Hospital Centro Ginecológico - HCG

TERCER AÑO DE BACHILLERATO
DESARROLLO DE SOFTWARE

Manual técnico - Proyecto Web – Versión 3.0

Integrantes:

Luis Eduardo Alvarenga Claros #20220298

Francisco Daniel García Ramos #20220326

Rodrigo Alejandro Alvarado Rodríguez #20220495

Francisco Javier Guadrón Vásquez #20220332

Jeremy Leandro Castro Moreno #20220638

Instituto Técnico Ricaldone

2 de octubre del 2024

ÍNDICE DE TEMAS

INTRODUCCIÓN

Este manual técnico está diseñado para ofrecer una descripción clara y completa del proyecto web desarrollado, brindando una visión general de sus componentes clave. Comienza con una presentación de las tecnologías y herramientas utilizadas, proporcionando un resumen de los lenguajes de programación, frameworks y plataformas que sustentan el proyecto, tales como PHP, MySQL, JavaScript, HTML, CSS, y otros recursos técnicos. Posteriormente, se adentra en la estructura de la base de datos y el diccionario de datos, explicando cómo se organizan y gestionan los datos dentro del sistema, con detalles sobre las tablas, sus campos y las relaciones entre ellas.

La arquitectura del software se describe para mostrar cómo se estructura el código y cómo interactúan los distintos módulos del sistema, asegurando una visión integral del funcionamiento del proyecto. También se explora la estructura del proyecto, donde se detalla la organización de archivos y carpetas, con el objetivo de facilitar la comprensión y el mantenimiento del código. Se destacan elementos importantes como los controllers, resources y views, esenciales para la gestión del backend y la presentación del frontend.

El manual incluye una sección dedicada al diseño de la aplicación, que pone énfasis en la usabilidad y la estética, asegurando que la experiencia del usuario sea intuitiva y atractiva. Además, se incluye un apartado sobre buenas prácticas de desarrollo, ofreciendo recomendaciones útiles para escribir un código eficiente, seguro y mantenible.Por último, se especifican los requerimientos de hardware y software, describiendo lo que se necesita para implementar el proyecto correctamente, seguido de una sección detallada que guía a los usuarios a través de los pasos para la instalación y configuración del sistema.

Este documento está dirigido tanto a desarrolladores como a usuarios técnicos, brindándoles una guía completa que les permitirá comprender el funcionamiento del proyecto, así como desplegar e implementar el sistema de manera efectiva y sin contratiempos.

TECNOLOGÍAS Y HERRAMIENTAS UTILIZADAS.

Servidor Web:

- **Debian v5.10.223-1**

  Sistema operativo de código abierto basado en Linux, utilizado como servidor web para garantizar un entorno seguro y estable en el desarrollo de aplicaciones web.

  Sitio web oficial: https://www.debian.org

- **XAMPP v3.3.0**

  XAMPP es una distribución que incluye Apache, MySQL, PHP y Perl, utilizada como servidor local para pruebas y desarrollo en entornos locales antes de pasar a producción.

  Sitio web oficial: https://www.apachefriends.org

Lenguajes de Programación (Lado del cliente):

- **JavaScript v20.11.1**

  Lenguaje de programación que permite crear interactividad dentro de las páginas web. Utilizado para manejar eventos, validaciones y comunicación con el servidor a través de peticiones asíncronas (AJAX).

  Sitio web oficial: https://www.javascript.com

- **HTML 5**

  Lenguaje estándar para la creación de páginas web y su estructura básica. HTML5 es la versión más reciente que soporta multimedia y otras tecnologías modernas.

  Sitio web oficial: https://html.spec.whatwg.org

- **CSS**

  Hojas de estilo en cascada utilizadas para definir la apariencia y el diseño de las páginas web.

  Sitio web oficial: https://www.w3.org/Style/CSS/

Lenguajes de Programación (Lado del Servidor):

- **PHP v7.4.33**

  PHP es un lenguaje de programación del lado del servidor que se utiliza principalmente para el

desarrollo web. Se encarga de procesar la lógica del servidor, generar contenido dinámico y comunicarse con la base de datos.

Sitio web oficial: https://www.php.net

Base de Datos:

- **MySQL v8.0.37**

  Sistema de gestión de bases de datos relacional (RDBMS) de código abierto. MySQL permite almacenar y gestionar grandes volúmenes de datos para aplicaciones web.

  Sitio web oficial: https://www.mysql.com

Frameworks y Librerías:

- **Bootstrap v5.3.0**

  Framework front-end que facilita el desarrollo de aplicaciones web responsivas mediante el uso de componentes predefinidos para diseño y estructura.

  Sitio web oficial: https://getbootstrap.com

- **SweetAlert v2**

  Librería de JavaScript que permite crear alertas personalizadas y atractivas en las interfaces web.

  Sitio web oficial: https://sweetalert2.github.io

Entorno de Desarrollo:

- **Visual Studio Code v1.93.1**

  Editor de código fuente gratuito y multiplataforma, utilizado ampliamente para el desarrollo de aplicaciones web. Cuenta con numerosas extensiones que facilitan la programación.

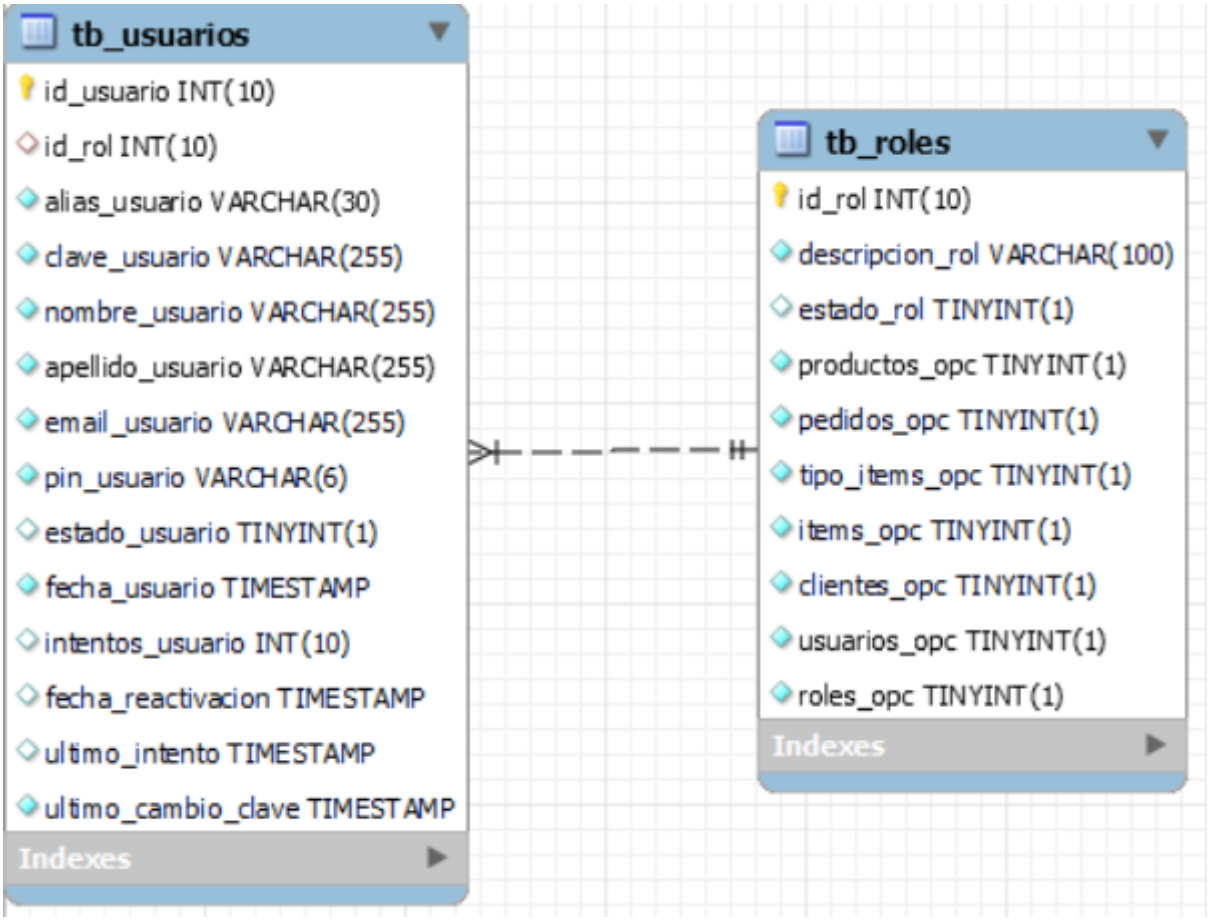
  Sitio web oficial: https://code.visualstudio.com

