E.P.N

MENU DE CADENA DE RESTAURANTES

INTEGRANTES:
ANDRÉS OTO
MATHÍAS BORJA
DAVID PILATUÑA

CONTENIDOS

INTRODUCCIÓN

Este documento describe el desarrollo, funcionamiento y posibles modificaciones del proyecto CRUD Cadenas de Cadena de Restaurantes desarrollado en C++. El sistema permite gestionar cadena de restaurantes mediante las operaciones básicas: Crear, Leer, Actualizar y Eliminar, facilitando la administración de productos o platillos de manera ordenada.

El proyecto fue diseñado con fines académicos, aplicando conceptos fundamentales de programación estructurada, uso de estructuras, vectores y control de flujo.

```
#include "mainwindow.h"
#include "./ui_mainwindow.h"
#include <iostream>
#include <fstream>
#include <vector>
#include <string>
#include <QMessageBox>   //LO PUSE PPARA MENSAJES
using namespace std;
```

```
private:
    Ui::MainWindow *ui;

    //definimos funciones
    std::vector<Menu> cargarMenu();
    Menu  agregarMenu();
    void  guardarMenu();
    void  leerMenu();
    void  buscarMenu();
    void actualizarMenu();
    void eliminarMenu();
```

OBJETIVOS

OBJETIVO GENERAL

Desarrollar un sistema CRUD en C++ que permita administrar una cadena de restaurante de manera eficiente mediante consola.

OBJETIVOS ESPECÍFICOS

- Implementar estructuras para representar los datos de la cadena de restaurante.
- Aplicar las operaciones CRUD (Crear, Leer, Actualizar, Eliminar).
- Utilizar vectores para el almacenamiento dinámico de datos.
- Manejar entradas y salidas por consola.
- Facilitar futuras modificaciones y mejoras al sistema.

```
Menu MainWindow::agregarMenu() {        //AGREGAR
    Menu nuevo;
    QString idTexto = ui->digitar_id->text();
    QString precioTexto = ui->escribir_precio->text();
```

```
void MainWindow::leerMenu() {       //PARA MOSTRAR TODOS LOS MENUS
    ui->tabla_registros->setRowCount(0);
    if (listaMenus.empty()) {
        QMessageBox::information(this, "Información","No hay menús registrados.");
```

```
void MainWindow::actualizarMenu() {       //PARA MOSTRAR TODOS LOS MENUS
    vector<Menu> menus = cargarMenu(); // cargar desde archivo
    QString idTexto = ui->digitar_id->text();
    bool encontrado = false;

    if (idTexto.isEmpty()) {
        QMessageBox::warning(this, "Advertencia", "Ingrese el ID del menú a actualizar.");
        return;
```

```
void MainWindow::eliminarMenu() {
    QString idTexto = ui->digitar_id->text();

    if (idTexto.isEmpty()) {
        QMessageBox::warning(this, "Advertencia",
                             "Ingrese el ID del menú a eliminar.");
        return;
```

ALCANCE DEL PROYECTO

EL SISTEMA PERMITE:

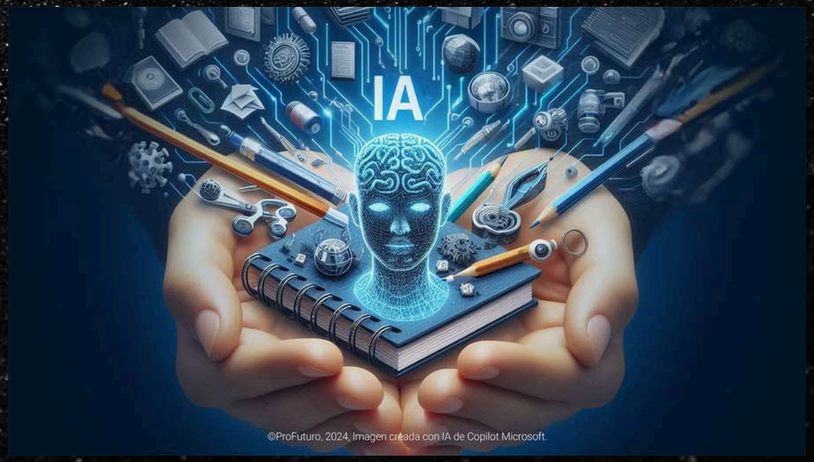
- Registrar nuevos cadena de restaurantes.
- Mostrar toda la cadena de restaurantes registrados.
- Modificar información de una cadena de restaurante existente.
- Eliminar cadena de restaurantes mediante su identificador.

