = "[DeleteAccountWithRole]";

string procedureParams = "@AccountName";

string desc = "Xoá tài khoản thành công!";

handleProcedure(mainquery, nameProcedure, procedureParams, gvAccount, desc, new object[] { name });

handleResetTextbox(gvAccount, txtUserName, myTextBoxes);

}

else

{

MessBox mb = new MessBox("Chọn tài khoản để xoá");

mb.ShowDialog();

}

}

private void gvAccount\_SelectionChanged(object sender, EventArgs e)

{

if (gvAccount.SelectedRows.Count > 0)

{

DataGridViewRow selectedRow = gvAccount.SelectedRows[0];

txtUserName.Text = selectedRow.Cells[0].Value.ToString();

txtUserNameDisplay.Text = selectedRow.Cells[1].Value.ToString();

txtPass.Text = selectedRow.Cells[2].Value.ToString();

dpdType.Text = selectedRow.Cells[3].Value.ToString();

}

}

private void txtSearch\_TextChange(object sender, EventArgs e)

{

string columnName = "Name";

handleFilter(gvAccount, txtSearch, mainquery, columnName);

if (String.IsNullOrEmpty(txtSearch.Text))

{

handleResetTextbox(gvAccount, txtSearch, myTextBoxes);

}

}

}

}

1. **Bill manage.cs**

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

using System.Data.SqlClient;

using static ui\_qlnhahang.Order;

using static ui\_qlnhahang.FormUltility;

using ui\_qlnhahang.DAo;

namespace ui\_qlnhahang

{

public partial class BillManage : Form

{

string mainquery = "select ID, TableID, Amount, Status,CheckoutDate, Account from [Bills]";

public BillManage()

{

InitializeComponent();

}

private void Form4\_Load(object sender, EventArgs e)

{

dpFrom.Value = DateTime.Today;

dpTo.Value = DateTime.Today.AddDays(1);

dpFrom.Format = DateTimePickerFormat.Custom;

dpFrom.CustomFormat = "dd/MM/yyyy";

dpTo.Format = DateTimePickerFormat.Custom;

dpTo.CustomFormat = "dd/MM/yyyy";

GetAllData(mainquery, gvBill);

gvBill.ClearSelection();

handleFormatDate();

}

private void handleFormatDate()

{

foreach (DataGridViewColumn column in gvBill.Columns)

{

foreach (DataGridViewRow row in gvBill.Rows)

{

if (row.Cells[column.Index].Value is DateTime)

{

column.DefaultCellStyle.Format = "dd/MM/yyyy";

return;

}

}

}

}

private void dpTo\_ValueChanged(object sender, EventArgs e)

{

if (dpTo.Value < dpFrom.Value)

{

MessBox mb = new MessBox("Giá trị ngày không hợp lệ!");

mb.ShowDialog();

dpTo.Value = dpFrom.Value;

}

}

private void dpFrom\_ValueChanged(object sender, EventArgs e)

{

if (dpFrom.Value > dpTo.Value)

{

dpTo.Value = dpFrom.Value;

}

}

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

{

if (gvBill.SelectedRows.Count > 0)

{

// Lấy ID của hóa đơn được chọn

int selectedBillID = Convert.ToInt32(gvBill.SelectedRows[0].Cells["billID"].Value);

// Tạo form Chi tiết hóa đơn và truyền ID cho nó

BillDetail detailForm = new BillDetail(selectedBillID);

detailForm.ShowDialog();

}

}

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

{

DateTime startDate = dpFrom.Value;

DateTime endDate = dpTo.Value;

filterBill(startDate, endDate);

}

private void filterBill(DateTime start, DateTime end)

{

string columnName = "billCheckout";

foreach (DataGridViewRow row in gvBill.Rows)

{

if (row.Cells[columnName].Value != null && row.Cells[columnName].Value is DateTime)

{

DateTime dateValue = (DateTime)row.Cells[columnName].Value;

if (dateValue >= start && dateValue < end.AddDays(1))

{

row.Visible = true;

}

else

{

row.Visible = false;

}

}

else

{

row.Visible = false;

}

}

}

}

}

1. **Order.cs**

using System;

using System.Collections;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Data.SqlClient;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

using ui\_qlnhahang.DAo;

using ui\_qlnhahang.Properties;

using static System.Net.Mime.MediaTypeNames;

using static ui\_qlnhahang.Order;

using Image = System.Drawing.Image;

using static ui\_qlnhahang.FormUltility;

namespace ui\_qlnhahang

{

public partial class Order : Form

{

private string tk;

public int totalTable;

string query = "select ID from [Table]";