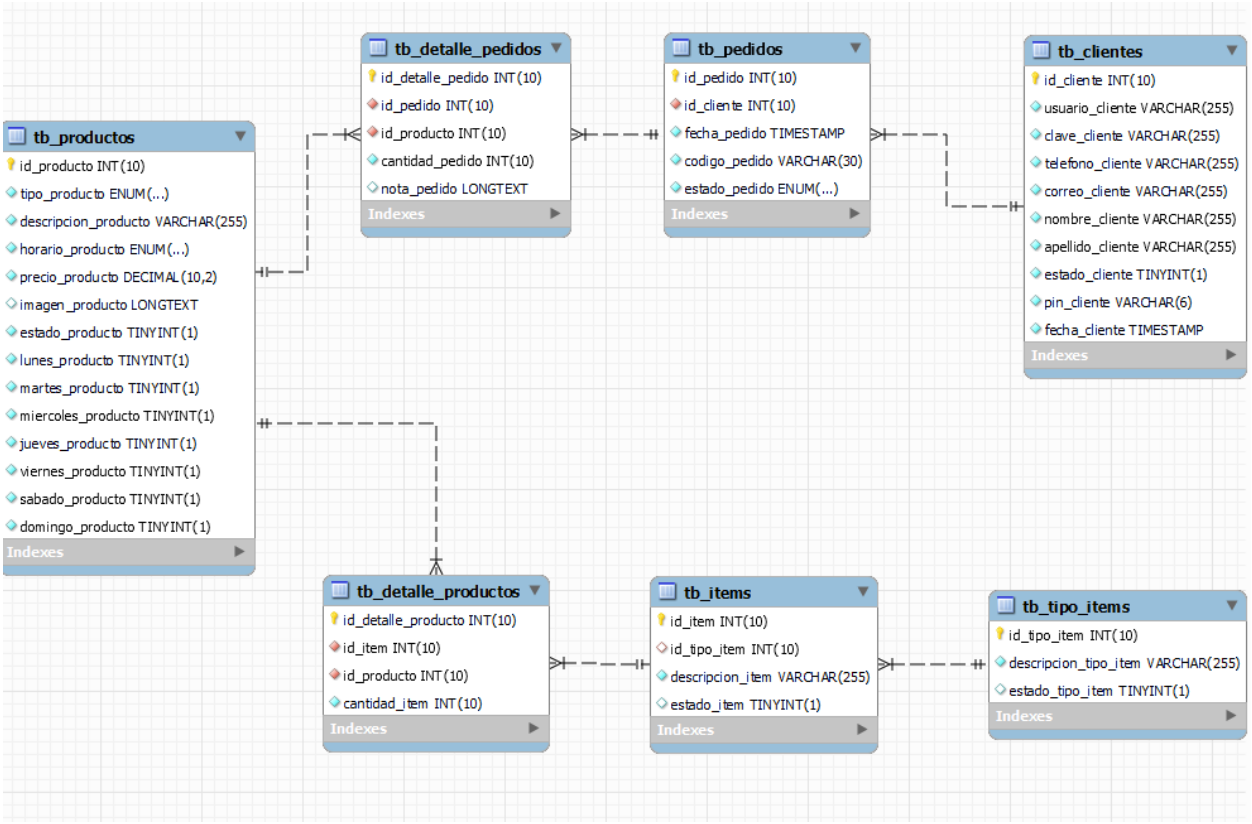
- **MySQL Workbench v8.0**

  Herramienta de diseño de bases de datos visual utilizada para modelar, diseñar, y administrar bases de datos MySQL.

  Sitio web oficial: https://www.mysql.com/products/workbench/

ESTRUCTURA DE LA BASE DE DATOS.





SCRIPT SQL

```sql
DROP DATABASE IF EXISTS db_hcg;
CREATE DATABASE IF NOT EXISTS db_hcg;
USE db_hcg;

CREATE TABLE tb_roles(
  id_rol INT UNSIGNED,
  descripcion_rol VARCHAR(100) NOT NULL,
  estado_rol BOOLEAN DEFAULT TRUE,
  productos_opc BOOLEAN NOT NULL,
  pedidos_opc BOOLEAN NOT NULL,
  tipo_items_opc BOOLEAN NOT NULL,
  items_opc BOOLEAN NOT NULL,
  clientes_opc BOOLEAN NOT NULL,
  usuarios_opc BOOLEAN NOT NULL,
  roles_opc BOOLEAN NOT NULL,
  PRIMARY KEY (id_rol)
);

CREATE TABLE tb_usuarios(
  id_usuario INT UNSIGNED,
  id_rol INT UNSIGNED,
```

```
  alias_usuario VARCHAR(30) NOT NULL,
  clave_usuario VARCHAR(255) NOT NULL,
  nombre_usuario VARCHAR(255) NOT NULL,
  apellido_usuario VARCHAR(255) NOT NULL,
  email_usuario VARCHAR(255) NOT NULL,
  pin_usuario VARCHAR(6) NOT NULL,
  estado_usuario BOOLEAN DEFAULT TRUE,
  fecha_usuario TIMESTAMP DEFAULT CURRENT_TIMESTAMP NOT NULL,
  intentos_usuario INT UNSIGNED DEFAULT 0,
  fecha_reactivacion TIMESTAMP NULL DEFAULT NULL,
  ultimo_intento TIMESTAMP NULL DEFAULT NULL,
  ultimo_cambio_clave TIMESTAMP DEFAULT CURRENT_TIMESTAMP NOT NULL,
  factor_autenticacion boolean default false,
  PRIMARY KEY (id_usuario),
  CONSTRAINT alias_usuario UNIQUE (alias_usuario),
  CONSTRAINT fk_usuario_rol
  FOREIGN KEY(id_rol) REFERENCES tb_roles(id_rol)
);


CREATE TABLE tb_clientes(
  id_cliente INT UNSIGNED,
  usuario_cliente VARCHAR(255) NOT NULL,
  clave_cliente VARCHAR(255) NOT NULL,
  telefono_cliente VARCHAR(255) NOT NULL,
  correo_cliente VARCHAR(255) NOT NULL,
  nombre_cliente VARCHAR(255) NOT NULL,
  apellido_cliente VARCHAR(255) NOT NULL,
  estado_cliente BOOLEAN DEFAULT TRUE NOT NULL,
  pin_cliente VARCHAR(6) NOT NULL DEFAULT '000000',
  fecha_cliente TIMESTAMP DEFAULT CURRENT_TIMESTAMP NOT NULL,
  PRIMARY KEY (id_cliente),
  CONSTRAINT usuario_unique UNIQUE (usuario_cliente),
  CONSTRAINT telefono_unique UNIQUE (telefono_cliente),
  CONSTRAINT correo_unique UNIQUE (correo_cliente)
);

CREATE TABLE tb_tipo_items(
  id_tipo_item INT UNSIGNED,
  descripcion_tipo_item VARCHAR(255) NOT NULL,
  estado_tipo_item BOOLEAN DEFAULT TRUE,
  PRIMARY KEY (id_tipo_item)
);

CREATE TABLE tb_items(
```

```
  id_item INT UNSIGNED,
  id_tipo_item INT UNSIGNED ,
  descripcion_item VARCHAR(255) NOT NULL,
  estado_item BOOLEAN DEFAULT TRUE,
  PRIMARY KEY (id_item),
  CONSTRAINT fk_item_tipo
  FOREIGN KEY(id_tipo_item) REFERENCES tb_tipo_items(id_tipo_item)
);
CREATE TABLE tb_productos(
  id_producto INT UNSIGNED,
  tipo_producto ENUM('Conjunto','Complementario') NOT NULL,
  descripcion_producto VARCHAR(255) NOT NULL,
  horario_producto ENUM('Desayuno','Almuerzo','Cena','Típico','Todo el
día','Desayuno y Almuerzo',
  'Desayuno y Cena','Almuerzo y Cena','Típico y Desayuno','Típico y
Almuerzo','Tipico y Cena') NOT NULL,
  precio_producto DECIMAL(10, 2) UNSIGNED NOT NULL,
  imagen_producto LONGTEXT,
  estado_producto BOOLEAN NOT NULL,
  lunes_producto BOOLEAN NOT NULL,
  martes_producto BOOLEAN NOT NULL,
  miercoles_producto BOOLEAN NOT NULL,
  jueves_producto BOOLEAN NOT NULL,
  viernes_producto BOOLEAN NOT NULL,
  sabado_producto BOOLEAN NOT NULL,
  domingo_producto BOOLEAN NOT NULL,
  PRIMARY KEY (id_producto)
);
CREATE TABLE tb_detalle_productos(
  id_detalle_producto INT UNSIGNED /*auto_increment*/,
  id_item INT UNSIGNED  NOT NULL,
  id_producto INT UNSIGNED  NOT NULL, /*TRAE EL ID DEL producto*/
  cantidad_item INT UNSIGNED NOT NULL,
  PRIMARY KEY (id_detalle_producto),
  CONSTRAINT fk_item_producto_item /*LLAVE FORANEA*/
  FOREIGN KEY(id_item) REFERENCES tb_items(id_item)
  ON DELETE CASCADE ON UPDATE CASCADE,
  CONSTRAINT fk_item_producto_producto /*LLAVE FORANEA*/
  FOREIGN KEY(id_producto) REFERENCES tb_productos(id_producto) ON DELETE CASCADE
ON UPDATE CASCADE
);
CREATE TABLE tb_pedidos(
    id_pedido INT UNSIGNED NOT NULL,
    id_cliente INT UNSIGNED  NOT NULL,
    fecha_pedido TIMESTAMP DEFAULT CURRENT_TIMESTAMP NOT NULL,
```

```
    codigo_pedido VARCHAR(30) NOT NULL,
    estado_pedido enum('No escogido','Pendiente','Finalizado') NOT NULL,
    PRIMARY KEY (id_pedido),
    CONSTRAINT fk_pedido_cliente
    FOREIGN KEY(id_cliente) REFERENCES tb_clientes(id_cliente) ON DELETE CASCADE
ON UPDATE CASCADE
);

CREATE TABLE tb_detalle_pedidos (
  id_detalle_pedido INT UNSIGNED ,
  id_pedido INT UNSIGNED NOT NULL,
  id_producto INT UNSIGNED NOT NULL,
  cantidad_pedido INT UNSIGNED NOT
NULL,
  nota_pedido LONGTEXT,
  PRIMARY KEY (id_detalle_pedido),
  FOREIGN KEY (id_pedido) REFERENCES tb_pedidos(id_pedido) ON DELETE CASCADE ON
UPDATE CASCADE,
  FOREIGN KEY (id_producto) REFERENCES tb_productos(id_producto) ON DELETE
CASCADE ON UPDATE CASCADE
);
```

