Informe Proyecto
"Website Sattva"

Título del Proyecto:

WebSite Sattva (Espacio de Salud y Bienestar)

Autor:

Alumno: Fausto Villegas

DNI: 45923040

Nombre del Tutor/Asesor:

Miguel Mendez Garabetti

Fecha de Presentación:

13/12/2024

Universidad/Institución:

Universidad CAECE

Carrera:

Tecnicatura Universitaria en Programación

RESUMEN

Este proyecto consiste en la creación de un sitio web para el centro de entrenamiento Sattva. Sattva ofrece una variedad de actividades físicas como yoga, entrenamiento funcional y streching. El sitio web tendrá las siguientes secciones:

- **Página de inicio**: Proporcionará información sobre el centro, los tipos de clases disponibles, horarios de clases y eventos programados (al seleccionar una clase o evento, se brindará información más detallada para cada una de estas). Se podrán inscribir a clases o eventos que deseen mediante el pago de una cuota o inscripción. También tendrá información de contacto (email, teléfono).
- **Tienda**: Los usuarios podrán comprar productos relacionados con las actividades ofrecidas. Para realizar una compra, los usuarios deberán registrarse e iniciar sesión en sus cuentas. En la página de la tienda, podrán agregar productos a un carrito y completar la compra realizando el pago.

La funcionalidad básica que se desarrolla para el proyecto es un e-commerce que nos presenta una lista de productos disponibles de los cuales el usuario podrá seleccionar y agregar al carrito para luego comprarlos. Por lo que tendrá las funcionalidades básicas de un sistema de login de usuarios, una interfaz que presente los productos disponibles y un carrito en el que se podrá finalizar las compras de los productos seleccionados por el usuario.

Para extender y completar el proyecto se tendrá una página de inicio, en la cual se tendrá un diseño estético y desde donde se podrá visualizar los diferentes tipos de clases disponibles a las cuales el usuario se podrá anotar y asistir, cada clase contará con una descripción y se mostrarán los horarios disponibles. Además de una lista de Eventos, los cuales son

actividades que se realizan en una fecha y horario específico y es dado por un profesor/a del centro Sattva.

La plataforma utiliza un diseño full-stack con Angular en el frontend, Node.js y MySQL en el backend, y se despliega mediante Vercel y Railway. Se espera que el sitio incremente la accesibilidad de los servicios de Sattva, proporcionando un entorno eficiente y seguro para los usuarios.

Conclusiones alcanzadas:

El sistema ha permitido integrar las operaciones de Sattva en una plataforma accesible, cumpliendo con los objetivos planteados, aunque se identificaron áreas de mejora como optimización de la interfaz y pruebas de seguridad.

Índice

Introducción

Ámbito del proyecto

Hoy en día las aplicaciones Web son muy importantes en las empresas, estas permiten llegar a muchas más cantidad de clientes incrementando así el volumen de negocio y por consiguiente haciendo que la empresa y sus beneficios aumenten. Existen muchas empresas que solo tienen base en internet, por lo que no trabajan necesariamente con un lugar físico.
Las aplicaciones Web nos dan una gran cantidad de opciones adicionales para manejar un negocio. Dado que Sattva trabaja sin ningún sistema que controle el pago de los alumnos a las clases, que le permita a los clientes pagar a distancia o que puedan comprar productos que ofrece Sattva sin estar en el local físico. Es por esto que se decide desarrollar una página Web para el centro de entrenamiento y salud Sattva.

El centro de entrenamiento y meditación Sattva quiere diseñar su propia página web para poder administrar su negocio. En esta página se va a encontrar una sección de e-commerce donde se venderán diferentes tipos de productos. Además, los usuarios se podrán inscribir a diferentes tipos de clases y eventos en diferentes horarios disponibles, por lo que la página contará con un catálogo de clases y eventos en el que se tendrá una breve descripción de cada uno.

Planteamiento del problema

El centro de entrenamiento y meditación Sattva necesita una plataforma digital para gestionar su negocio, incluyendo una sección de e-commerce para la venta de diferentes productos, un

sistema de registro de usuarios que almacene la información de los clientes que accedan a la página, realizar un manejo de clases y eventos a los cuales los usuarios podrán inscribirse, y un sistema de pagos para realizar el pago de lo que desee el usuario (productos, inscripción a eventos y/o cuotas de clases).

