Universidad Interamericana

Recinto de Arecibo

Proyecto Final-restaurantes

Joseph J. Rodríguez Hernández

Yeriel Mejías

Juan Pérez

COMP3400 Software Engineering

Javier Dasta Méndez

Tabla de contenido:

Resumen:

El trabajo estaba lleno de problemas con la base de datos y NGNIX. La parte de html funcionaba completamente pero la base de datos no quería conectarse a visual studio o datagrip. Nginix simplemente no quería funcionar tampoco por algún fallo o no quería acceder al port.

Objetivos del proyecto:

- Proveer una pagina html con base de datos a la cual se le pueda añadir mas datos desde un botón
- Pueda correr fluidamente y sin errores
- Sea simple de usar

Lista de miembros y rol

- Joseph Rodríguez- base de datos
- Yeriel mejías- html y ayuda en base de datos
- Juan Pérez- NGNIX

Producto log

Base de datos funcional

Conexión a NGNIX

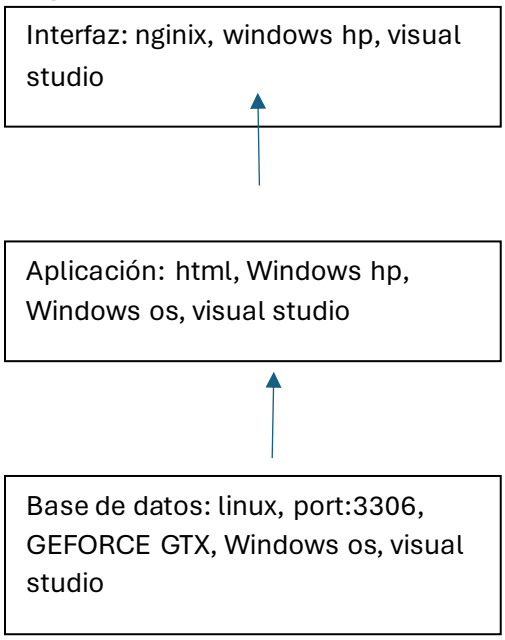
Corrección de errores entre html y base de datos y ngnix

WBS – división de tareas

- Joseph Rodriguez- trabajar con la base de datos y hacer que funcione con el código
- Yeriel mejias- trabajar con el código html y revisar que conecte con la base de datos
- Juan perez- gestionar GNIX para permitir que nuestra aplicación pueda salir en browsers como Firefox

Diseños del proyecto

- Diagrama de las arquitectura
- Diagrama ER



○ · “Screen Shoot” de las funcionalidades de la aplicación (aplicación

corriendo). No habrá Screen shot por que no corría la aplicación a pesar de los varios cambios y intentos

Retrospectiva del trabajo.

■ o Preguntas:

● § ¿Qué salió bien? Por qué

Joseph- El código html porque ya había experiencia con esto

Yeriel- lo único que salió bien fue el código html que hice

Juan- el html pero eso era normal

● § ¿Qué salió mal? Por qué

Joseph- la base de datos porque el command prompt estaba dando problemas

Yeriel- tratando de hacer la base de datos. Razón: Intenté hacerlo en visual studio code.

Juan- A la hora de tratar de que el código NGINIX funcionará junto a lo demás del proyecto tenía problemas para el servidor corriera.

● § ¿Qué aprendió?

Joseph- como activar una base de datos

Yeriel- Aprendí hacer el html más rapido con CSS.

Juan- Aprendí, cómo se debería de hacer un proyecto, mucho más ágil y en relación a los servidores.

● § ¿Qué requiere fortalecer?

Joseph- como trabajar con las bases de datos afuera de command prompt

Yeriel- Debo fortalecer en hacer base de datos fuera de VSC.

Juan- Debo de fortalecer, mucho más la aplicación de NGINIX el cómo abrir puertos y cómo poder implementarlo en desarrollo.

● § ¿Qué beneficios pudiera aportar la metodología AGILE

o SCRUM en el proyecto?