SCRIPT DDL

```
/*FUNCION QUE ME DEVUELVE EL ID SIGUIENTE QUE SE REGISTRARIA EN ORDEN Y SE MANDA
A QUE TABLA SE AGARRARIA
ESTO SE USA EN LUGAR DEL AUTO_INCREMENT*/
DELIMITER //
CREATE FUNCTION get_next_id(table_name VARCHAR(255))
RETURNS INT UNSIGNED
BEGIN
    DECLARE next_id INT UNSIGNED;

    CASE table_name
        WHEN 'tb_roles' THEN
            SELECT IFNULL(MAX(id_rol), 0) + 1 INTO next_id FROM tb_roles;
        WHEN 'tb_usuarios' THEN
            SELECT IFNULL(MAX(id_usuario), 0) + 1 INTO next_id FROM tb_usuarios;
        WHEN 'tb_clientes' THEN
```

```
            SELECT IFNULL(MAX(id_cliente), 0) + 1 INTO next_id FROM tb_clientes;
        WHEN 'tb_tipo_items' THEN
            SELECT IFNULL(MAX(id_tipo_item), 0) + 1 INTO next_id FROM
tb_tipo_items;
        WHEN 'tb_items' THEN
            SELECT IFNULL(MAX(id_item), 0) + 1 INTO next_id FROM tb_items;
        WHEN 'tb_productos' THEN
            SELECT IFNULL(MAX(id_producto), 0) + 1 INTO next_id FROM
tb_productos;
        WHEN 'tb_detalle_productos' THEN
            SELECT IFNULL(MAX(id_detalle_producto), 0) + 1 INTO next_id FROM
tb_detalle_productos;
        WHEN 'tb_detalle_pedidos' THEN
            SELECT IFNULL(MAX(id_detalle_pedido), 0) + 1 INTO next_id FROM
tb_detalle_pedidos;
        WHEN 'tb_pedidos' THEN
            SELECT IFNULL(MAX(id_pedido), 0) + 1 INTO next_id FROM tb_pedidos;
    END CASE;

    RETURN next_id;
END
//DELIMITER ;

/*FUNCION QUE ME DEVUELVE UN CODIGO  DE 6 DIGITOS EL CUAL VERIFICA QUE NO SE
REPITA EN LA FECHA ACTUAL*/
DELIMITER //
CREATE FUNCTION generar_codigo() RETURNS VARCHAR(6)
BEGIN
    DECLARE codigo VARCHAR(6);
    SET codigo = '';
    REPEAT
        SET codigo = '';
        WHILE LENGTH(codigo) < 6 DO
            SET codigo = CONCAT(codigo,
SUBSTRING('ABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789', RAND() * 62 + 1, 1));
        END WHILE;
    UNTIL NOT EXISTS (SELECT 1 FROM tb_pedidos WHERE DATE(fecha_pedido) =
CURDATE() AND codigo_pedido = codigo) END REPEAT;
    RETURN codigo;
END//
DELIMITER ;

/*PROCEDIMIENTO ALMACENADO PARA INSERTAR UN PRODUCTO*/
DELIMITER //
```

```
CREATE PROCEDURE InsertarProducto(IN tipo_producto VARCHAR(255),IN
descripcion_producto VARCHAR(255),IN horario_producto VARCHAR(255),
IN precio_producto DECIMAL(10,2),IN imagen_producto VARCHAR(255),IN
estado_producto BOOLEAN,IN lunes_producto BOOLEAN,IN martes_producto BOOLEAN,
IN miercoles_producto BOOLEAN,IN jueves_producto BOOLEAN,IN viernes_producto
BOOLEAN,IN sabado_producto BOOLEAN,IN domingo_producto BOOLEAN
)
BEGIN
    INSERT INTO tb_productos
(id_producto,tipo_producto,descripcion_producto,horario_producto,precio_producto,
imagen_producto,estado_producto,
        lunes_producto,martes_producto,miercoles_producto,jueves_producto,viernes
_producto,sabado_producto,domingo_producto
    )
    VALUES (
        (SELECT
get_next_id("tb_productos")),tipo_producto,descripcion_producto,horario_producto,
precio_producto,imagen_producto,
        estado_producto,lunes_producto,martes_producto,miercoles_producto,jueves_
producto,viernes_producto,sabado_producto,domingo_producto
    );
END //
DELIMITER ;

/*VERIFICA QUE PRODUCTO EL CUAL SE VA A ESOGER ESTE DENTRO DEL HORARIO Y DIA
DISPONIBLE*/
DELIMITER //
CREATE TRIGGER before_detalle_pedido_insert
BEFORE INSERT ON tb_detalle_pedidos
FOR EACH ROW
BEGIN
    DECLARE v_disponible BOOLEAN;

    -- Verificar disponibilidad del producto en el horario y día correspondiente
    SELECT COUNT(*)
    INTO v_disponible
    FROM tb_productos
    WHERE estado_producto = 1 AND tipo_producto = "Conjunto"
    AND (
        (DAYOFWEEK(NOW()) = 1 AND domingo_producto = 1)
        OR (DAYOFWEEK(NOW()) = 2 AND lunes_producto = 1)
        OR (DAYOFWEEK(NOW()) = 3 AND martes_producto = 1)
        OR (DAYOFWEEK(NOW()) = 4 AND miercoles_producto = 1)
        OR (DAYOFWEEK(NOW()) = 5 AND jueves_producto = 1)
        OR (DAYOFWEEK(NOW()) = 6 AND viernes_producto = 1)
```

```
        OR (DAYOFWEEK(NOW()) = 7 AND sabado_producto = 1)
    )
    AND (
        (horario_producto = "Desayuno" AND TIME(NOW()) BETWEEN "06:00:00" AND
"11:00:00")
        OR (horario_producto = "Almuerzo" AND TIME(NOW()) BETWEEN "11:00:00" AND
"15:00:00")
        OR (horario_producto = "Típico" AND TIME(NOW()) BETWEEN "15:00:00" AND
"18:00:00")
        OR (horario_producto = "Cena" AND TIME(NOW()) BETWEEN "18:00:00" AND
"22:00:00")
        OR (horario_producto = "Todo el día")
        OR (horario_producto = "Desayuno y Almuerzo" AND (TIME(NOW()) BETWEEN
"06:00:00" AND "11:00:00" OR TIME(NOW()) BETWEEN "11:00:00" AND "15:00:00"))
        OR (horario_producto = "Desayuno y Cena" AND (TIME(NOW()) BETWEEN
"06:00:00" AND "11:00:00" OR TIME(NOW()) BETWEEN "18:00:00" AND "22:00:00"))
        OR (horario_producto = "Almuerzo y Cena" AND (TIME(NOW()) BETWEEN
"11:00:00" AND "15:00:00" OR TIME(NOW()) BETWEEN "18:00:00" AND "22:00:00"))
        OR (horario_producto = "Típico y Desayuno" AND (TIME(NOW()) BETWEEN
"06:00:00" AND "10:00:00" OR TIME(NOW()) BETWEEN "15:00:00" AND "18:00:00"))
        OR (horario_producto = "Típico y Almuerzo" AND (TIME(NOW()) BETWEEN
"11:00:00" AND "15:00:00" OR TIME(NOW()) BETWEEN "18:00:00" AND "22:00:00"))
        OR (horario_producto = "Típico y Cena" AND (TIME(NOW()) BETWEEN
"06:00:00" AND "10:00:00" OR TIME(NOW()) BETWEEN "15:00:00" AND "18:00:00"))
    );

    IF v_disponible = 0 THEN
        SIGNAL SQLSTATE '45000'
        SET MESSAGE_TEXT = 'El producto solicitado no está disponible en este
horario';
    END IF;
END;
//DELIMITER ;

/*PROCEDIMIENTO QUE ELIMINA UN PRODUCTO DEL CARRITO SI NO ESTA EN EL HORARIO*/
DELIMITER //
CREATE PROCEDURE sp_verificar_y_obtener_detalles_pedido()
BEGIN
    DECLARE v_disponible INT;
    DECLARE v_id_detalle_pedido INT;
    DECLARE v_id_producto INT;
    DECLARE done INT DEFAULT 0;
    -- Cursor para iterar sobre los detalles de pedidos pendientes
    DECLARE cur_detalle CURSOR FOR
        SELECT dp.id_detalle_pedido, dp.id_producto
```

```
        FROM tb_detalle_pedidos dp
        JOIN tb_pedidos p ON dp.id_pedido = p.id_pedido
        WHERE p.estado_pedido = 'No escogido';

    -- Handler para salir del loop
    DECLARE CONTINUE HANDLER FOR NOT FOUND SET done = 1;
    -- Abrir el cursor
    OPEN cur_detalle;
    -- Bucle para iterar sobre los detalles de pedidos pendientes
    read_loop: LOOP
        FETCH cur_detalle INTO v_id_detalle_pedido, v_id_producto;
         -- Si no hay más filas, salir del bucle
        IF done THEN
            LEAVE read_loop;
        END IF;
        -- Verificar si el producto está disponible en el horario actual
        SELECT COUNT(*)
        INTO v_disponible
        FROM tb_productos
        WHERE id_producto = v_id_producto
        AND estado_producto = 1
        AND (
            (DAYOFWEEK(NOW()) = 1 AND domingo_producto = 1)
            OR (DAYOFWEEK(NOW()) = 2 AND lunes_producto = 1)
            OR (DAYOFWEEK(NOW()) = 3 AND martes_producto = 1)
            OR (DAYOFWEEK(NOW()) = 4 AND miercoles_producto = 1)
            OR (DAYOFWEEK(NOW()) = 5 AND jueves_producto = 1)
            OR (DAYOFWEEK(NOW()) = 6 AND viernes_producto = 1)
            OR (DAYOFWEEK(NOW()) = 7 AND sabado_producto = 1)
        )
        AND (
            (horario_producto = "Desayuno" AND TIME(NOW()) BETWEEN "06:00:00" AND
"11:00:00")
            OR (horario_producto = "Almuerzo" AND TIME(NOW()) BETWEEN "11:00:00"
AND "15:00:00")
            OR (horario_producto = "Típico" AND TIME(NOW()) BETWEEN "15:00:00"
AND "18:00:00")
            OR (horario_producto = "Cena" AND TIME(NOW()) BETWEEN "18:00:00" AND
"22:00:00")
            OR (horario_producto = "Todo el día")
            OR (horario_producto = "Desayuno y Almuerzo" AND (TIME(NOW()) BETWEEN
"06:00:00" AND "11:00:00" OR TIME(NOW()) BETWEEN "11:00:00" AND "15:00:00"))
            OR (horario_producto = "Desayuno y Cena" AND (TIME(NOW()) BETWEEN
"06:00:00" AND "11:00:00" OR TIME(NOW()) BETWEEN "18:00:00" AND "22:00:00"))
```

```
            OR (horario_producto = "Almuerzo y Cena" AND (TIME(NOW()) BETWEEN
"11:00:00" AND "15:00:00" OR TIME(NOW()) BETWEEN "18:00:00" AND "22:00:00"))
            OR (horario_producto = "Típico y Desayuno" AND (TIME(NOW()) BETWEEN
"06:00:00" AND "10:00:00" OR TIME(NOW()) BETWEEN "15:00:00" AND "18:00:00"))
            OR (horario_producto = "Típico y Almuerzo" AND (TIME(NOW()) BETWEEN
"11:00:00" AND "15:00:00" OR TIME(NOW()) BETWEEN "18:00:00" AND "22:00:00"))
            OR (horario_producto = "Típico y Cena" AND (TIME(NOW()) BETWEEN
"06:00:00" AND "10:00:00" OR TIME(NOW()) BETWEEN "15:00:00" AND "18:00:00"))
        );

        -- Si el producto no está disponible en el horario actual, eliminar el
detalle del pedido
        IF v_disponible = 0 THEN
            DELETE FROM tb_detalle_pedidos WHERE id_detalle_pedido =
v_id_detalle_pedido;
        END IF;
    END LOOP read_loop;

    -- Cerrar el cursor
    CLOSE cur_detalle;

    -- Devolver los detalles de pedidos pendientes
    SELECT dp.*
    FROM tb_detalle_pedidos dp
    JOIN tb_pedidos p ON dp.id_pedido = p.id_pedido
    WHERE p.estado_pedido = 'No escogido';
END//
DELIMITER ;

/*TRIGGER QUE VERIFICA QUE SI UNA NOTA ESTA VACIA SE AGREGUE ESE TEXTO DE NOTA
VACIA*/
DELIMITER //
CREATE TRIGGER before_insert_tb_detalle_pedidos
BEFORE INSERT ON tb_detalle_pedidos
FOR EACH ROW
BEGIN
  IF NEW.nota_pedido IS NULL OR NEW.nota_pedido = '' THEN
    SET NEW.nota_pedido = 'Nota vacía';
  END IF;
END; //
DELIMITER ;
```