Objetivos

El objetivo de este proyecto es mostrar y explicar las etapas del desarrollo de la página Web de Sattva. Poder mostrar los conocimientos adquiridos durante el tiempo de estudio en la Universidad CAECE. Para eso, se decidió el desarrollo full-stack de una aplicación Web, en la que principalmente se destacan los componentes: frontend, backend y la base de datos (se hablará de las tecnologías utilizadas más adelante).

Objetivos específicos.

El objetivo principal de este proyecto es proporcionar a los usuarios interesados en las actividades de Sattva una plataforma donde puedan obtener información detallada sobre las clases, sus beneficios y características. Además, podrán comprar productos relacionados. La página web servirá como herramienta de marketing para Sattva, permitiendo a los visitantes informarse sobre las actividades y eventos, así como solicitar inscripciones.

Los ingresos provendrán principalmente de las inscripciones mensuales o únicas, y también de las ventas en la tienda en línea.

- Implementar un sistema de e-commerce para productos.
- Crear un sistema de registro y autenticación de usuarios.
- Desarrollar un sistema de gestión de clases y eventos.
- Implementar un sistema de pagos para clases, eventos y productos.

Público objetivo

El público objetivo de este proyecto son personas interesadas en actividades físicas, relajación, y adquisición de productos relacionados con el bienestar. Además, la página atraerá a quienes buscan información y participación en eventos especiales, como meditaciones y charlas informativas. La interfaz del sitio web se diseñará de manera amigable, siendo accesible y fácil de usar para personas sin experiencia previa en sitios web.

Estudio de otros proyectos

Se realizó una revisión de diferentes páginas web para poder ver y tomar como referencia proyectos ya realizados y funcionando para poder tomar una guía. Por ejemplo, utilicé como referencia los detalles del producto de mercado libre.

Desarrollo

Análisis del Proyecto

Levantamiento de requerimientos:

Requerimientos Funcionales (RF)

Sección Página de Inicio

1. La página de inicio debe mostrar información general sobre el centro Sattva.
2. Debe listar los tipos de clases disponibles con una descripción breve y horarios.
3. Los usuarios deben poder seleccionar una clase para ver información más detallada.
4. Debe incluir una lista de eventos programados con información de estos.
5. Los eventos deben mostrar una descripción detallada al seleccionarlos.
6. Debe existir una lista de información de contacto en la página de inicio que contenga una lista de posibles métodos para contactar con el centro.
7. El sistema deberá mandar una confirmación de pago de inscripción a un evento o clase al mail del usuario.

Sección Tienda

1. La página de la tienda debe mostrar una lista de productos disponibles con su título, precio e imagen.
2. Los usuarios deben poder agregar productos al carrito de compras.
3. El sistema debe permitir a los usuarios editar las cantidades de productos en el carrito.
4. Los usuarios deben poder eliminar productos del carrito.
5. Los usuarios deben poder finalizar la compra desde el carrito.

6. Los usuarios deben poder realizar los pagos necesarios mediante MercadoPago.
7. El sistema deberá mandar una confirmación de compra de productos cuando se realice un pago al mail del usuario.
8. Los usuarios deben poder seleccionar un producto y ver una página de detalle de producto con opción de poder agregar la cantidad que deseen de este al carrito.
9. Los usuarios deben poder filtrar productos por su nombre mediante un buscador.
10. Los usuarios deben poder filtrar productos por categoría (ejercicio, meditación, decoración).
11. Los usuarios deben poder filtrar productos por precio (ascendente, descendente).

Gestión de Usuarios

1. El sistema debe permitir a los usuarios registrarse proporcionando los datos: nombre, correo electrónico, contraseña, y confirmación de contraseña.
2. Los usuarios registrados deben poder iniciar sesión con su correo electrónico y contraseña.
3. Los usuarios deben poder registrarse o iniciar sesión mediante su cuenta de Google.
4. Los usuarios deben poder cerrar sesión.
5. El sistema debe permitir que los usuarios establezcan su contraseña mediante una confirmación que se manda al mail.
6. El sistema debe permitir a los usuarios actualizar información de su perfil.