Joseph- nada

Yeriel- Yo no tuve ningun beneficio son los dos

Juan- no

● § ¿Cree que debería haber alguien que distribuya el

trabajo en su grupo? Por qué

Joseph- no, creo que seria mejor ponernos de acuerdo con que cosa hace cada uno

Yeriel- no sé.

Juan- no, es mejor preguntar a los otras que escogen

código

main.py

```
# Cursor para ejecutar consultas SQL
```

```
cursor = mariadb()

# Función para mostrar todas las mesas disponibles
def mostrar_mesas_disponibles():
    query = "SELECT * FROM Mesas"
    cursor.execute(query)
    mesas = cursor.fetchall()
    print("Mesas Disponibles:")
    for mesa in mesas:
        print(f"Número: {mesa[1]}, Capacidad: {mesa[2]}")

# Función para agregar un nuevo cliente
def agregar_cliente(nombre, telefono):
    query = "INSERT INTO Clientes (nombre, telefono) VALUES (%s, %s)"
    cursor.execute(query, (nombre, telefono))
    db_connection.commit()
    print("Cliente agregado exitosamente.")

# Función para realizar un pedido
def realizar_pedido(mesa_id, cliente_id, detalle_pedido):
    query = "INSERT INTO Pedidos (mesa_id, cliente_id, detalle_pedido) VALUES
(%s, %s, %s)"
    cursor.execute(query, (mesa_id, cliente_id, detalle_pedido))
    db_connection.commit()
    print("Pedido realizado con éxito.")

# Función principal
def main():
    print("Bienvenido al Sistema de Administración de Restaurantes")

    while True:
        print("\nOpciones:")
        print("1. Mostrar mesas disponibles")
        print("2. Agregar nuevo cliente")
        print("3. Realizar pedido")
        print("4. Salir")

        opcion = input("Seleccione una opción: ")

        if opcion == "1":
            mostrar_mesas_disponibles()
        elif opcion == "2":
            nombre = input("Ingrese el nombre del cliente: ")
            telefono = input("Ingrese el teléfono del cliente: ")
            agregar_cliente(nombre, telefono)
```

```
        elif opcion == "3":
            mesa_id = input("Ingrese el ID de la mesa: ")
            cliente_id = input("Ingrese el ID del cliente: ")
            detalle_pedido = input("Ingrese los detalles del pedido: ")
            realizar_pedido(mesa_id, cliente_id, detalle_pedido)
        elif opcion == "4":
            print("¡Hasta luego!")
            break
        else:
            print("Opción no válida. Por favor, seleccione una opción válida.")

if _name_ == "_main_":
    main()
```

app.py

```
import mariadb

# Conexión a la base de datos
db_connection = mariadb(
    host="localhost",
    user="ymejias3879@linux-vm10",
    password="r00593879",
    database="Restaurante"
)

# Cursor para ejecutar consultas SQL
cursor = mariadb()

# Función para mostrar todas las mesas disponibles
def mostrar_mesas_disponibles():
    query = "SELECT * FROM Mesas"
    cursor.execute(query)
    mesas = cursor.fetchall()
    print("Mesas Disponibles:")
    for mesa in mesas:
        print(f"Número: {mesa[1]}, Capacidad: {mesa[2]}")

# Función para agregar un nuevo cliente
def agregar_cliente(nombre, telefono):
    query = "INSERT INTO Clientes (nombre, telefono) VALUES (%s, %s)"
    cursor.execute(query, (nombre, telefono))
    db_connection.commit()
    print("Cliente agregado exitosamente.")
```

```
# Función para realizar un pedido
def realizar_pedido(mesa_id, cliente_id, detalle_pedido):
    query = "INSERT INTO Pedidos (mesa_id, cliente_id, detalle_pedido) VALUES
(%s, %s, %s)"
    cursor.execute(query, (mesa_id, cliente_id, detalle_pedido))
    db_connection.commit()
    print("Pedido realizado con éxito.")

# Función principal
def main():
    print("Bienvenido al Sistema de Administración de Restaurantes")

    while True:
        print("\nOpciones:")
        print("1. Mostrar mesas disponibles")
        print("2. Agregar nuevo cliente")
        print("3. Realizar pedido")
        print("4. Salir")

        opcion = input("Seleccione una opción: ")

        if opcion == "1":
            mostrar_mesas_disponibles()
        elif opcion == "2":
            nombre = input("Ingrese el nombre del cliente: ")
            telefono = input("Ingrese el teléfono del cliente: ")
            agregar_cliente(nombre, telefono)
        elif opcion == "3":
            mesa_id = input("Ingrese el ID de la mesa: ")
            cliente_id = input("Ingrese el ID del cliente: ")
            detalle_pedido = input("Ingrese los detalles del pedido: ")
            realizar_pedido(mesa_id, cliente_id, detalle_pedido)
        elif opcion == "4":
            print("¡Hasta luego!")
            break
        else:
            print("Opción no válida. Por favor, seleccione una opción válida.")

if _name_ == "_main_":
    main()
```

