Entrega de fase de preparación

PRESENTADO POR:

- Alisson Ivania Zepeda Caceres 20220426
- Ricardo Nicolas Melara Rauda 20220685
- Juan Pablo Montes Espinoza 20210600
- Jhonny Alejandro Amaya Perez 20220658
- Manuel Enrique Contreras Rivera 20220481

TERCER AÑO DE BACHILLERATO,

ESPECIALIDAD DE DESARROLLO DE SOFTWARE

GRUPO 2B

02 octubre 2024

Índice

Introducción

Este manual técnico tiene como finalidad ofrecer una guía detallada sobre el funcionamiento y la implementación de la página web de la clínica quiropráctica [Nombre de la Clínica]. Esta plataforma ha sido diseñada con el objetivo de proporcionar una experiencia accesible y amigable para pacientes y personal administrativo. La clínica se compromete a facilitar la gestión de citas, la consulta de información sobre los servicios y tratamientos disponibles, así como a fomentar la interacción entre los pacientes y los profesionales de la salud. A través de este sitio web, los usuarios podrán registrarse, actualizar su información personal, agendar citas y acceder a recursos educativos relacionados con la quiropráctica. La estructura del sistema se basa en una base de datos bien organizada que almacena información esencial sobre pacientes, empleados, servicios y citas. En este manual, se describen la estructura de la base de datos, las funcionalidades de la aplicación web y las recomendaciones para su uso y mantenimiento.La creación de esta herramienta digital no solo optimiza los procesos internos de la clínica, sino que también mejora la comunicación y la atención al cliente, contribuyendo a ofrecer una atención quiropráctica de alta calidad.

Tecnologías y herramientas empleadas

BACKEND

- XAMPP v8.2.0 (PHP v8.2.0, Apache v2.4.54 y MariaDB v10.4.27)
- FPDF v1.85 (http://www.fpdf.org/)

FRONTEND

- JavaScript (Vanilla JS)
- Bootstrap v5.3.0 (https://getbootstrap.com/)
- Bootstrap Icons v1.10.5 (https://icons.getbootstrap.com/)
- SweetAlert v2.0 (https://sweetalert.js.org/)
- Chart.js v4.3.0 (https://www.chartjs.org/)