Panel de Administración

1. Los administradores deben poder agregar, editar y eliminar clases y eventos desde un panel de administración.

2. Los administradores deben poder gestionar productos en la tienda (crear, editar, eliminar).
3. Los administradores deben poder ver las órdenes de compra de los usuarios.
4. Los administradores deben poder gestionar a los instructores del centro (crear, editar, eliminar)

Requerimientos No Funcionales (RNF)

1. La base de datos debe ser capaz de manejar al menos 1,000 usuarios registrados simultáneamente.
2. Las contraseñas de los usuarios deben ser almacenadas de manera segura usando hashing.
3. El sitio debe ser compatible con los navegadores más utilizados (Chrome, Firefox, Safari, Edge).
4. Debe implementarse un sistema de validación de entrada para prevenir inyecciones SQL o ataques XSS.
5. El sitio debe ser fácil de navegar, con menús claros y accesibles.
6. Deben usarse mensajes de error claros para guiar al usuario en caso de entradas incorrectas (Ej: “La contraseña debe tener al menos 5 caracteres”).
7. El sitio debe ser compatible con los navegadores más utilizados (Chrome, Firefox, Safari, Edge)
8. El sistema debe incluir herramientas de monitoreo de errores (e.g., logs).

Lista de casos de uso

Gestión de Usuarios

CU001: Registrar usuario: Un usuario puede registrarse proporcionando su información (nombre, correo electrónico y contraseña).

CU002 Iniciar sesión: Un usuario registrado puede autenticarse en el sistema ingresando su correo electrónico y contraseña.

CU003 Cerrar sesión: Un usuario autenticado puede salir de su cuenta.

CU004 Actualizar perfil: Un usuario autenticado puede modificar su información personal (nombre, dirección, número de teléfono, etc.).

CU005 Recuperar contraseña: Un usuario puede solicitar el restablecimiento de su contraseña mediante su correo electrónico.

CU006 Iniciar sesión con Google: Un usuario podrá registrarse o iniciar sesión ingresando su cuenta de Google.

Página de Inicio

CU007 Ver lista de clases: Los usuarios pueden visualizar las clases disponibles con sus descripciones y horarios.

CU008 Consultar detalles de una clase: Los usuarios pueden ver información detallada de una clase específica.

CU009 Ver lista de eventos: Los usuarios pueden consultar los eventos programados.

CU10 Consultar detalles de un evento: Los usuarios pueden ver información detallada de un evento específico.

CU011 Consultar clases y eventos inscriptos: Los usuarios registrados consultan por las clases y eventos a los que están inscriptos.

Tienda

CU012 Consultar productos: Los usuarios pueden ver una lista de productos disponibles en la tienda con sus títulos, precios e imágenes.

CU013 Agregar producto al carrito: Los usuarios autenticados pueden añadir productos al carrito de compras.

CU014 Ver su carrito: Los usuarios pueden ver su carrito con los productos que agregaron y sus cantidades.

CU015 Editar cantidades en el carrito: Los usuarios pueden modificar la cantidad de cada producto en su carrito.

CU016 Eliminar producto del carrito: Los usuarios pueden quitar productos del carrito.

CU017 Realizar compra: Los usuarios autenticados pueden completar una compra ingresando los datos necesarios (Datos del usuario completos, método de pago).

CU018 Filtrar productos por nombre: Los usuarios podrán filtrar la lista de productos por el nombre del producto mediante un buscador.

CU019 Filtrar productos por categoría: Los usuarios podrán filtrar la lista de productos por una de las categorías disponibles (ejercicio, meditación, decoración).

CU020 Filtrar productos por precio: Los usuarios podrán filtrar la lista de productos por orden de precio (ascendente, descendente).

CU021 Ver detalles de producto: Los usuarios podrán seleccionar un producto y ver una pantalla con los detalles de este.

Administrador

CU022 Agregar nueva clase: El administrador accede a la interfaz para agregar una nueva clase.

CU023 Editar datos de clase: El administrador accede a la interfaz para editar una clase específica.

CU024 Eliminar clase: El administrador selecciona eliminar una clase específica.

CU025 Agregar nuevo evento: El administrador accede a la interfaz para crear un evento nuevo.

CU026 Editar datos de evento: El administrador accede a la interfaz para editar un evento específico.

CU027 Eliminar evento: El administrador selecciona eliminar un evento específico.

CU028 Agregar nuevo producto: El administrador accede a la interfaz para agregar un producto nuevo.

CU029 Editar datos de producto: El administrador accede a la interfaz para editar los datos de un producto específico.