DataTable listTable;

public Order(string tk)

{

InitializeComponent();

this.tk = tk;

//listTable = GetAllDataNew(query);

GetAllFoodData();

LoadTables();

for (int i = 0; i <= 100; i++)

{

orderManager.CreateOrder(i, DateTime.Now);

}

//orderManager.CreateOrder(0, DateTime.Now);

//foreach (DataRow item in listTable.Rows)

//{

// orderManager.CreateOrder(Convert.ToInt32(item["ID"]), DateTime.Now);

// MessageBox.Show(item["ID"].ToString());

//}

}

OrderManager orderManager = new OrderManager();

public class OrderItem

{

public int FoodId { get; set; }

public string FoodName { get; set; }

public int Quantity { get; set; }

public int Price { get; set; }

}

public class OrderListed

{

public int OrderId { get; set; }

public DateTime OrderDate { get; set; }

public List<OrderItem> OrderItems { get; set; }

public OrderListed()

{

OrderItems = new List<OrderItem>();

}

public void AddItem(int foodId, string foodName, int quantity, int price)

{

OrderItem item = new OrderItem

{

FoodId = foodId,

FoodName = foodName,

Quantity = quantity,

Price = price

};

OrderItems.Add(item);

}

public void RemoveItem(int foodId)

{

OrderItem item = OrderItems.Find(i => i.FoodId == foodId);

if (item != null)

{

OrderItems.Remove(item);

}

}

public void UpdateQuantity(int foodId, int newQuantity)

{

OrderItem item = OrderItems.Find(i => i.FoodId == foodId);

if (item != null)

{

item.Quantity = newQuantity;

}

}

public int GetTotalQuantity()

{

int totalQuantity = 0;

foreach (OrderItem item in OrderItems)

{

totalQuantity += item.Quantity;

}

return totalQuantity;

}

}

public class OrderManager

{

public List<OrderListed> orders;

public OrderManager()

{

orders = new List<OrderListed>();

}

public void loadorderToGridView(int index, DataGridView dataGridView)

{

OrderListed order = orders[index];

foreach (OrderItem item in order.OrderItems)

{

string foodName = item.FoodName;

int price = item.Price;

int quantity = item.Quantity;

int totalPrice = price \* quantity;

dataGridView.Rows.Add(foodName, price, quantity, totalPrice);

}

}

int invoiceID = 0;

public void checkoutToBills(int index,string staffname)

{

DataProvider provider = new DataProvider();

OrderListed order = orders[index];

int totalAmount = 0;

foreach (OrderItem item in order.OrderItems)

{

int quantity = item.Quantity;

int amount = item.Price \* quantity;

totalAmount += amount;

}

if (totalAmount != 0) {

string name = "Hóa đơn " + (invoiceID + 1);

int tableID = index;

//int tableID = index + 1;

float status = 1;

// đổi thành định dạng chuẩn ISO "yyyy-MM-dd"

string checkoutDate = DateTime.Now.ToString("yyyy-MM-dd");

string account = staffname.ToLower();

string query = "INSERT INTO [dbo].[Bills] ([Name], [TableID], [Amount], [Status], [CheckoutDate], [Account]) " +

"VALUES (@Name, @TableID, @Amount, @Status, @CheckoutDate, @Account)";

using (SqlConnection connection = new SqlConnection(provider.getconnectionSTR()))

using (SqlCommand command = new SqlCommand(query, connection))

{

command.Parameters.AddWithValue("@Name", name);

command.Parameters.AddWithValue("@TableID", tableID);

command.Parameters.AddWithValue("@Amount", totalAmount);

command.Parameters.AddWithValue("@Status", status);

command.Parameters.AddWithValue("@CheckoutDate", checkoutDate);

command.Parameters.AddWithValue("@Account", account);

connection.Open();

command.ExecuteNonQuery();

connection.Close();

}

}

else

{

MessBox mb = new MessBox("Vui lòng đặt món ăn!");

mb.ShowDialog();

return;

}

}

public void checkoutToBillDetails(int index)

{

OrderListed order = orders[index];

int totalAmount = 0;

DataProvider provider = new DataProvider();

string query1 = "SELECT MAX(ID) FROM Bills";

using (SqlConnection connection = new SqlConnection(provider.getconnectionSTR()))

{

connection.Open();

using (SqlCommand command = new SqlCommand(query1, connection))