Estructura de la base de datos

```
DROP DATABASE if EXISTS db_quiropractica;

CREATE DATABASE db_quiropractica;

USE db_quiropractica;

CREATE TABLE
   tb_clientes (
      id_cliente INT PRIMARY KEY AUTO_INCREMENT NOT NULL,
      nombre_cliente VARCHAR(50) NOT NULL,
      apellido_cliente VARCHAR(50) NOT NULL,
      dui_cliente VARCHAR(10) unique,
      correo_cliente VARCHAR(100) NOT NULL,
      contrasenia_cliente VARCHAR(200) not null,
      fecha_contrasenia DATE,
      telefono_cliente VARCHAR(9) NOT NULL,
      nacimiento_cliente DATE NOT NULL,
      estado_cliente tinyint (1) NOT NULL DEFAULT 1,
      codigo_cliente VARCHAR(6) NOT NULL,
      imagen_cliente VARCHAR(500)
   );

-- proximamente
-- CREATE TABLE tb_testimonios(
-- id_testimonio INT PRIMARY KEY AUTO_INCREMENT NOT NULL,
-- titulo_testimonio VARCHAR(50) NOT NULL,
-- contenido_testimonio VARCHAR(200)NOT NULL,
-- id_cliente INT ,
-- estado_testimonio tinyint(1) NOT NULL,
-- FOREIGN KEY (id_cliente) REFERENCES tb_clientes (id_cliente)
-- );
CREATE TABLE
   tb_empleados (
      id_empleado INT PRIMARY KEY AUTO_INCREMENT NOT NULL,
      nombre_empleado VARCHAR(50) NOT NULL,
      apellido_empleado VARCHAR(50) NOT NULL,
      especialidad_empleado VARCHAR(50) DEFAULT 'especialidad no especificada',
      dui_empleado VARCHAR(10) NOT NULL unique,
      correo_empleado VARCHAR(100) NOT NULL unique,
      nacimiento_empleado DATE NOT NULL,
      estado_empleado TINYINT (1) NOT NULL,
      imagen_empleado VARCHAR(200)
   );
```

```
CREATE TABLE
   tb_admin (
      id_admin INT PRIMARY KEY AUTO_INCREMENT NOT NULL,
      nombre_admin VARCHAR(50) UNIQUE NOT NULL,
      correo_admin VARCHAR(100) unique NOT NULL,
      contrasenia_admin VARCHAR(500) NOT NULL,
      fecha_contrasenia date not null,
      id_empleado INT,
      FOREIGN KEY (id_empleado) REFERENCES tb_empleados (id_empleado) ON
DELETE CASCADE,
      codigo_admin VARCHAR(6) NOT NULL
   );

CREATE TABLE
   tb_imagenes (
      id_imagen INT PRIMARY KEY AUTO_INCREMENT NOT NULL,
      nombre_imagen VARCHAR(100) not null,
      imagen_1 VARCHAR(250) NOT NULL,
      imagen_2 VARCHAR(250),
      imagen_3 VARCHAR(250)
   );

CREATE TABLE
   tb_servicios (
      id_servicio INT PRIMARY KEY AUTO_INCREMENT NOT NULL,
      tipo_servicio VARCHAR(45) NOT NULL unique,
      descripción servicio VARCHAR(250) NOT NULL,
      imagen_servicio varchar(300) DEFAULT 'servicio.jpg',
      id_empleado INT,
      FOREIGN KEY (id_empleado) REFERENCES tb_empleados (id_empleado),
      id_imagen INT,
      FOREIGN KEY (id_imagen) REFERENCES tb_imagenes (id_imagen)
   );

CREATE TABLE
   tb_beneficios (
      id_beneficio INT PRIMARY KEY AUTO_INCREMENT NOT NULL,
      titulo_beneficio VARCHAR(30) unique,
      contenido_beneficio VARCHAR(200),
      id_servicio INT,
      FOREIGN KEY (id_servicio) REFERENCES tb_servicios (id_servicio)
   );

CREATE TABLE
   tb_preguntas (
      id_pregunta INT AUTO_INCREMENT PRIMARY KEY,
      nombre_pregunta VARCHAR(250) NOT NULL unique,
      contenido_pregunta VARCHAR(255) NOT NULL,
```

```
    imagen_pregunta VARCHAR(100) NOT NULL,
    id_empleado INT,
    FOREIGN KEY (id_empleado) REFERENCES tb_empleados (id_empleado),
    id_imagen INT,
    FOREIGN KEY (id_imagen) REFERENCES tb_imagenes (id_imagen)
  );

CREATE TABLE
  tb_citas (
    id_cita INT PRIMARY KEY AUTO_INCREMENT NOT NULL,
    nombre_cita VARCHAR(100) DEFAULT 'cita predeterminada',
    fecha_creacion_cita DATETIME NOT NULL DEFAULT current_timestamp(),
    fecha_asignacion_cita DATETIME NULL,
    estado_cita ENUM ('pendiente', 'proceso', 'terminado') NOT NULL DEFAULT
'pendiente',
    numero_seciones INT,
    id_cliente INT,
    FOREIGN KEY (id_cliente) REFERENCES tb_clientes (id_cliente),
    id_servicio INT,
    FOREIGN KEY (id_servicio) REFERENCES tb_servicios (id_servicio),
    id_empleado INT NULL,
    FOREIGN KEY (id_empleado) REFERENCES tb_empleados (id_empleado)
  );

CREATE TABLE
  tb_nombres_tratamientos (
    id_tratamiento INT PRIMARY KEY AUTO_INCREMENT NOT NULL,
    nombre_tratamiento VARCHAR(75) NOT NULL,
    notas_adicionales VARCHAR(250),
    id_cita INT,
    FOREIGN KEY (id_cita) REFERENCES tb_citas (id_cita)
  );

CREATE TABLE
  tb_comentarios (
    id_comentario INT PRIMARY KEY AUTO_INCREMENT NOT NULL,
    contenido_comentario VARCHAR(250) NOT NULL,
    id_cliente INT,
    FOREIGN KEY (id_cliente) REFERENCES tb_clientes (id_cliente),
    id_servicio INT,
    FOREIGN KEY (id_servicio) REFERENCES tb_servicios (id_servicio),
    estado_comentario tinyint (1) NOT NULL DEFAULT 0
  );
```