TECNOLOGÍAS UTILIZADAS

LENGUAJE DE PROGRAMACIÓN: C++

ENTORNO DE DESARROLLO INTEGRADO

TECNOLOGIAS TIPO IA

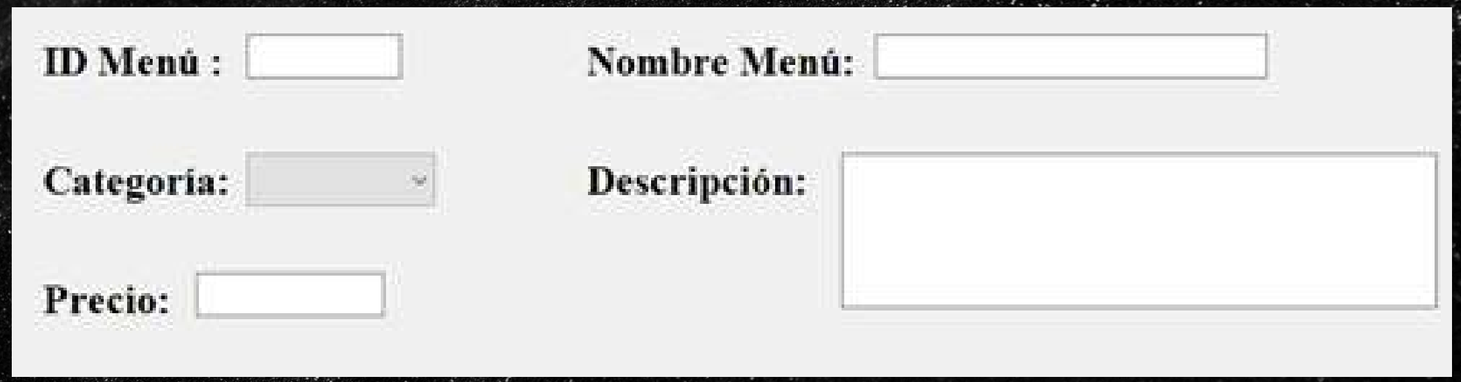


ESTRUCTURA DEL SISTEMA

ESTRUCTURA CADENA RESTAURANTE

LA ESTRUCTURA CADENA RESTAURANTE REPRESENTA CADA ELEMENTO DEL SISTEMA

ID: IDENTIFICADOR ÚNICO DEL CADENA DE RESTAURANTE
NOMBRE: NOMBRE DEL PLATILLO O PRODUCTO
DESCRIPCION: DESCRIPCIÓN DEL CADENA DE RESTAURANTE
CATEGORIA: TIPO DE CADENA DE RESTAURANTE (BEBIDA, COMIDA, POSTRE, ETC)
PRECIO: COSTO DEL CADENA DE RESTAURANTE



ESTA ESTRUCTURA INCLUYE UNA FUNCIÓN PARA MOSTRAR LA INFORMACIÓN POR PANTALLA.

FUNCIONALIDADES DEL CRUD

CREAR (CREATE)

PERMITE REGISTRAR UNA NUEVA CADENA DE RESTAURANTE SOLICITANDO LOS DATOS AL USUARIO Y ALMACENÁNDOLOS EN UN VECTOR

```
Menu agregarMenu() {
    Menu nuevoMenu;
    cout << "Ingrese el ID del menú: ";
    cin >> nuevoMenu.id;

    for (const auto& menu : listaMenus) {
        if (nuevoMenu.id == menu.id) {
            cout << "\nEl menú ya existe (ID duplicado).\n";
            return {};
        }
    }
    cin.ignore();
    cout << "Ingrese el nombre del menú: ";
    getline(cin, nuevoMenu.nombre);
    cout << "Ingrese la descripción del menú: ";
    getline(cin, nuevoMenu.descripcion);
    cout << "Ingrese la categoría del menú: ";
    getline(cin, nuevoMenu.categoria);
    cout << "Ingrese el precio del menú: ";
    cin >> nuevoMenu.precio;
    cout<<"\nMenu Registrado Con Éxito.\n";
    return nuevoMenu;
}
```

LEER (READ)

MUESTRA TODA LA CADENA DE RESTAURANTES REGISTRADOS EN EL SISTEMA RECORRIENDO EL VECTOR

```
void leerMenu() {
    if (listaMenus.empty()) {
        cout << "\nNo hay menús registrados\n";
    } else {
        cout << "\n*=*=*=*=*=*=*=*=*=* LISTA DE MENÚS REGISTRADOS *=*=*=*=*=*=*=*=*=*\n";
        for (const auto& m : listaMenus) {
            cout <<"ID: "<< m.id << "\nNombre: " << m.nombre << "\nDescripción: "
                 << m.descripcion << "\nCategoria: " << m.categoria
                 << "\nPrecio: $" << m.precio << endl;
        }
    }
}
```