{

object result = command.ExecuteScalar();

if (result != DBNull.Value)

{

invoiceID = Convert.ToInt32(result);

}

}

connection.Close();

}

foreach (OrderItem item in order.OrderItems)

{

int name = invoiceID;

int fooid = item.FoodId;

int quantity = item.Quantity;

string query = "INSERT INTO [dbo].[BillDetails] ([InvoiceID], [FoodID], [Quantity]) " +

"VALUES (@Name, @TableID, @Amount)";

using (SqlConnection connection = new SqlConnection(provider.getconnectionSTR()))

using (SqlCommand command = new SqlCommand(query, connection))

{

command.Parameters.AddWithValue("@Name", invoiceID);

command.Parameters.AddWithValue("@TableID", fooid);

command.Parameters.AddWithValue("@Amount", quantity);

connection.Open();

command.ExecuteNonQuery();

connection.Close();

}

}

// Lấy ID của hóa đơn được chọn

if ( order.OrderItems.Count != 0) {

BillDetail detailForm = new BillDetail(invoiceID);

detailForm.ShowDialog();

}

order.OrderItems.Clear();

// Tạo form Chi tiết hóa đơn và truyền ID cho nó

}

public void CreateOrder(int orderId, DateTime orderDate)

{

OrderListed order = new OrderListed

{

OrderId = orderId,

OrderDate = orderDate

};

orders.Add(order);

}

public void AddItemToOrder(int orderId, int foodId, string foodName, int quantity, int price)

{

OrderListed order = orders.Find(o => o.OrderId == orderId);

if (order != null)

{

OrderItem existingItem = order.OrderItems.FirstOrDefault(item => item.FoodId == foodId);

if (existingItem != null)

{

existingItem.Quantity += quantity;

}

else

{

order.AddItem(foodId, foodName, quantity, price);

}

}

}

public void RemoveItemFromOrder(int orderId, int foodId)

{

OrderListed order = orders.Find(o => o.OrderId == orderId);

if (order != null)

{

order.RemoveItem(foodId);

}

}

public void UpdateQuantityInOrder(int orderId, int foodId, int newQuantity)

{

OrderListed order = orders.Find(o => o.OrderId == orderId);

if (order != null)

{

order.UpdateQuantity(foodId, newQuantity);

}

}

public int GetTotalQuantityInOrder(int orderId)

{

OrderListed order = orders.Find(o => o.OrderId == orderId);

if (order != null)

{

return order.GetTotalQuantity();

}

return 0;

}

}

List<string> loadFoodlist(int foodCategoryId)

{

/\*string query = "Select FoodCategoryID, Name from [Food]";

DataProvider provider = new DataProvider();

AddNamesToDropdown(provider.ExecuteQuery(query));\*/

string query = $"SELECT Name FROM Food WHERE FoodCategoryID = {foodCategoryId}";

DataProvider provider = new DataProvider();

return AddNamesToDropdown(provider.ExecuteQuery(query));

}

void loadCatlist()

{

string query = "Select ID, Name from [Category]";

DataProvider provider = new DataProvider();

AddCatsToDropdown(provider.ExecuteQuery(query));

}

int getInvoiceID()

{

int invoiceID = 0;

string query = "SELECT InvoiceID FROM BillDetails WHERE ID = (SELECT MAX(ID) FROM BillDetails)";

DataProvider provider = new DataProvider();

DataTable dataTable = provider.ExecuteQuery(query);

if (dataTable.Rows.Count > 0)

{

DataRow lastRow = dataTable.Rows[dataTable.Rows.Count - 1];

invoiceID = Convert.ToInt32(lastRow["InvoiceID"]);

}

return invoiceID;

}

public class Food

{

public int FoodId { get; set; }

public string FoodName { get; set; }

public int FoodCategoryId { get; set; }

public int Price { get; set; }

}

/\*public List<Food> GetAllFoodData()

{

List<Food> foodList = new List<Food>();

string query = "SELECT ID, Name, FoodCategoryID, Price FROM Food";

DataProvider provider = new DataProvider();

DataTable dataTable = provider.ExecuteQuery(query);

foreach (DataRow row in dataTable.Rows)

{

int foodId = (int)row["ID"];

string foodName = (string)row["Name"];

int foodCategoryId = (int)row["FoodCategoryID"];

int foodPrice = (int)row["Price"];

Food food = new Food

{

FoodId = foodId,

FoodName = foodName,

FoodCategoryId = foodCategoryId,

Price = foodPrice

};

foodList.Add(food);