Diccionario de datos

Tabla: tb_clientes

Columna	Tipo	Longitud	Nulo	Índice	Predeterminado
id_cliente	INT	-	NO	PRIMARY KEY	AUTO_INCREMENT
nombre_cliente	VARCHAR	50	NO	-	-
apellido_cliente	VARCHAR	50	NO	-	-
dui_cliente	VARCHAR	10	YES	UNIQUE	-
correo_cliente	VARCHAR	100	NO	-	-
contrasenia_cliente	VARCHAR	200	NO	-	-
fecha_contrasenia	DATE	-	YES	-	-
telefono_cliente	VARCHAR	9	NO	-	-
nacimiento_cliente	DATE	-	NO	-	-
estado_cliente	TINYINT	1	NO	-	1
codigo_cliente	VARCHAR	6	NO	-	-
imagen_cliente	VARCHAR	500	YES	-	-

Tabla: tb_empleados

Columna	Tipo	Longitud	Nulo	Índice	Predeterminado
id_empleado	INT	-	NO	PRIMARY KEY	AUTO_INCREMENT
nombre_empleado	VARCHAR	50	NO	-	-
apellido_empleado	VARCHAR	50	NO	-	-
especialidad_empleado	VARCHAR	50	YES	-	-
dui_empleado	VARCHAR	10	NO	UNIQUE	-
correo_empleado	VARCHAR	100	NO	UNIQUE	-
nacimiento_empleado	DATE	-	NO	-	-
estado_empleado	VARCHAR	1	NO	-	-
imagen_empleado	DATE	200	YES	-	-

Tabla: tb_admin

Columna	Tipo	Longitud	Nulo	Índice	Predeterminado
id_admin	INT	-	NO	PRIMARY KEY	AUTO_INCREMENT
nombre_admin	VARCHAR	50	NO	UNIQUE	-
correo_admin	VARCHAR	100	NO	UNIQUE	-
contrasenia_admin	VARCHAR	500	NO	-	-
fecha_contrasenia	DATE	-	NO	-	-
id_empleado	INT	-	YES	-	-
codigo_admin	VARCHAR	6	NO	-	-

Tabla: tb_imagenes

Columna	Tipo	Longitud	Nulo	Índice	Predeterminado
id_imagen	INT	-	NO	PRIMARY KEY	AUTO_INCREMENT
nombre_imaen	VARCHAR	100	NO	-	-
imagen_1	VARCHAR	250	NO	-	-
imagen_2	VARCHAR	250	YES	-	-
imagen_3	VARCHAR	250	YES	-	-

Tabla: tb_servicios

Columna	Tipo	Longitud	Nulo	Índice	Predeterminado
id_servicio	INT	-	NO	PRIMARY KEY	AUTO_INCREMENT
tipo_servicio	VARCHAR	45	NO	UNIQUE	-
descripcion_servicio	VARCHAR	250	NO	-	-
id_imagen	INT	-	NO	FOREIGN KEY	-
id_empleado	INT	-	YES	-	-

Tabla: tb_benificios

Columna	Tipo	Longitud	Nulo	Índice	Predeterminado
id_beneficio	INT	-	NO	PRIMARY KEY	AUTO_INCREMENT
titulo_beneficio	VARCHAR	30	YES	UNIQUE	-
contenido_beneficio	VARCHAR	200	YES	-	-
id_servicio	INT	-	YES	-	-

Tabla: tb_preguntas

Columna	Tipo	Longitud	Nulo	Índice	Predeterminado
id_pregunta	INT	-	NO	PRIMARY KEY	AUTO_INCREMENT
nombre_pregunta	VARCHAR	250	NO	UNIQUE	-
contenido_pregunta	VARCHAR	255	NO	-	-
imagen_pregunta	VARCHAR	100	NO	-	-
id_empleado	INT	-	YES	-	-

Tabla: tb_citas

Columna	Tipo	Longitud	Nulo	Índice	Predeterminado
id_cita	INT	-	NO	PRIMARY KEY	AUTO_INCREMENT
nombre_cita	DATETIME	100	YES	-	-
fecha_creacion_cita	DATETIME	-	NO	-	CURRENT_TIMESTAMP
fecha_asignacion_cita	ENUM	-	NO	-	-
numero_seciones	INT	-	YES	-	-
id_cliente	INT	-	YES	-	-
id_servicio	INT	-	YES	-	-
id_empleado	INT	-	YES	-	-