CU030 Eliminar producto: El administrador selecciona eliminar un producto específico.

CU031 Ver lista de ordenes: El administrador consulta por las ordenes que fueron realizadas y sus estados.

CU032 Agregar nuevo instructor: El administrador accede a la interfaz para agregar un instructor nuevo.

CU033 Editar datos de instructor: El administrador accede a la interfaz para editar los datos de un instructor específico.

CU034 Eliminar instructor: El administrador selecciona eliminar un instructor específico.

Metodología

Enfoque y estrategia de desarrollo.

El proyecto sigue una metodología ágil, permitiendo iteraciones rápidas y adaptaciones basadas en el feedback. Se centra en el desarrollo incremental, entregando partes funcionales del sistema de manera continua.

Para desarrollar este proyecto se fue utilizando un repositorio remoto github para ir almacenando los avances del trabajo a medida que se fueron realizando. Se utiliza una rama 'main', esta es la rama principal la cual contiene el proyecto principal, y una rama 'test', en la cual se guardaba siempre primero los cambios antes de realizarlos en la rama principal.

Contenidos y estructura del sitio web

El sitio web cuenta con diferentes secciones que la separan según su contenido y funcionalidad. Las **secciones** son:

/ o **/home** : Esta es la **página de inicio**, es lo que el usuario ve en primera instancia cuando abre el sitio web, se puede ver un título y un fondo para llamar la atención de los usuarios, en esta mísma sección podremos encontrar una lista de clases y eventos a los cuales el usuario se podrá inscribir.

/products: Esta es la **tienda** de la página web, aquí se encontrará una lista de productos disponibles para que el usuario pueda comprar

/product/:id: Muestra información específica de un producto con su descripción.

/login: esta es la sección que le permite a el usuario registrarse con una cuenta y/o iniciar sesión.

/cart: En esta sección podremos ver los diferentes productos que el usuario decidió agregar al carrito para su próxima compra, cuenta con información de cada producto que se encuentra en este.

/profile: Esta sección incluye información del usuario registrado y la posibilidad de editar esta información.

/my-registrations: Esta sección mostrará dos listas, en donde aparecerán los eventos y las clases a los que el usuario está inscripto, mostrando información sobre ellas.

/reset-password: Esta interfaz le permite al usuario cambiar su contraseña, esto solo se puede hacer mediante el uso del boton (Olvidé mi contraseña)

Hay secciones adicionales las cuales solo son accesibles por usuarios con el rol de administrador:

/addClass: Cuenta con una interfaz para que el administrador agregue una clase nueva.

/addEvent: Cuenta con una interfaz para que el administrador agregue un evento nuevo.

/edit-class/:id: Permite editar la información de una clase en específico para actualizar sus datos.

/edit-event/:id: Permite editar la información de un evento en específico para actualizar sus datos.

/instructors: Muestra los instructores que están almacenados y permite eliminar, modificar o agregar instructores.

/addProduct: Le permite al administrador crear y almacenar un nuevo producto para poner a la venta.

/edit-product/:id: Muestra una interfaz para modificar los datos de un producto que ya está en la base de datos.

/orders-list: Muestra una lista de las diferentes ordenes de productos que realizaron los usuarios.

Dominio de la página Web:

https://web-sattva.vercel.app/

Marco Teórico

Frontend:

- **Angular**: Framework principal para la construcción de la interfaz de usuario.
- **TypeScript**: Lenguaje de programación utilizado en Angular.
- **HTML/CSS**: Para el diseño y maquetación de la UI.
- **Bootstrap - Material Design**: Frameworks de diseño para estilos y componentes.

Backend:

- **Node.js**: Entorno de ejecución de JavaScript en el servidor.
- **Express**: Framework para la creación de APIs.
- **JWT (JSON Web Tokens)**: Para autenticación y manejo de sesiones.
- **Multer**: un middleware para manejar cargas de archivos en Node.js

Base de Datos:

- **MySQL**: Para el almacenamiento de datos.
- **Vercel Blob**: Para almacenar las imágenes de clases, eventos y productos.