DICCIONARIO DE DATOS.

tb_roles

Llave	Campo	Tipo de Dato	Tamaño	Restricción	Descripción
PK	id_rol	INT UNSIGNED	11	AUTO_INCREMENT	Identificador único del rol
	descripcion_rol	VARCHAR	100	NOT NULL	Descripción del rol
	estado_rol	BOOLEAN	True y false	DEFAULT TRUE	Estado del rol (activo/inactivo)
	pedidos_opc	BOOLEAN	True y false	NOT NULL	Opción de acceso a pedidos
	tipo_items_opc	BOOLEAN	True y false	NOT NULL	Opción de acceso a tipos de items
	items_opc	BOOLEAN	True y false	NOT NULL	Opción de acceso a items
	productos_opc	BOOLEAN	True y false	NOT NULL	Opción de acceso a productos
	clientes_opc	BOOLEAN	True y false	NOT NULL	Opción de acceso a clientes
	usuarios_opc	BOOLEAN	True y false	NOT NULL	Opción de acceso a usuarios
	roles_opc	BOOLEAN	True y false	NOT NULL	Opción de acceso a roles

tb_usuarios

Llave	Campo	Tipo de Dato	Tamaño	Restricción	Descripción
PK	id_usuario	INT UNSIGNED	11	Auto Increment	Identificador único del usuario

FK	id_rol	INT UNSIGNED	11	ON DELETE CASCADE	Llave foránea que referencia a tb_roles
	alias_usuario	VARCHAR	30	UNIQUE, NOT NULL	Alias del usuario con el cual ingresara al sistema
	clave_usuario	VARCHAR	255	NOT NULL	Clave del usuario
	nombre_usuario	VARCHAR	255	NOT NULL	Nombre del usuario
	apellido_usuario	VARCHAR	255	NOT NULL	Apellido del usuario
	email_usuario	VARCHAR	255	NOT NULL	Correo electrónico del usuario
	pin_usuario	VARCHAR	6	NOT NULL	PIN de seguridad del usuario. Este se utilizará para la recuperación de contraseña (se actualizará cada vez que se use)
	estado_usuario	BOOLEAN	True y false	DEFAULT TRUE	Estado del usuario (activo/inactivo)
	fecha_usuario	TIMESTAMP		DEFAULT CURRENT_TIMESTAMP NOT NULL	Campo para guardar fecha del registro del usuario
	intentos_usuario	INT		UNSIGNED DEFAULT 0	Verificar intentos fallidos de inicio de sesión
	fecha_reactivacion	TIMESTAMP		NULL DEFAULT NULL	Fecha en que se reactivo por ultima vez la cuenta
	ultimo_intento	TIMESTAMP		NULL DEFAULT NULL	Verfiicar la fecha en que fue su ultimo intento
	ultimo_cambio_clave	TIMESTAMP		DEFAULT CURRENT_TIMESTAMP NOT NULL	Ultima fecha en que se cambió la clave del usuario
	factor_autenticacion	BOOLEAN	True y false	DEFAULT FALSE	Habilitar la doble autenticación

tb_clientes

Llave	Campo	Tipo de Dato	Tamaño	Restricción	Descripción
PK	id_cliente	INT UNSIGNED	11	Auto increment	Identificador único del cliente
	usuario_cliente	VARCHAR	255	UNIQUE	Alias del cliente
	clave_cliente	VARCHAR	255		Contraseña del cliente

	telefono_cliente	VARCHAR	255	NOT NULL	Número de teléfono del cliente
	nombre_cliente	VARCHAR	255	NOT NULL	Nombre del cliente
	apellido_cliente	VARCHAR	255	NOT NULL	Apellido del cliente
	estado_cliente	BOOLEAN		DEFAULT TRUE NOT NULL	Campo para manejar el estado del cliente
	pin_cliente	VARCHAR	6	NOT NULL DEFAULT 000000	Pin para recuperar clave
	fecha_cliente				Fecha de registro del cliente

tb_tipo_items

Llave	Campo	Tipo de Dato	Tamaño	Restricción	Descripción
PK	id_tipo_item	INT UNSIGNED	11	Auto increment	Identificador único del tipo de ítem
	descripcion_tipo_item	VARCHAR	255	NOT NULL	Descripción del tipo de ítem
	estado_tipo_item	BOOLEAN	True y false	DEFAULT TRUE	Estado del tipo de ítem, si esta disponible o no está en existencia.