Tabla: tb_nombres_tratamientos

Columna	Tipo	Longitud	Nulo	Índice	Predeterminado
id_tratamiento	INT	-	NO	PRIMARY KEY	AUTO_INCREMENT
nombre_tratamiento	VARCHAR	75	NO	-	-
notas_adicionales	VARCHAR	250	YES	-	-
id_cita	INT	-	YES	-	-

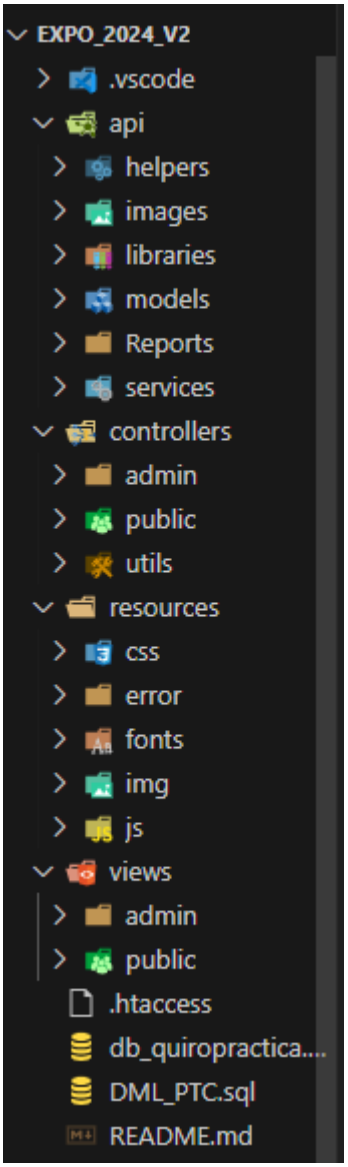
Tabla: tb_comentarios

Columna	Tipo	Longitud	Nulo	Índice	Predeterminado
id_comentario	INT	-	NO	PRIMARY KEY	AUTO_INCREMENT
contenido_comentario	VARCHAR	250	NO	-	-
id_cliente	INT	-	YES	-	-
id_servicio	INT	-	YES	-	-
estado_comentario	TINYINT	1	NO	-	0

Arquitectura de software

Las vistas son componentes de la interfaz web desarrolladas en HTML5 y Bootstrap, diseñadas para interactuar con controladores en JavaScript mediante programación asíncrona. Esta técnica permite realizar peticiones al servidor y recibir respuestas en tiempo real, facilitando la manipulación del DOM sin necesidad de recargar la página. Las solicitudes de datos se gestionan utilizando los métodos HTTP GET y POST, dependiendo de si se busca obtener información o enviar datos para actualizaciones. Los elementos del frontend, es decir, las vistas y controladores, son los únicos que se ejecutan en el navegador. Aunque se puede usar Vanilla JS, también se pueden emplear librerías y frameworks como React, Vue o Angular para implementar la programación asíncrona. Cada página web cuenta con un controlador principal que organiza las interacciones y maneja las peticiones y respuestas a través de diversos métodos y eventos. En el backend, la API es fundamental para gestionar las solicitudes y respuestas en formato JSON. Esta API está construida en PHP y se apoya en una capa de servicios que procesa las acciones definidas en los endpoints. La validación de datos de entrada se realiza en los modelos, que son representaciones de las tablas de la base de datos y están diseñados utilizando programación orientada a objetos (OOP). Los modelos están organizados en clases DATA y HANDLER, donde la clase HANDLER se encarga de las operaciones en la base de datos mediante la extensión PDO, que permite un acceso eficiente a múltiples gestores de bases de datos. También se implementan medidas de seguridad para la autenticación de usuarios y manejo de errores, asegurando que las operaciones sean seguras y controladas.

Estructura del proyecto



- **API**: este directorio contiene la interfaz de programación del lado del servidor. En él, el servicio gestiona la lógica empresarial para recibir solicitudes y enviar respuestas. Los models manejan el procesamiento de datos, mientras que los helpers proporcionan funcionalidad general como validación y conexiones de bases de datos. Images almacena archivos de imágenes, libraries contiene bibliotecas de terceros que amplían la funcionalidad de la aplicación, como PHPMailer y fpdf, y reports contiene la programación necesaria para generar informes desde la base de datos.