Diseño Web

Aspectos generales

Lenguajes y tecnologías utilizadas

- HTML
- CSS
- TypeScript
- JavaScript
- JSON
- Angular
- Express

Aplicaciones utilizadas

- VisualStudioCode
- MySQL Workbench
- Postman

Frontend

Frameworks y Librerías Principales

- **Angular:**
    - @angular/animations
    - @angular/cdk
    - @angular/common
    - @angular/compiler
    - @angular/core
    - @angular/forms

    - @angular/material
    - @angular/platform-browser
    - @angular/platform-browser-dynamic
    - @angular/router

Herramientas de Desarrollo y Construcción

- Angular CLI: @angular/cli
- Angular DevKit: @angular-devkit/build-angular
- TypeScript: typescript
- Karma (para pruebas):
    - karma
    - karma-chrome-launcher
    - karma-coverage
    - karma-jasmine
    - karma-jasmine-html-reporter
- Jasmine (para pruebas):
    - jasmine-core
    - @types/jasmine
- Autoprefixer: autoprefixer
- PostCSS: postcss
- TailwindCSS: tailwindcss

Librerías de Utilidad

- RxJS: rxjs
- Zone.js: zone.js
- JWT Decode: jwt-decode
- JWT Encode: jwt-encode

- Multer (para manejo de archivos): multer
- MercadoPago SDK: @mercadopago/sdk-react
- Vercel Blob: @vercel/blob

Utilidades de Angular para el Desarrollo

- Angular Compiler CLI: @angular/compiler-cli

Backend:

Frameworks y Librerías Principales

- **NodeJS**
- **Express**: express
- **MySQL:** mysql2

Herramientas y Utilidades

- Axios: axios (para realizar solicitudes HTTP)
- Body-parser: body-parser (para analizar cuerpos de solicitudes)
- CORS: cors (para habilitar solicitudes entre dominios)
- Crypto: crypto (para criptografía)
- Dotenv: dotenv (para gestionar variables de entorno)
- Express-validator: express-validator (para validaciones)
- JSON Web Token (JWT): jsonwebtoken (para autenticación y manejo de tokens)
- Multer: multer (para manejo de archivos)
- Morgan: morgan (para registros de solicitudes HTTP)
- Bcrypt.js: bcryptjs (para hash de contraseñas)
- Nodemailer: nodemailer (para envío de correos electrónicos)
- MercadoPago SDK: mercadopago (para integración con MercadoPago)

- Vercel Blob: @vercel/blob

Herramientas de Desarrollo

- Nodemon: nodemon (para reiniciar automáticamente el servidor durante el desarrollo

Manejo de errores

Para el manejo de errores se utilizaron bloques try-catch y el servicio error-handler.service.ts, para capturar los errores y poder mostrarlos por consola y/o pantalla para así poder observar la información sobre estos.

Siempre que se realiza una acción se tiene en cuenta que puede surgir un error y si este es provocado, se mostrará la información necesaria para una correcta retroalimentación.

Deployment

- **Vercel** (proporciona las herramientas de desarrollo y la infraestructura en la nube para crear, escalar y proteger una web)
    - **Frontend**
    - **Backend**
- **Railway** (Railway es una plataforma en la nube que le permite aprovisionar, desarrollar e implementar aplicaciones con facilidad)
    - **Base de Datos**

Monetización

Fuentes de ingresos

Las fuentes de ingresos con las que contará la página son la venta de productos online, las inscripciones a clases con una cuota mensual y las inscripciones a eventos específicos.

Formas de cobro

Para que los usuarios puedan realizar pagos se integró Mercado Pago para que se pueda tanto utilizar directamente una cuenta de Mercado Pago o una tarjeta de credito o debito.

Venta online de productos y/o servicios

Algunos ejemplos de productos y servicios que se puede vender en la página son:

Productos:

- Cuencos
- Soga para saltar de entrenamiento
- Sahumerios
- Mancuernas
- (Productos similares)

Servicios:

- Clases de yoga
- Clases de funcional
- Evento de meditación
- (Diferentes tipos de clases y eventos relacionados)

No se incorporó ningún sistema de publicidad, pero esta es una característica futura a agregar.