tb_items

Llave	Campo	Tipo de Dato	Tamaño	Restricción	Descripción
PK	id_item	INT UNSIGNED	11	Auto increment	Identificador único del ítem
FK	id_tipo_item	INT UNSIGNED	11	ON DELETE CASCADE	Llave foránea que referencia a tb_tipo_items
	nombre_item	VARCHAR	255	NOT NULL	Nombre del ítem. Por ejemplo: papas lays, una coca-cola, etc.
	estado_item	BOOLEAN	True y false	DEFAULT TRUE	Estado del ítem

tb_productos

Llave	Campo	Tipo de Dato	Tamaño	Restricción	Descripción
PK	id_producto	INT UNSIGNED	11	Auto increment	Identificador único del producto
	tipo_producto	ENUM	' Conjunto ', 'Complementario'	NOT NULL	Tipo de producto
	descripcion_producto	VARCHAR	255	NOT NULL	Descripción del producto. Lo que incluye el plato.
	horario_producto	ENUM	'D','A','C','T','TD','D-A','D-C','A-C','T-D','TA','T-C'	NOT NULL	Horario de disponibilidad del producto
	precio_producto	DECIMAL	10, 2	NOT NULL	Precio del producto
	estado_producto	BOOLEAN	True y false	NOT NULL	Estado del producto
	lunes_producto	BOOLEAN	True y false	NOT NULL	Disponibilidad del producto los lunes
	martes_producto	BOOLEAN	True y false	NOT NULL	Disponibilidad del producto los martes

	miercoles_producto	BOOLEAN	True y false	NOT NULL	Disponibilidad del producto los miércoles
	jueves_producto	BOOLEAN	True y false	NOT NULL	Disponibilidad del producto los jueves
	viernes_producto	BOOLEAN	True y false	NOT NULL	Disponibilidad del producto los viernes
	sabado_producto	BOOLEAN	True y false	NOT NULL	Disponibilidad del producto los sábados
	domingo_producto	BOOLEAN	True y false	NOT NULL	Disponibilidad del producto los domingos

tb_detalle_productos

Llave	Campo	Tipo de Dato	Tamaño	Restricción	Descripción
PK	id_detalle_producto	INT UNSIGNED	11	AUTO_INCREMENT	Identificador único del detalle del producto
FK	id_item	INT UNSIGNED	11	NOT NULL	Identificador del ítem asociado al detalle
FK	id_producto	INT UNSIGNED	11	NOT NULL	Identificador del producto asociado al detalle
	cantidad_item	INT UNSIGNED	11	NOT NULL	Cantidad del ítem en el det

tb_pedidos

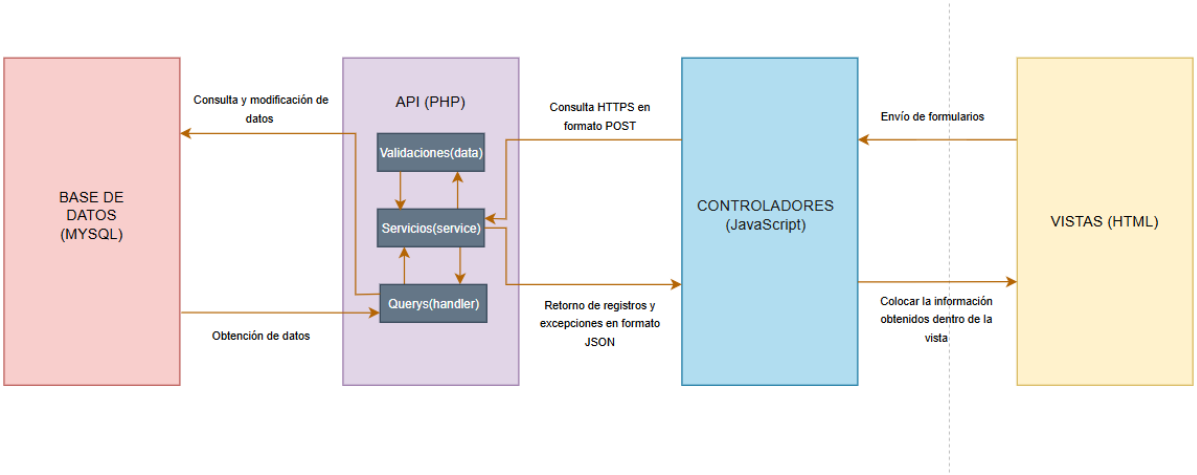
Llave	Campo	Tipo de Dato	Tamaño	Restricción	Descripción
PK	id_pedido	INT UNSIGNED	11	Auto increment	Identificador único del pedido

FK	id_cliente	INT UNSIGNED	11	ON DELETE CASCADE	Llave foránea que referencia a tb_clientes
	fecha_pedido	TIMESTAMP	32	DEFAULT CURRENT_TIMESTAMP	Fecha y hora del pedido
	estado_pedido	ENUM	'Pendiente','Entregado'	NOT NULL	Estado del pedido ('Pendiente' o 'Entregado')

tb_detalle_pedidos

Llave	Campo	Tipo de Dato	Tamaño	Restricción	Descripción
PK	id_detalle_pedido	INT UNSIGNED	11	Auto increment	Identificador único del detalle de pedido
FK	id_pedido	INT UNSIGNED	11	ON DELETE CASCADE	Llave foránea que referencia a tb_pedidos
FK	id_producto	INT UNSIGNED	11	ON DELETE CASCADE	Llave foránea que referencia a tb_productos
	cantidad_pedido	INT UNSIGNED	11	NOT NULL	Cantidad del producto en el pedido

ARQUITECTURA DEL SOFTWARE.



Base de Datos (MySQL)

En la base de datos se almacenan y gestionan todos los datos de la aplicación. Aquí se realizan las operaciones de consulta, inserción, actualización y eliminación de datos. La base de datos está estructurada para soportar las operaciones de negocio necesarias para la aplicación, garantizando integridad y seguridad en la gestión de la información. Esta capa es consultada y modificada por la API a través de querys específicas que utilizan SQL.

API (PHP)

La API actúa como el intermediario entre la base de datos y los controladores. Está dividida en varias subcapas:

- **Validaciones:** Se encarga de comprobar que los datos enviados desde los controladores cumplen con los formatos y reglas definidas antes de realizar cualquier operación sobre los datos.
- **Servicios:** Contienen la lógica de negocio de la aplicación. Aquí se definen las reglas y operaciones que deben realizarse para cumplir con los requisitos funcionales del sistema.
- **Querys:** Gestionan las interacciones directas con la base de datos. A través de este componente, la API puede obtener datos, insertar nuevos registros o actualizar los existentes. Esta capa devuelve la información en formato JSON para que pueda ser procesada por los controladores.

Controladores (JavaScript):

Los controladores son responsables de gestionar las solicitudes enviadas desde la vista (interfaz de usuario) y enviar las respuestas correspondientes. Los controladores se comunican con la API utilizando solicitudes HTTP en formato POST para obtener datos o enviar formularios. Una vez que reciben una respuesta en formato JSON, se encargan de procesar la información y determinar qué acciones tomar. Si la operación fue exitosa, los controladores actualizan la vista con la nueva información; si hubo un error, lo muestran de manera adecuada al usuario.

Vistas (HTML)

Las vistas son las encargadas de mostrar la interfaz gráfica de la aplicación. En esta capa, el usuario puede interactuar con los formularios, botones y demás componentes visuales. Los controladores colocan la información obtenida desde la API dentro de la vista para que el usuario final la visualice. A su vez, las vistas envían los formularios al controlador cuando se requiere realizar una acción, como ingresar datos o hacer una búsqueda.