- **Controllers**: representan la lógica del lado del cliente que interactúa con la API realizando solicitudes y recibiendo respuestas. Las carpetas admin y public contienen controladores específicos para cada tipo de sitio, mientras que la carpeta utils contiene scripts genéricos.

- **Resources**: este directorio es una colección de archivos estáticos comunes, como archivos propios o de terceros, como hojas de estilo CSS, fuentes, imágenes, scripts JS y páginas personalizadas de manejo de errores.

- **Views**: todas las páginas web de la aplicación se encuentran aquí. Esta carpeta contiene carpetas para sitios públicos (public) y sitios privados (admin).

Diseño de la aplicación

Paleta de Colores

Colores Primarios:

- Negro (#000000): Utilizado principalmente para el texto y elementos de interfaz donde se requiere un alto contraste.
- Blanco (#FFFFFF): Usado como fondo principal en las vistas para mantener una apariencia limpia y minimalista.
- Azul Claro (#ADD8E6): Empleado para destacar elementos interactivos como botones, enlaces y notificaciones.

Tipografía

Fuente para Subtítulos:

- Tipo de Letra: Arial, Helvetica
- Tamaño: 12 pt

Fuente para Títulos:

- Tipo de Letra: Arial, Helvetica
- Tamaño: 20 pt

Imágenes

Dimensiones de las Imágenes:

- Máximo Ancho: 600 px
- Mínimo Ancho: 400 px
- Las imágenes se adaptan a estas dimensiones para asegurar una presentación visual coherente y estética dentro de la aplicación.

Adaptabilidad

Sistemas Operativos Soportados:

- Android 7 y superiores

Tamaños de Pantalla:

- Pantallas de 6 pulgadas: La interfaz está optimizada para ofrecer una experiencia de usuario fluida y responsiva en dispositivos con pantallas de 6 pulgadas, ajustando automáticamente los elementos de diseño para mantener la usabilidad y accesibilidad en diversas resoluciones y orientaciones de pantalla.

Implementación

Elementos UI (Interfaz de Usuario):

- Botones y enlaces destacados en azul claro para indicar interactividad.
- Textos principales en negro para garantizar legibilidad.
- Fondos blancos para proporcionar un contraste claro y minimizar la fatiga visual.

Composición y Estilo

- La combinación de negro, blanco y azul claro crea un contraste visual efectivo y un diseño moderno.
- Las fuentes y tamaños están seleccionados para asegurar que la información sea legible y jerárquicamente organizada, mejorando así la experiencia del usuario.

Pruebas y Optimización

- Se realizarán pruebas exhaustivas en dispositivos con Android 7 y pantallas de 6 pulgadas para asegurar la compatibilidad y el rendimiento óptimo.
- La adaptabilidad del diseño garantizará una visualización adecuada en diferentes condiciones de uso y orientaciones de pantalla.

Buenas prácticas de desarrollo

JavaScript:
Nombres de Funciones **(**camelCase**)**:

- Propósito: Facilita la lectura y entendimiento del código. Permite identificar rápidamente qué se trata de una función.

Constantes (Mayúsculas):

- Propósito: Indica que el valor es constante y no debe cambiar, ayudando a prevenir modificaciones accidentales.

Terminación de Sentencias con Punto y Coma (;):

- Propósito: Previene errores en la ejecución del código, asegurando que cada sentencia se ejecute de forma independiente.

Variables (let en minúsculas):

- Propósito: Mantiene una convención clara, diferenciando variables de otros tipos como funciones o constantes.

sangría (4 espacios):

- Propósito: Mejora la legibilidad del código, facilitando la comprensión de la estructura y el flujo del mismo.

Comentarios:

- Propósito: Ayudan a documentar el código, explicando su lógica y propósito, lo que es útil para otros desarrolladores y para el mantenimiento futuro.

PHP:

Nombres de Clases (Study Caps o PascalCase):

- Propósito: Facilita la identificación de clases dentro del código, diferenciándose de funciones y variables.

Nombres de Métodos (camelCase):

- Propósito: Mantiene una convención clara que indica que se trata de un método, ayudando en la legibilidad.