}

return foodList;

}\*/

private List<Food> foodList;

void GetAllFoodData()

{

string query = "SELECT [ID], [Name], [FoodCategoryID], [Price] FROM [dbo].[Food]";

DataProvider provider = new DataProvider();

DataTable dataTable = provider.ExecuteQuery(query);

foodList = new List<Food>();

foreach (DataRow row in dataTable.Rows)

{

int foodId = Convert.ToInt32(row["ID"]);

string foodName = row["Name"].ToString();

int foodCategoryId = Convert.ToInt32(row["FoodCategoryID"]);

int price = Convert.ToInt32(row["Price"]);

Food food = new Food

{

FoodId = foodId,

FoodName = foodName,

FoodCategoryId = foodCategoryId,

Price = price

};

foodList.Add(food);

}

}

int tableindex = 1;

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

{

//string selectedFoodName = FoodlistDropdown1.SelectedItem.ToString();

if (FoodDataGridView1.SelectedRows.Count > 0)

{

int selectedIndex = FoodDataGridView1.SelectedRows[0].Index;

string selectedFoodName = FoodDataGridView1.SelectedRows[0].Cells[0].Value.ToString();

Food selectedFood = foodList.Find(food => food.FoodName == selectedFoodName);

orderManager.RemoveItemFromOrder(tableindex, selectedFood.FoodId);

FoodDataGridView1.Rows.Clear();

orderManager.loadorderToGridView(tableindex, FoodDataGridView1);

getTotalBill();

} else

{

MessBox mess = new MessBox("Vui lòng chọn món để xoá!");

mess.ShowDialog();

}

if (FoodDataGridView1.Rows.Count == 0)

{

clearTable(tableindex);

}

}

// button thêm món t đổi tên

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

{

string buttonName = "btnTable\_" + tableindex.ToString();

Control[] buttons = this.Controls.Find(buttonName, true);

if (buttons.Length > 0 && buttons[0] is Button)

{

Button button = (Button)buttons[0];

button.BackgroundImage = Resources.dining\_table; // Set the desired color

}

string selectedFoodName = FoodlistDropdown1.SelectedItem.ToString();

int quantity = (int)numericUpDown1.Value;

Food selectedFood = foodList.Find(food => food.FoodName.Equals(selectedFoodName));

if (selectedFood != null)

{

//int totalPrice = selectedFood.Price \* quantity;

int totalPrice = selectedFood.Price;

orderManager.AddItemToOrder(tableindex, selectedFood.FoodId, selectedFood.FoodName, quantity, totalPrice);

//FoodDataGridView1.Rows.Add(selectedFood.FoodName, selectedFood.Price, quantity, totalPrice);

FoodDataGridView1.Rows.Clear();

orderManager.loadorderToGridView(tableindex, FoodDataGridView1);

getTotalBill();

// reset sau khi add món ăn

numericUpDown1.Value = 1;

}

}

public int getTotalBill() {

int totalValue = 0;

foreach (DataGridViewRow row in FoodDataGridView1.Rows)

{

if (row.Cells["TotalPrice"].Value != null && int.TryParse(row.Cells["TotalPrice"].Value.ToString(), out int totalPrice))

{

totalValue += totalPrice;

}

}

bunifuLabel7.Text = totalValue.ToString("N0") + " VND";

return totalValue;

}

public List<string> AddNamesToDropdown(DataTable data)

{

FoodlistDropdown1.Items.Clear();

List<string> foodNames = new List<string>();

foreach (DataRow row in data.Rows)

{

string name = row["Name"].ToString();

FoodlistDropdown1.Items.Add(name);

foodNames.Add(name);

}

return foodNames;

}

public void AddCatsToDropdown(DataTable data)

{

/\*TypeDropdown.Items.Clear(); // Clear existing items

foreach (DataRow row in data.Rows)

{

string name = row["Name"].ToString();

TypeDropdown.Items.Add(name);

}\*/

TypeDropdown.Items.Clear();

foreach (DataRow row in data.Rows)

{

string name = row["Name"].ToString();

TypeDropdown.Items.Add(name);

}

// Select the default category

TypeDropdown.SelectedIndex = 0;

}

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

{

//loadFoodlist();

loadCatlist();

}

public void LoadTables()

{

panelBtns.Controls.Clear();

panelBtns.Padding = new Padding(0);

// Tạo kết nối đến cơ sở dữ liệu và truy vấn danh sách bàn ăn

DataProvider provider = new DataProvider();