ACTUALIZAR (UPDATE) PERMITE MODIFICAR LOS DATOS DE UNA CADENA DE RESTAURANTE EXISTENTE UTILIZANDO EL ID COMO REFERENCIA

```
void actualizar() {
    vector<Menu> menus = cargarMenu(); // cargar desde archivo
    int idBuscar;
    bool encontrado = false;

    cout << "\nIngrese el ID del menú a actualizar: ";
    cin >> idBuscar;
    cin.ignore();

    for (Menu &m : menus) {
        if (m.id == idBuscar) {
            cout << "\n=== MENÚ ENCONTRADO ===\n";
            cout << "ID: " << m.id << endl;
            cout << "Nombre actual: " << m.nombre << endl;
            cout << "Descripción actual: " << m.descripcion << endl;
            cout << "Categoría actual: " << m.categoria << endl;
            cout << "Precio actual: $" << m.precio << endl;

            cout << "\nIngrese el nuevo nombre: ";
            getline(cin, m.nombre);

            cout << "Ingrese la nueva descripción: ";
            getline(cin, m.descripcion);

            cout << "Ingrese la nueva categoría: ";
            getline(cin, m.categoria);

            cout << "Ingrese el nuevo precio: ";
            cin >> m.precio;

            encontrado = true;
            break;
        }
    }

    if (!encontrado) {
        cout << "\nNo se encontró un menú con ese ID\n";
        return;
    }

    // Reescribir archivo con datos actualizados
    ofstream archivo(ruta);
    if (!archivo) {
        cerr << "\nError al abrir el archivo para actualizar.\n";
        return;
    }

    for (const auto& m : menus) {
        archivo << m.id << "|"
                << m.nombre << "|"
                << m.descripcion << "|"
                << m.categoria << "|"
                << m.precio << "\n";
    }

    cout << "\nMENÚ ACTUALIZADO CORRECTAMENTE\n";
```

6.4 ELIMINAR (DELETE) ELIMINA UNA CADENA DE RESTAURANTE DEL SISTEMA MEDIANTE SU ID. ACTUALIZANDO EL VECTOR

```
void eliminarMenu() {
    vector<Menu> menus = cargarMenu();
    int idEliminar;
    bool encontrado = false;

    cout << "\nIngrese el ID del menú a eliminar: ";
    cin >> idEliminar;

    auto it = remove_if(menus.begin(), menus.end(),
                        [idEliminar](const Menu& m) {
                            return m.id == idEliminar;
                        });

    if (it != menus.end()) {
        menus.erase(it, menus.end());
        encontrado = true;
    }

    if (!encontrado) {
        cout << "\nNo se encontró un menú con ese ID\n";
        return;
    }

    ofstream archivo(ruta);
    if (!archivo) {
        cerr << "\nError al abrir el archivo para eliminar\n";
        return;
    }

    for (const auto& m : menus) {
        archivo << m.id << "|"
                << m.nombre << "|"
                << m.descripcion << "|"
                << m.categoria << "|"
                << m.precio << "\n";
    }

    listaMenus = menus;
    cout << "\nMENÚ ELIMINADO CORRECTAMENTE\n";
}
```

FLUJO DEL PROGRAMA

- Mostrar cadena de restaurante principal.
- Solicitar opción al usuario.
- Ejecutar la operación seleccionada.
- Retornar a la cadena de restaurante principal hasta que el usuario decida salir.

VALIDACIONES IMPLEMENTADAS

- Verificación de ID existente.
- Control de opciones inválidas de la cadena de restaurante.
- Prevención de accesos fuera de rango del vector.

POSIBLES MODIFICACIONES Y MEJORAS

1

MODIFICACIONES SIMPLES

- Cambiar nombres de variables.
- Ajustar categorías de la cadena de restaurante.
- Modificar el formato de impresión.
- Cambiar el tipo de dato del precio

2

MEJORAS INTERMEDIAS

- Implementar búsqueda por nombre o categoría.
- Ordenar cadena de restaurantes por precio o nombre.

3

MEJORAS AVANZADAS

- Implementar clases y programación orientada a objetos.
- Conectar con una base de datos.
- Implementar roles de usuario

CONCLUSIÓN

El proyecto CRUD de Cadenas de CadenaRestaurantes en C++ cumple con los objetivos planteados, permitiendo aplicar conceptos fundamentales de programación. Además, sirve como base para proyectos más complejos, ya que puede ser ampliado con nuevas funcionalidades y mejoras.


GRACIAS