base_datos

```
-- Creación de la base de datos
```

```sql
CREATE DATABASE IF NOT EXISTS Restaurante;

-- Uso de la base de datos
USE Restaurante;

-- Creación de la tabla para las mesas
CREATE TABLE IF NOT EXISTS Mesas (
    id INT AUTO_INCREMENT PRIMARY KEY,
    numero INT NOT NULL,
    capacidad INT NOT NULL
);

-- Creación de la tabla para los clientes
CREATE TABLE IF NOT EXISTS Clientes (
    id INT AUTO_INCREMENT PRIMARY KEY,
    nombre VARCHAR(100) NOT NULL,
    telefono VARCHAR(20),
    email VARCHAR(100)
);

-- Creación de la tabla para los pedidos
CREATE TABLE IF NOT EXISTS Pedidos (
    id INT AUTO_INCREMENT PRIMARY KEY,
    mesa_id INT,
    cliente_id INT,
    fecha TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    FOREIGN KEY (mesa_id) REFERENCES Mesas(id),
    FOREIGN KEY (cliente_id) REFERENCES Clientes(id)
);

-- Creación de la tabla para los detalles del pedido
CREATE TABLE IF NOT EXISTS DetallesPedido (
    id INT AUTO_INCREMENT PRIMARY KEY,
    pedido_id INT,
    producto VARCHAR(100) NOT NULL,
    cantidad INT NOT NULL,
    precio DECIMAL(10, 2) NOT NULL,
    FOREIGN KEY (pedido_id) REFERENCES Pedidos(id)
);
```