Interacción

El flujo comienza cuando el usuario interactúa con las vistas, ya sea para solicitar información o enviar un formulario. Esta acción desencadena una llamada al controlador, el cual procesa la solicitud y se comunica con la API mediante una petición HTTPS en formato POST. La API, a su vez, realiza las validaciones pertinentes y ejecuta los servicios necesarios, que interactúan con la base de datos a través de consultas SQL gestionadas.

Una vez obtenida la respuesta desde la base de datos, la API retorna los resultados o excepciones en formato JSON al controlador, que finalmente envía estos datos a la vista, permitiendo que la información solicitada sea presentada de manera amigable para el usuario. Este ciclo se repite cada vez que se realiza una nueva interacción en la aplicación.

Esta estructura basada en el patrón MVC permite una clara separación de responsabilidades, donde cada componente tiene un rol definido: las vistas se encargan de la presentación, los controladores de la lógica de interacción, la API de la lógica de negocio y validación, y la base de datos de la persistencia de los datos.

Esta arquitectura modular facilita el mantenimiento, mejora la escalabilidad y asegura que los diferentes equipos de desarrollo puedan trabajar en paralelo en distintas capas sin interferencias.

ESTRUCTURA DEL PROYECTO.

RAÍZ DEL PROYECTO

- **ER.drawio**: Diagrama Entidad-Relación en formato de archivo de Draw.io, que probablemente detalla la estructura de la base de datos.
- **.htaccess**: Archivo de configuración para el servidor Apache.
- **Buenas practicas**: Parece ser un archivo o directorio relacionado con las buenas prácticas de desarrollo (revisar su contenido específico).
- **Estandares**: Posiblemente contiene normas o estándares que deben seguirse en el proyecto.
- **HCG.sql**, **HCG_DDL.sql**, **HCG_DML.sql**: Archivos SQL para la creación y gestión de la base de datos (DDL para definir estructuras, DML para manipulación de datos).
- **README.md**: Archivo de documentación general del proyecto.
- **ZDOCS**: Documentos relacionados con el diseño del proyecto.

API

En esta carpeta se encuentran los modelos de datos principales y archivos php para manejar la información consultada dentro de la base de datos.

- **Helpers**: Directorio con scripts de ayuda y utilidades.
- **Config.php**: Archivo de configuración general para la API.
- **Database.php**: Gestión de la conexión y operaciones con la base de datos.
- **Report.php**: Generación de reportes dentro de la API.
- **Validator.php**: Validación de datos enviados a la API.
- **Images**: Fondos e imágenes utilizados posiblemente en reportes o plantillas.
- **Libraries**: En esta subcarpeta se encuentran las librerías que ayudan para generar el pdf de cada reporte y el envío de correo electrónicos.

- **Models:** En esta subcarpeta se encuentran el código php para validar y asignar variables en los archivos data, para luego ser usados en las consultas SQL de los archivos handler. Existen un archivo data y handler por cada tabla de la base de datos.
- **Reports:** En esta carpeta se encuentran los diseños de los reportes del sitio público y privado.

CONTROLLERS

Esta carpeta contiene los archivos de javascript principales para manejar la información que viene del API , organizada en tres subcarpetas: **admin**, **public**, y **utils**.

- **Admin**: Controladores relacionados con la gestión administrativa.
- **Public**: Controladores de las vistas accesibles para los usuarios públicos.
- **Utils**: Aquí se encuentran plantillas como el navbar, footer y funciones reutilizables en toda la aplicación.

RESOURCES

Esta carpeta contiene los recursos estáticos como CSS, imágenes, fuentes, y scripts de JavaScript externos.

- **CSS**: Archivos CSS para el diseño visual de la aplicación y componentes reusables.
- **Error**: Aquí se encuentran las páginas de error personalizadas como 403.html y 404.html. Las cuales sirven cuando una url no existe dentro del proyecto o una página está restringida.
- **Fonts**: Contiene fuentes como bootstrap-icons.woff2 para íconos.
- **Img**: Imágenes utilizadas en la interfaz (íconos, fondos, y más).
- **JS:** Scripts de JavaScript externos como bootstrap.bundle.min.js, chart.umd.min.js, y sweetalert.min.js. Los cuales sirven para manipular ciertos elementos de html y darles un diseño desde JavaScript.

VIEWS

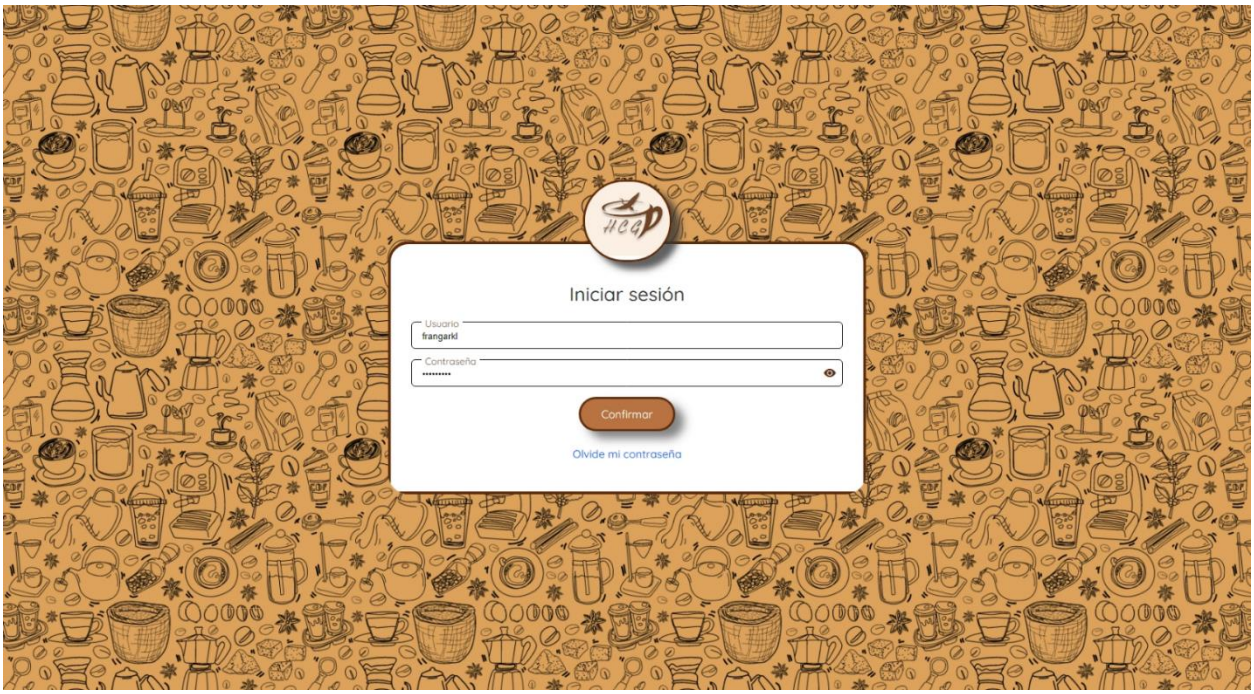
En esta carpeta se encuentran todas las pantallas y diseños en html con los que trabaja el proyecto web. El cual se divide en dos subcarpetas que son **admin** y **public**.

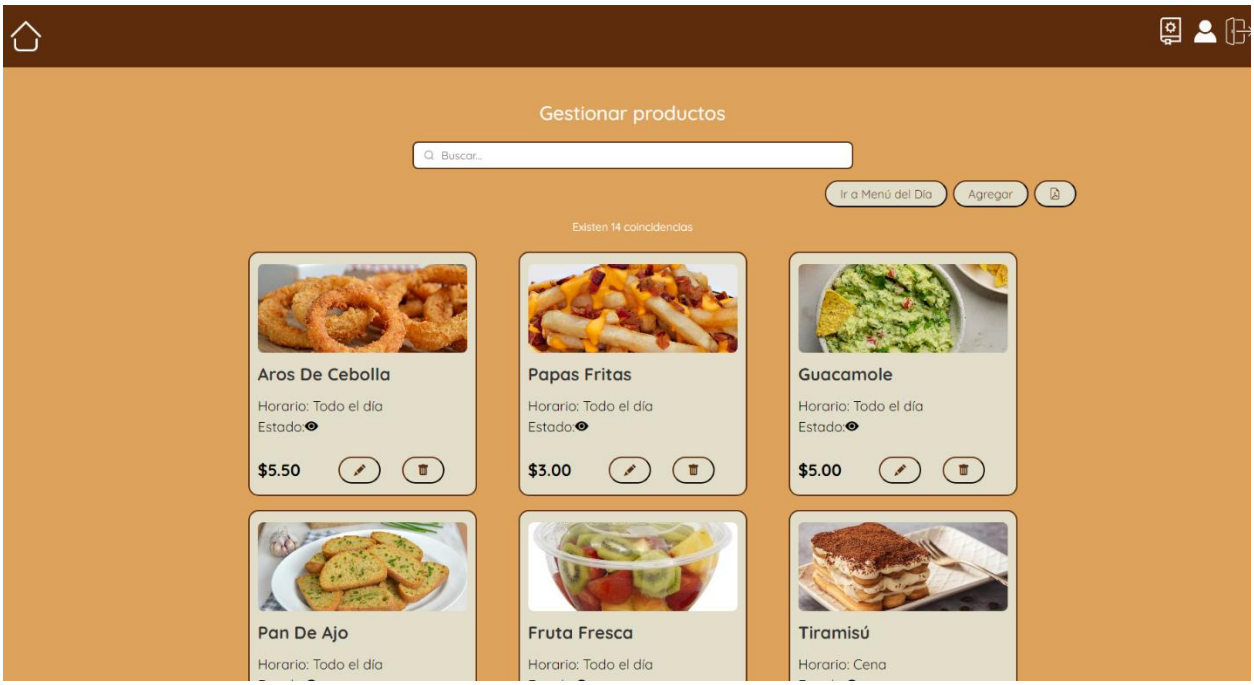
- **Admin**: Pantallas html usadas dentro del proyecto administrativo.
- **Public**: Pantallas html usadas dentro del proyecto público.