DataTable tableData = provider.ExecuteQuery("select \* from [Table]");

// Sử dụng FlowLayoutPanel chính để cuộn dọc

FlowLayoutPanel mainFlowLayoutPanel = new FlowLayoutPanel();

mainFlowLayoutPanel.FlowDirection = FlowDirection.TopDown;

mainFlowLayoutPanel.WrapContents = false;

mainFlowLayoutPanel.AutoSize = true;

mainFlowLayoutPanel.Anchor = AnchorStyles.None;

mainFlowLayoutPanel.Dock = DockStyle.Fill;

int maxButtonsPerRow = 4; // Số lượng button tối đa trong một dòng

int buttonWidth = 100;

int buttonHeight = 100;

int rowWidth = maxButtonsPerRow \* buttonWidth;

FlowLayoutPanel currentFlowLayoutPanel = null; // FlowLayoutPanel hiện tại đang chứa các button

// Duyệt qua từng dòng dữ liệu trong bảng

foreach (DataRow row in tableData.Rows)

{

totalTable++;

// Nếu currentFlowLayoutPanel chưa được tạo hoặc đã đầy dòng, tạo một dòng mới

if (currentFlowLayoutPanel == null || currentFlowLayoutPanel.Controls.Count == maxButtonsPerRow)

{

currentFlowLayoutPanel = new FlowLayoutPanel();

currentFlowLayoutPanel.FlowDirection = FlowDirection.LeftToRight;

currentFlowLayoutPanel.WrapContents = false;

currentFlowLayoutPanel.AutoSize = true;

// Thêm FlowLayoutPanel hiện tại vào FlowLayoutPanel chính

mainFlowLayoutPanel.Controls.Add(currentFlowLayoutPanel);

}

TableLayoutPanel tableLayout = new TableLayoutPanel();

tableLayout.ColumnCount = 1;

tableLayout.RowCount = 2;

tableLayout.AutoSize = true;

Label tableLabel = new Label();

tableLabel.Text = row["Name"].ToString();

tableLabel.TextAlign = ContentAlignment.MiddleCenter;

tableLabel.Font = new Font("Tahoma", 10, FontStyle.Bold);

// Tạo một nút đại diện cho mỗi bàn ăn

Button tableButton = new Button();

//tableButton.Text = row["Name"].ToString();

tableButton.Name = "btnTable\_" + row["ID"].ToString();

tableButton.Width = buttonWidth;

tableButton.Height = buttonHeight;

tableButton.BackgroundImage = Resources.emty\_table;

tableButton.BackgroundImageLayout = ImageLayout.Stretch;

tableButton.FlatStyle = FlatStyle.Flat;

tableButton.Margin = new Padding(10);

// Xử lý sự kiện khi nút được nhấp

tableButton.Click += TableButton\_Click;

tableLayout.Controls.Add(tableLabel, 0, 0);

tableLayout.Controls.Add(tableButton, 0, 1);

currentFlowLayoutPanel.Controls.Add(tableLayout);

}

// Thêm FlowLayoutPanel chính vào panel chứa

panelBtns.Controls.Add(mainFlowLayoutPanel);

panelBtns.AutoScroll = true;

panelBtns.AutoScrollMinSize = mainFlowLayoutPanel.PreferredSize;

mainFlowLayoutPanel.Anchor = AnchorStyles.None;

mainFlowLayoutPanel.AutoSizeMode = AutoSizeMode.GrowAndShrink;

// Tính toán vị trí căn giữa

int x = (panelBtns.Width - mainFlowLayoutPanel.Width) / 2;

mainFlowLayoutPanel.Location = new Point(x, mainFlowLayoutPanel.Location.Y);

}

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

{

Button tableButton = (Button)sender;

int tableID = Convert.ToInt32(tableButton.Name.Split('\_')[1]);

tableindex = tableID;

FoodDataGridView1.Rows.Clear();

orderManager.loadorderToGridView(tableID, FoodDataGridView1);

getTotalBill();

//MessageBox.Show(tableID.ToString());

}

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

{

int selectedCategoryId = TypeDropdown.SelectedIndex + 1;

List<string> foodNames = loadFoodlist(selectedCategoryId);

if (foodNames.Count > 0)

{

FoodlistDropdown1.SelectedIndex = 0;

}

}

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

{

//Application.Exit();

}

List<OrderListed> tableList = new List<OrderListed>();

private int currentRowIndex = 0;

void clearTable(int tableindex)