users

```javascript
$(document).ready(function () {
    let crudAction = "edit";
```

```
    var table = $("#tablaUsuarios").DataTable({
      ajax: {
        url: "http://linux-vm10:5000/api/get_users", // Cambia esto por la URL de
tu API
      },
      columns: [
        { data: "Mesas" },
        { data: "Nombre" },
        { data: "Apellido" },
        { data: "Clientes" },
        { data: "Pedidos" },
        {
          data: null,
          defaultContent:
            "<div class='text-center'>" +
            "<button type='button' class='btn btn-info btn-sm btnEdit'><i
class='bi bi-pencil'></i></button> " +
            "<button type='button' class='btn btn-danger btn-sm btnDelete'><i
class='bi bi-trash3'></i></button>" +
            "</div>",
        },
      ],
    });

    $("#btn-add-new").click(function () {
      crudAction = "new";
      $("form").get(0).reset();

      $("#nombre").prop("readonly", false);
      $("#apellido").prop("readonly", false);
      $("#email").prop("readonly", false);
      $("#contrasena").prop("readonly", false);
      $("#password").prop("readonly", false);
      $("#rol").prop("readonly", false);

      updateModal("Añadir Usuario");
    });

    $("#tablaUsuarios tbody").on("click", ".btnEdit", function () {
      var data = table.row($(this).parents("tr")).data(); // Get ROW DATA in
TABLE

      setDataElementsReadOnly(data, false);
      fillForm(data);
```

```
      crudAction = "edit";

      // $("#modalTitle").text("Editar Usuario");
      // $("#modalBody").text(
      //   ¿Deseas editar al usuario ${data.nombre} ${data.apellido}?
      // );
      updateModal("Editar Usuario");
    });

    $("#tablaUsuarios tbody").on("click", ".btnDelete", function () {
      var data = table.row($(this).parents("tr")).data(); // Get ROW DATA in
TABLE

      setDataElementsReadOnly(data, true);
      fillForm(data);
      crudAction = "delete";
      $("#modalTitle").text("Eliminar Usuario");
      $("#modalMessages").html(
        '<div class="alert alert-danger" role="alert">' +
          "¿Estás segur@ que quieres eliminar al usuario <strong>${data.nombre}
${data.apellido}</strong>?" +
          "</div>"
      );

      $("#btn-modal-crud-action").addClass("btn-danger");
      $("#btn-modal-crud-action").removeClass("btn-primary");
      $("#btn-modal-crud-action").text("Borrar");

      $("#actionModal").modal("show");
    });

    $("#btn-modal-crud-action").click(function () {
      // alert(crudAction);
      if (crudAction === "new") {
        // Execute REST API to INSERT
      } else if (crudAction === "edit") {
        // Execute REST API to EDIT
      } else {
        // if crudAction === "delete";
        // Execute REST API to DELETE
      }
      $("#actionModal").modal("hide");
    });

    function updateModal(title) {
```

```
      $("#modalTitle").text(title);

      $("#btn-modal-crud-action").addClass("btn-primary");
      $("#btn-modal-crud-action").removeClass("btn-danger");
      $("#btn-modal-crud-action").text("Guardar");

      $("#actionModal").modal("show");
    }
  });
```

Project-tools

```
function fillForm(formData) {
    $.each(formData, function (key, valor) {
      var elemento = $('[name="' + key + '"]');
      if (elemento.length > 0) {
        if (elemento.is(":checkbox") || elemento.is(":radio")) {
          elemento.prop("checked", elemento.val() === valor);
        } else if (elemento.is("select")) {
          elemento.val(valor);
        } else {
          elemento.val(valor);
        }
      }
    });
  }

  function setDataElementsReadOnly(formData, isReadOnly) {
    $.each(formData, function (key) {
      var elemento = $('[name="' + key + '"]');
      elemento.prop("readonly", isReadOnly ? "readonly" : "");
    });
  }
```

NGINIX

Código NGINIX.

server {

  listen 80;

  server_name tu_dominio.com; # Reemplaza con tu dominio

```
    location / {
        proxy_pass http://127.0.0.1:8000;
        proxy_set_header Host $host;
        proxy_set_header X-Real-IP $remote_addr;
        proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
        proxy_set_header X-Forwarded-Proto $scheme;
    }
}
```

NGINIX de Flask app.py

```
import mysql.connector
from flask import Flask, jsonify

app = Flask(_name_)

# Configuración de la conexión a la base de datos
db_config = {
    'user': 'tu_usuario',  # Reemplaza con tu usuario de MariaDB
    'password': 'tu_contraseña',  # Reemplaza con tu contraseña de MariaDB
    'host': 'localhost',
    'database': 'tu_base_de_datos'  # Reemplaza con el nombre de tu base de datos
}

@app.route('/')
def index():
    connection = mysql.connector.connect(**db_config)
    cursor = connection.cursor()
```

```
    cursor.execute("SELECT '¡Bienvenido a la Administración de Restaurantes!'")
    result = cursor.fetchone()
    connection.close()
    return jsonify(result)

if _name_ == "_main_":
    app.run()
```

Para abrir el puerto 8000.

```
server {
    listen 80;
    server_name tu_dominio.com;

    location / {
        proxy_pass http://unix:/ruta/a/tu/aplicacion/gunicorn.sock;
        proxy_set_header Host $host;
        proxy_set_header X-Real-IP $remote_addr;
        proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
        proxy_set_header X-Forwarded-Proto $scheme;
    }
}
```

Link a github

https://github.com/pepebotellas67/restaurante_base_de_datos