DISEÑO DE LA APLICACIÓN

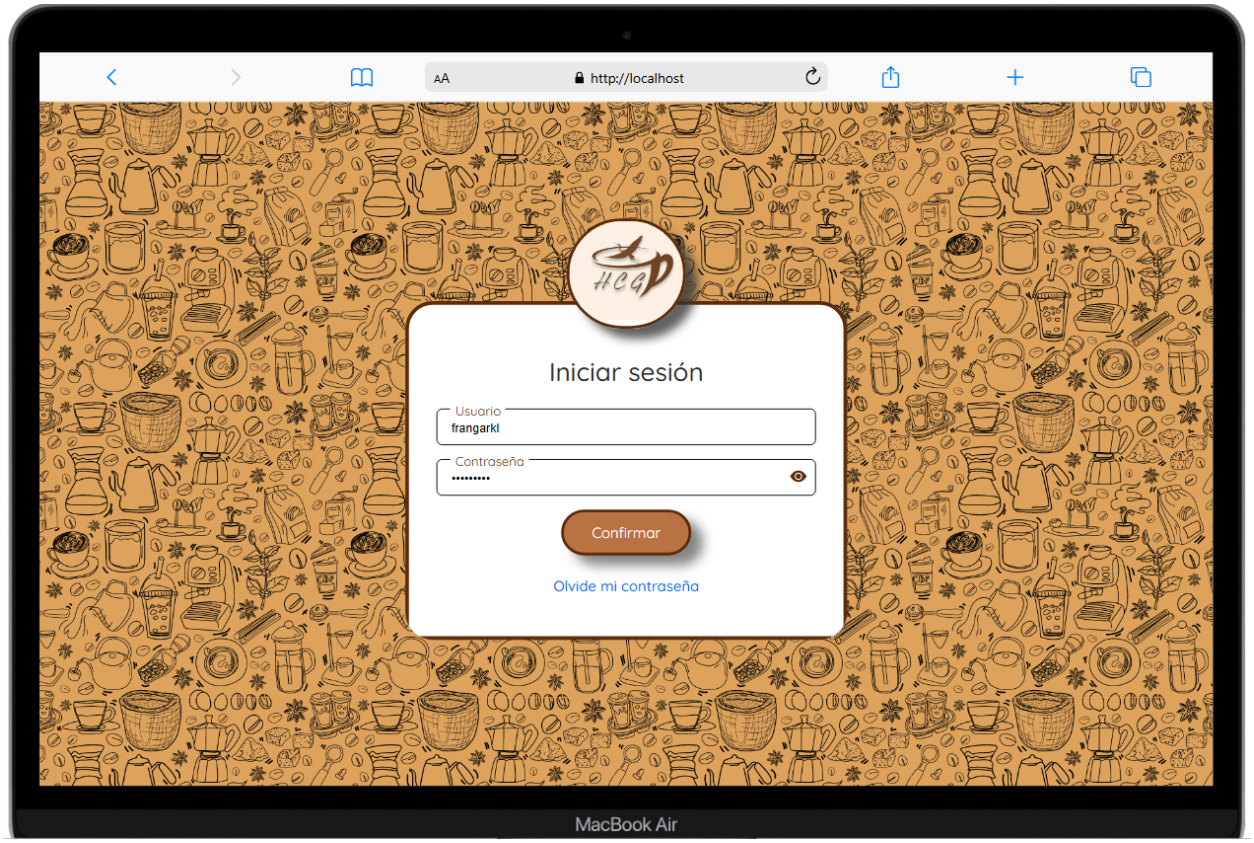
Paleta de colores:

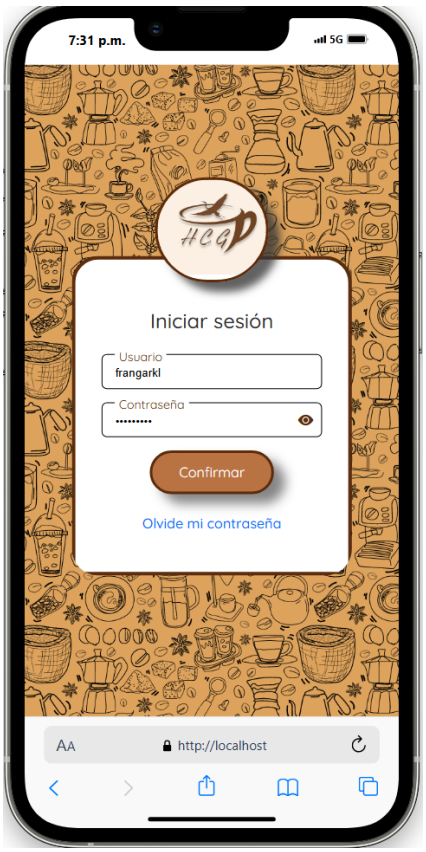
Fuente utilizada en las pantallas: QuickSand

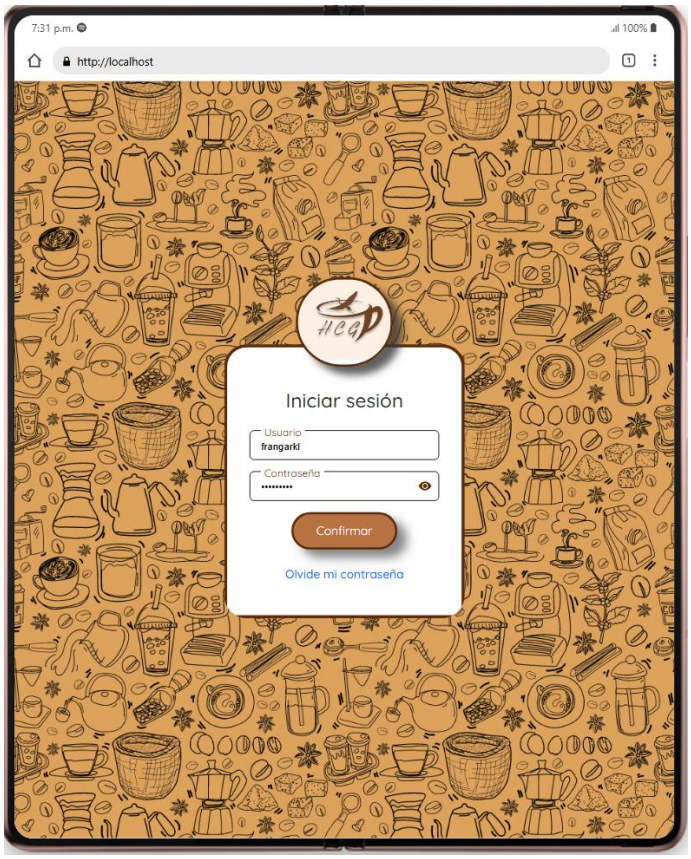




La aplicacion en difentes tamaños




7:31 p.m.
5G
Iniciar sesión
Usuario
frangarkl
Contraseña
Confirmar
Olvide mi contraseña
AA
http://localhost


7:31 p.m.
100%
http://localhost
Iniciar sesión
Usuario
frangarkl
Contraseña
Confirmar
Olvide mi contraseña

BUENAS PRÁCTICAS DE DESARROLLO

A la hora de programar utilizamos buenas prácticas en nuestro código para que sea más fácil de entender a la hora de ser revisado por otros programadores y también para que sea más limpio y que no genere problema.

Algunas de las buenas prácticas que poseemos son estas:

Escribir Código Claro y Legible:

Hemos usado nombres descriptivos para variables, funciones y clases.

Documentamos el propósito del código y explicamos las partes más complejas. Sin embargo, no abusamos de los comentarios; el código debe ser auto explicativo.

Modularidad y Reutilización:

Dividimos el código en funciones pequeñas que realizan una sola tarea (Single Responsibility Principle), lo que facilita la lectura y mantenimiento.

