**Informe de Desarrollo del Proyecto Gastrobar**

**1. Configuración del Proyecto**

* **Archivo .env**: Configuramos las variables de entorno para la conexión a la base de datos PostgreSQL:

port=8080

DB\_HOST=localhost

DB\_PORT=5432

DB\_USER=postgres

DB\_PASSWORD=123456

DB\_NAME=Gastrobar

**2. Modelos Implementados**

Se definieron las estructuras necesarias para manejar el sistema de usuarios, menú, pedidos, mesas e inventario:

* **User**: Representa a los usuarios del sistema (Dueño, Admin y Empleados).

type User struct {

    ID       int    `json:"id"`

    Name     string `json:"name"`

    Email    string `json:"email"`

    Username string `json:"username"`

    Password string `json:"-"`    // No se incluye en la respuesta JSON por seguridad

    Role     string `json:"role"` // Puede ser "Dueño", "Admin", o "Employee"

}

**ItemMenu**: Representa los ítems del menú (comida y bebidas).

type ItemMenu struct {

    ID        int     `json:"id"`

    Nombre    string  `json:"nombre"`

    Categoria string  `json:"categoria"` // "comida" o "bebida"

    Precio    float64 `json:"precio"`

    Stock     int     `json:"stock"`

}

**Pedido**: Representa un pedido realizado por un cliente.

type Pedido struct {

    ID       int            `json:"id"`

    MesaID   int            `json:"mesa\_id"`

    Elementos []ElementoPedido `json:"elementos"`

    Total    float64        `json:"total"`

    Pagado   bool           `json:"pagado"`

}

**ElementoPedido**: Representa un ítem dentro de un pedido.

type ElementoPedido struct {

    ItemMenuID int `json:"item\_menu\_id"`

    Cantidad   int `json:"cantidad"`

}

**Mesa**: Representa una mesa en el restaurante.

type Mesa struct {

    ID     int    `json:"id"`

    Estado string `json:"estado"` // "libre" o "ocupada"

}

**3. Controladores Implementados**

Se desarrollaron las siguientes funciones en los controladores:

* **Usuarios**:
  + GetUsers: Obtiene todos los usuarios registrados.
  + CreateUser: Crea un nuevo usuario.
* **Menú**:
  + [GetMenu](vscode-file://vscode-app/c:/Users/Miraflores/AppData/Local/Programs/Microsoft%20VS%20Code/resources/app/out/vs/code/electron-sandbox/workbench/workbench.html): Obtiene el menú completo (comida y bebidas).
* **Pedidos**:
  + [CreateOrder](vscode-file://vscode-app/c:/Users/Miraflores/AppData/Local/Programs/Microsoft%20VS%20Code/resources/app/out/vs/code/electron-sandbox/workbench/workbench.html): Crea un pedido, valida el número de la mesa, verifica el stock de los ítems y descuenta del inventario.
* **Historial de Ventas**:
  + [GetSalesHistory](vscode-file://vscode-app/c:/Users/Miraflores/AppData/Local/Programs/Microsoft%20VS%20Code/resources/app/out/vs/code/electron-sandbox/workbench/workbench.html): Obtiene el historial de ventas, incluyendo los pedidos realizados.

**Rutas Configuradas**

Se definieron las rutas necesarias para interactuar con el sistema:

* **Usuarios**:
  + GET /users: Obtiene todos los usuarios.
  + POST /users: Crea un nuevo usuario.
* **Menú**:
  + GET /menu: Obtiene el menú completo.
* **Pedidos**:
  + POST /orders: Crea un nuevo pedido.
* **Historial de Ventas**:
  + GET /sales-history: Obtiene el historial de ventas

**Base de Datos**

Se diseñaron las tablas necesarias para almacenar los datos del sistema:

* **users**: Almacena los datos de los usuarios (Dueño, Admin y Empleados).

CREATE TABLE users (

    id SERIAL PRIMARY KEY,

    name VARCHAR(100) NOT NULL,

    email VARCHAR(100) UNIQUE NOT NULL,

    username VARCHAR(50) UNIQUE NOT NULL,

    password VARCHAR(255) NOT NULL,

    role VARCHAR(20) NOT NULL CHECK (role IN ('Dueño', 'Admin', 'Employee'))

);

**menu\_items**: Almacena los ítems del menú (comida y bebidas).

CREATE TABLE menu\_items (

    id SERIAL PRIMARY KEY,

    nombre VARCHAR(100) NOT NULL,

    categoria VARCHAR(50) NOT NULL CHECK (categoria IN ('comida', 'bebida')),

    precio NUMERIC(10, 2) NOT NULL,

    stock INT NOT NULL CHECK (stock >= 0)

);

**tables**: Almacena las mesas disponibles en el restaurante.

CREATE TABLE tables (

    id SERIAL PRIMARY KEY,

    estado VARCHAR(50) DEFAULT 'libre' CHECK (estado IN ('libre', 'ocupada'))

);

**orders**: Almacena los pedidos realizados.

CREATE TABLE orders (

    id SERIAL PRIMARY KEY,

    table\_id INT NOT NULL REFERENCES tables(id),

    total NUMERIC(10, 2) NOT NULL,

    paid BOOLEAN DEFAULT FALSE

);

**order\_items**: Almacena los ítems específicos dentro de cada pedido.

CREATE TABLE order\_items (

    id SERIAL PRIMARY KEY,

    order\_id INT NOT NULL REFERENCES orders(id) ON DELETE CASCADE,

    menu\_item\_id INT NOT NULL REFERENCES menu\_items(id),

    cantidad INT NOT NULL CHECK (cantidad > 0)

);

**. Datos Iniciales**

Se insertaron datos iniciales para probar el sistema:

* **Usuarios**:

INSERT INTO users (name, email, username, password, role) VALUES

('Juan Pérez', 'juan@example.com', 'juan', 'hashed\_password', 'Dueño'),

('Ana López', 'ana@example.com', 'ana', 'hashed\_password', 'Admin'),

('Carlos Gómez', 'carlos@example.com', 'carlos', 'hashed\_password', 'Employee');

* **Ítems del Menú**:

INSERT INTO menu\_items (nombre, categoria, precio, stock) VALUES

('Hamburguesa', 'comida', NULL, 50),

('Perro', 'comida', NULL, 50),

('Pasta', 'comida', NULL, 50),

('Plato del día', 'comida', NULL, 50),

('Coca-Cola', 'bebida', NULL, 100),

('Manzana', 'bebida', NULL, 100),

('Bretaña', 'bebida', NULL, 100),

('Hatsu Rosa', 'bebida', NULL, 100),

('Uva', 'bebida', NULL, 100),

('Cuatro', 'bebida', NULL, 100),

('Sprite', 'bebida', NULL, 100),

('Canada Dry', 'bebida', NULL, 100);

* **Mesas**:

INSERT INTO tables (id, estado) VALUES

(1, 'libre'),

(2, 'libre'),

(3, 'libre'),

(4, 'libre');

**7. Validaciones Implementadas**

* **Número de Mesa**:
  + Se valida que el número de la mesa esté entre 1 y 4.
* **Stock**:
  + Se verifica que haya suficiente stock antes de procesar un pedido.
* **Descuento del Inventario**:
  + El stock de los ítems se reduce automáticamente al realizar un pedido.

**8. Próximos Pasos**

* Agregar precios a los ítems del menú.
* Implementar autenticación para los usuarios.
* Crear un frontend para interactuar con las rutas del backend.

**Fin del Informe**