{

string buttonName = "btnTable\_" + tableindex.ToString();

Control[] buttons = this.Controls.Find(buttonName, true);

if (buttons.Length > 0 && buttons[0] is Button)

{

Button button = (Button)buttons[0];

button.BackgroundImage = Resources.emty\_table; // Set the desired color

}

}

private void btnCheckout\_Click\_1(object sender, EventArgs e)

{

clearTable(tableindex);

//orderManager.checkoutToBillDetails(tableindex, label1);

orderManager.checkoutToBills(tableindex, tk);

orderManager.checkoutToBillDetails(tableindex);

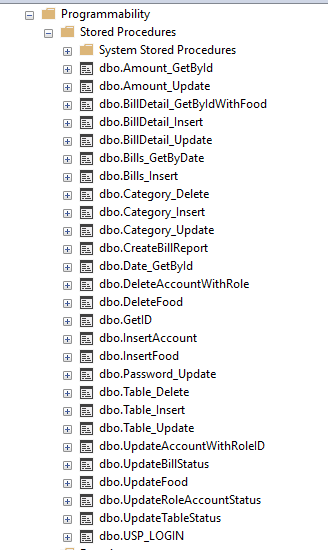
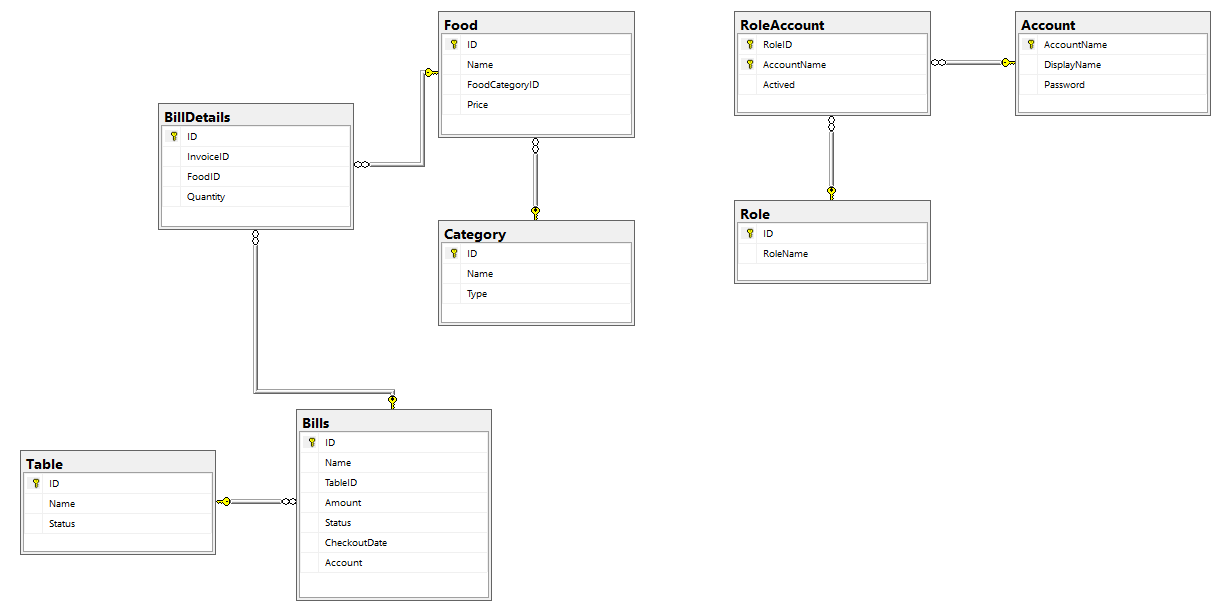
FoodDataGridView1.Rows.Clear();

}

}

}

# SQL



# Quá Trình

1. **Môi trường làm việc**

Visual Studio 2022

.NET Framework 4.7.

SQL Server

Bunifu UI/UX Framework

Github: <https://github.com/tranhuuloc4423/restaurant_manage>

1. **Phân công công việc**

|  |  |
| --- | --- |
| Lê Văn Tuấn | SQL Design |
| Trần Hữu Lộc | UI Management |
| Nguyễn Thiện Nhân | SQL Connection |
| Nguyễn Minh Quân | UI Order |
| Lê Tấn Phát | UI Login, Word & Powerpoint |

1. **Tài liệu tham khảo**

Youtube: <https://www.youtube.com/@KTeam>.

ChatGPT: [https://Poe.com](https://poe.com/).