Recursos

Herramientas de software

Todo el software utilizado:

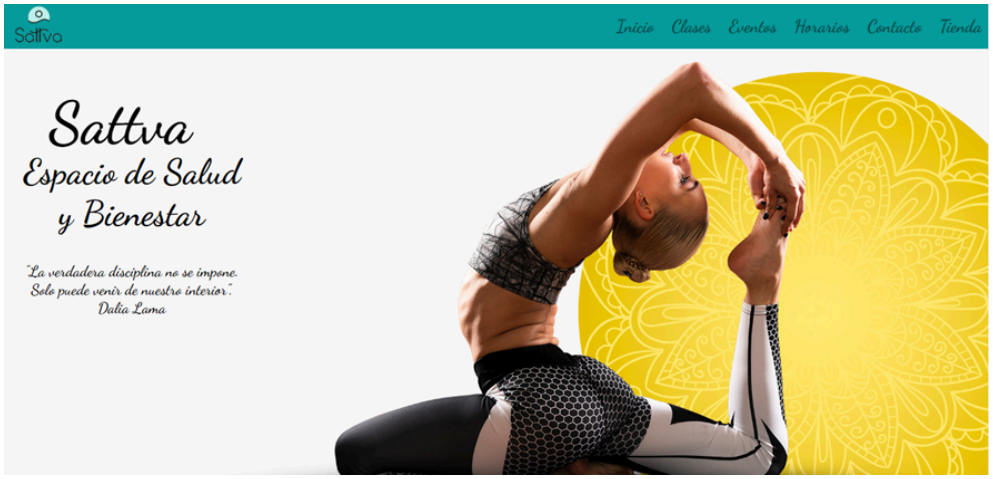
- Visual Studio Code
- MySQL Workbench
- Postman
- Vercel
- Railway

Prototipo

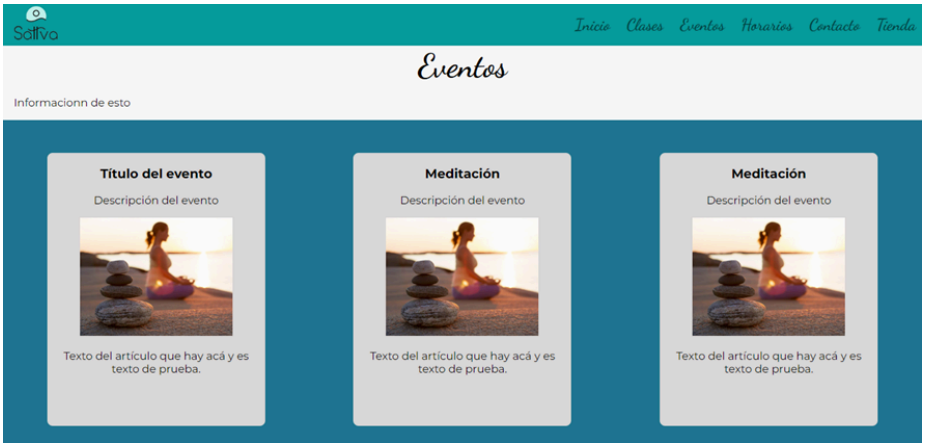
En primera instancia se desarrolló un prototipo básico de la aplicación Web únicamente utilizando HTML, CSS y JavaScript:

Primer prototipo:

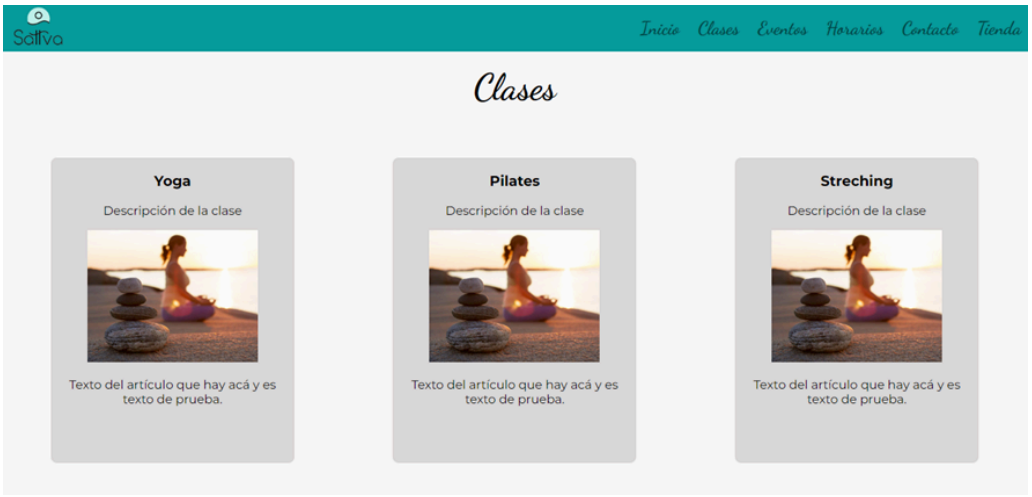
Página de Inicio (Un diseño básico con las funciones principales de la página ilustradas)