Evitamos la duplicación de código. Cuando identificamos patrones repetitivos, los extraemos en funciones o clases para mayor eficiencia.

Gestión de Errores y Excepciones:

Implementamos el manejo adecuado de errores con bloques try-catch o equivalentes según el lenguaje. Nos aseguramos de que las excepciones no oculten errores críticos.

Siempre validamos las entradas de usuarios o datos externos para prevenir errores o vulnerabilidades de seguridad.

Uso de Control de Versiones:

Hemos adoptado un sistema de control de versiones como Git. Realizamos commits pequeños y descriptivos, y trabajamos con ramas para desarrollar nuevas funcionalidades o corregir errores.

Escribimos mensajes de commit claros y concisos para facilitar el seguimiento de los cambios a lo largo del tiempo.

Documentación:

Nos aseguramos de que el código esté bien documentado, incluyendo instrucciones para su uso, configuración y dependencias necesarias.

Creamos archivos ReadMe para documentar cómo instalar, configurar y ejecutar el proyecto de manera clara.

Revisión de Código:

Participamos en revisiones de código, lo que no solo mejora la calidad del proyecto, sino que también fomenta el aprendizaje entre el equipo.

Utilizamos linters y herramientas de análisis estático para identificar posibles problemas en el código antes de la ejecución.

Seguridad:

Nunca exponemos credenciales, claves API u otros datos sensibles en el código. Utilizamos variables de entorno y sistemas de gestión de secretos.

Siempre sanitizamos las entradas para proteger la aplicación de ataques como inyección SQL o cross-site scripting (XSS).

También a la hora de programar utilizamos los estándares de programación adecuados para que el código sea entendible y evitar errores comunes:

Estándares de PHP:

- Sangría de 4 espacios: El código utiliza una sangría de 4 espacios para mejorar la legibilidad y organización del código. Esto es crucial para mantener la estructura clara, especialmente en bloques de control como `if`, `switch`, y `case`.

- Uso de CamelCase en nombres de métodos y variables: En el código, los nombres de métodos como `setNombre`, `setApellido`, `readExist` y las variables como `$clienteData`, siguen la convención CamelCase, lo que facilita la legibilidad y coherencia en la nomenclatura de las funciones y variables.

- Validación de entradas: Antes de procesar los datos, el código valida las entradas utilizando métodos como `Validator::validateForm($_POST)`. Esto es fundamental para prevenir ataques como la inyección SQL o la manipulación de datos inesperados.

- Uso de comentarios descriptivos: El código está bien documentado con comentarios que explican la funcionalidad de bloques específicos de código. Esto ayuda a otros desarrolladores a entender rápidamente el propósito de cada sección del script.

- Manejo de errores y excepciones: El código utiliza un enfoque estructurado para manejar errores y excepciones. Utiliza mensajes de error específicos y verificaciones de estado para manejar diferentes escenarios, lo que es esencial para una buena experiencia de usuario y para el mantenimiento del código.

Estándares de JS:

Consistencia en el uso de `const` y `let`: Verifica que todos los elementos que no cambian después de ser declarados estén definidos como `const`. En tu código, todos están bien definidos, pero asegúrate de mantener esa consistencia en todo el script.

Separación de lógica y manipulación del DOM: Aunque tu código ya sigue este principio, podrías considerar la posibilidad de separar aún más la lógica de obtención de datos (API calls) y la manipulación del DOM para mejorar la legibilidad y facilidad de mantenimiento.

Uso de funciones utilitarias: Podrías crear funciones utilitarias adicionales para evitar repetición de código, especialmente para tareas comunes como mostrar alertas o hacer fetch.

Control de errores y manejo de excepciones: Aunque tienes un manejo básico de errores con la función `sweetAlert`, considera implementar un manejo de excepciones más robusto para casos donde el fetch falle o se presenten errores inesperados.

Optimización del rendimiento: Observa el número de manipulaciones directas al DOM, ya que pueden afectar el rendimiento si hay muchas operaciones. Podrías considerar el uso de fragmentos de documento o técnicas similares para optimizar la manipulación del DOM.

REQUERIMIENTOS DE HARDWARE Y SOFTWARE.

Requerimientos de Hardware

Para el Usuario Final

- **Procesador**: Intel Core i3 o superior / AMD Ryzen 3 o superior. Un procesador moderno de doble núcleo puede manejar tareas de navegación básicas y carga de contenidos multimedia.
- **Memoria RAM**: Mínimo 4 GB (Recomendado 8 GB o más). Esto es suficiente para abrir múltiples pestañas de navegación simultáneamente sin ralentizaciones.
- **Espacio en Disco Duro**: Al menos 500 MB libres para almacenar archivos temporales y caché. Si el sitio web incluye videos o imágenes grandes, se requerirá espacio adicional.
- **Resolución de pantalla**: 1366x768 o superior. Recomendado 1920x1080 para una mejor experiencia de visualización, especialmente en sitios web con contenido multimedia.
- **Conexión a Internet**:
    - **Velocidad mínima**: 5 Mbps para una navegación fluida y cargas rápidas de páginas.
    - **Velocidad recomendada**: 10 Mbps o más si el sitio web incluye contenido interactivo, videos de alta definición o funciones en tiempo real.

Requerimientos de Software

Para el Usuario Final

- **Sistema Operativo**:
    - **Windows**: Windows 10 o superior.
    - **macOS**: macOS 10.15 (Catalina) o superior.
    - **Linux**: Cualquier distribución reciente (como Ubuntu 20.04, Fedora, etc.).
    - **Android e iOS**: Sitio web optimizado para dispositivos móviles con versiones recientes de Android o iOS.
- **Navegador Web**: Se recomienda utilizar la última versión de navegadores modernos para garantizar compatibilidad y seguridad:
    - **Google Chrome** (versión más reciente).
    - **Mozilla Firefox** (versión más reciente).
    - **Microsoft Edge** (basado en Chromium).
    - **Safari** (para usuarios de macOS y iOS).

- **Plugins y configuraciones adicionales**:
    - **JavaScript activado**: Es necesario para la mayoría de funcionalidades dinámicas y responsivas del sitio.
    - **Cookies habilitadas**: Para guardar preferencias de usuario, autenticación y mantener la experiencia personalizada.
    - **Almacenamiento local habilitado**: Para mejorar la velocidad de carga y permitir el almacenamiento temporal de datos en el navegador.
    - **Soporte WebGL** (si aplica): Si el sitio utiliza gráficos 3D o contenido interactivo avanzado, es importante tener habilitado WebGL en el navegador.
- **Seguridad**:
    - **Antivirus actualizado**: Se recomienda tener una solución antivirus activa para proteger contra amenazas al navegar por Internet.
    - **Cortafuegos**: Habilitado en el sistema operativo para mantener la seguridad mientras se accede a internet.

INSTALACIÓN Y CONFIGURACIÓN.

REQUISITOS PREVIOS

Conexión a Internet: Necesaria para descargar los archivos de instalación.

Requerimientos de hardware y software: Ver sección anterior sobre especificaciones mínimas.

Navegador compatible: Asegúrate de tener instalado un navegador moderno (Chrome, Firefox, Edge, Safari).

INSTALACIÓN

1. Inicia el navegador de tu preferencia (Google Chrome, Edge, etc.).
2. Escribir la URL del sitio en la barra de direcciones
3. El sitio debe cargarse correctamente en el navegador compatible. Si se experimentan problemas, asegúrate de que JavaScript y las cookies estén habilitados.

CONFIGURACIÓN

1. Dirígete al menú principal del sitio web y selecciona la opción **Registrarse**.
2. Completa el formulario con tus datos personales (nombre, correo electrónico, contraseña, etc.).
3. Una vez confirmada la cuenta, vuelve al sitio web y selecciona **Iniciar sesión**.
4. Ingresa tu usuario y contraseña para acceder a tu cuenta.

RESOLUCIÓN DE PROBLEMAS COMÚNES

- **Contraseña incorrecta**: Si olvidas tu contraseña, utiliza la opción de **¿Olvidaste tu contraseña?** y sigue los pasos para restablecerla.
- **Problemas de confirmación**: Si no recibes el correo de confirmación, revisa tu carpeta de spam o solicita un nuevo enlace desde la página de inicio de sesión.

- Si el sitio no se muestra correctamente, asegúrate de que estás usando la versión más reciente de tu navegador.
- Asegúrate de que JavaScript esté activado en la configuración del navegador.
- Si el sitio web ha tenido una actualización, asegúrate de limpiar la caché de tu navegador para evitar problemas de visualización.

PRUEBA DE CONFIGURACIÓN

Después de la instalación y configuración, asegúrate de realizar una prueba básica para verificar que todo funciona correctamente:

- Accede a tu cuenta y navega por las distintas secciones del sitio.
- Verifica que las funcionalidades principales, como formularios, carga de contenido multimedia, y enlaces, funcionen correctamente.