Identación (4 espacios):

- Propósito: Igual que en JavaScript, mejora la legibilidad y organización del código.

Comentarios:

- Propósito: Proporcionar contexto y explicaciones sobre la funcionalidad de métodos y clases, facilitando el entendimiento por parte de otros desarrolladores.

Estándares para la base de datos:

Nombres de Tablas (snake_case):

- Propósito: Establecer una convención que mejora la claridad y uniformidad al referirse a las tablas en las consultas.

Nombres de Columnas (snake_case):

- Propósito: Similar a las tablas, esta convención ayuda a mantener la claridad y la consistencia en la estructura de la base de datos.

Uso de Comentarios en Consultas SQL:

- Propósito: Explican el propósito de consultas complejas, lo que es útil para el mantenimiento y la colaboración con otros desarrolladores.

Requerimientos de hardware y software

1. Requisitos de Hardware

- **Servidor**:
  - Procesador: Al menos un procesador de doble núcleo a 2 GHz.
  - Memoria RAM: Mínimo 8 GB.
  - Almacenamiento: Espacio en disco de 100 GB (preferentemente en SSD para un acceso más rápido).
  - Conexión a Internet: Ancho de banda de un mínimo de 10 Mbps.
- **Estaciones de Trabajo**:
  - Procesador: Al menos un procesador de doble núcleo a 1.8 GHz.
  - Memoria RAM: Mínimo 4 GB.
  - Almacenamiento: Espacio en disco de 50 GB.
  - Pantalla: Resolución mínima de 1366x768.

2. Requisitos de Software

- **Servidor**:
  - Sistema Operativo: Windows, Linux (preferiblemente Ubuntu) o macOS.
  - Software de Servidor:
    - XAMPP (que incluye PHP, Apache y MariaDB).
    - PHP versión 8.2.0 o superior.
    - MariaDB versión 10.4.27 o superior.
- **Desarrollo Frontend**:
  - Navegador Web: La última versión de Google Chrome, Mozilla Firefox o Safari.
  - Frameworks y Bibliotecas:
    - Bootstrap 5.3.0 para el diseño responsivo.
    - JavaScript (Vanilla JS) para la lógica del lado del cliente.
    - Chart.js para la visualización de gráficos.
- **Herramientas de Desarrollo**:
  - Editor de Código: Visual Studio Code, Sublime Text o Atom.
  - Control de versiones: Git, recomendado para gestionar el historial de cambios.

3. Configuraciones Adicionales

- **Base de Datos**:
  - Configuración óptima de MariaDB para mejorar el rendimiento, incluyendo la creación de índices en las tablas.
  - Asegurar conexiones seguras y autenticadas para acceder a la base de datos.
- **Seguridad**:
  - Implementación de certificados SSL para cifrar los datos en tránsito.
  - Uso de software antivirus y cortafuegos en el servidor y en las estaciones de trabajo.

Instalación y configuración

1. Requisitos Previos

- **Hardware**: Servidor o computadora con procesador de doble núcleo y al menos 8 GB de RAM.
- **Software**: Instalar XAMPP (incluye Apache, PHP y MariaDB).

2. Descarga de Software

- Descarga XAMPP desde su sitio oficial y cualquier librería adicional necesaria (FPDF).

3. Instalación de XAMPP

- Ejecuta el instalador y selecciona los componentes que deseas instalar.
- Completa la instalación y abre el Panel de Control de XAMPP.

4. Configuración del Servidor

- Inicia los servicios de Apache y MySQL desde el Panel de Control.
- Accede a phpMyAdmin en http://localhost/phpmyadmin para gestionar bases de datos.

5. Creación de la Base de Datos

- Crea una nueva base de datos (quiropractica_db) y, si es necesario, importar el esquema desde un archivo SQL.

6. Configuración de Archivos del Proyecto

- Descarga el código fuente de la aplicación y colócalo en la carpeta htdocs de XAMPP.
- Ajusta los parámetros de conexión en el archivo de configuración (config.php).

7. Acceso a la Aplicación

- Abre un navegador y dirígete a http://localhost/quiropractica para acceder a la plataforma.

8. Pruebas y Mantenimiento

- Realiza pruebas funcionales para asegurar el correcto funcionamiento y realiza copias de seguridad periódicas.