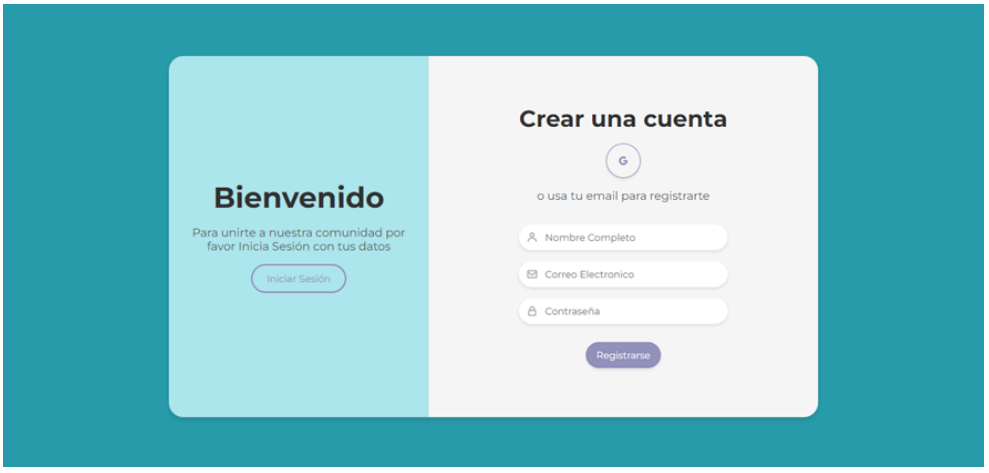
Lista de eventos disponibles en la página



Lista de clases disponibles en la página



Interfaz de login de usuario



Estructura del proyecto

(Se muestra cómo están organizados los directorios y archivos del proyecto para facilitar la comprensión de su funcionamiento y facilitar su mantenimiento)

```
/SattvaWebsite
    /backend
        /config
        /controllers
        /middleware
        /models
        /routes
        .env
        index.js
        package.json
        vercel.json
    /frontend
        /src
            /app
            /components
            /models
            /services
            app.component.css
            app.component.html
            app.component.ts
            app.module.ts
            /assets
            /environments
            index.html
            main.ts
            styles.css
        angular.json
    package.json
    tsconfig.json
```

Descripción de la estructura del proyecto.

Front-end:

- **Componentes:** Segmentos modulares que representan distintas partes de la UI, como formularios de producto, listas de órdenes, etc.
- **Servicios:** Lógica compartida para interactuar con la API del back-end, incluyendo la gestión de datos y autenticación.
- **Modelos:** Clases de objetos con sus propios atributos para manejar los datos con mayor facilidad.

Back-end:

- **Rutas:** Definen los endpoints de la API y las operaciones disponibles, como crear, actualizar y eliminar todo tipo de datos de la base de datos (productos, órdenes, clases, eventos, usuarios, instructores).
- **Controladores:** Gestionan la lógica de negocio, recibiendo solicitudes desde las rutas e interactuando con los modelos de datos.
- **Modelos:** Representaciones de las entidades del sistema (productos, usuarios, órdenes, clases, eventos), que definen la estructura y las relaciones de los datos en la base de datos. Cuentan con funciones que ejecutan código SQL para realizar diferentes operaciones sobre la base de datos.

Base de Datos:

- **Esquema y Tablas:** La base de datos MySQL está estructurada en tablas para diferentes entidades como usuarios, productos, órdenes, clases y eventos, con relaciones bien definidas para mantener la integridad referencial y facilitar consultas complejas. Cada tabla

almacena información específica de su respectivo entidad, y estas tablas se relacionan mediante claves foráneas.

El proyecto utiliza Angular para el front-end y Node.js con Express.js para el back-end. Para configurar el entorno, es necesario instalar Node.js y Angular CLI, configurar MySQL, y clonar el repositorio.

Descripción de cada módulo del sistema

Front-end:

- **Estructura del Proyecto Angular:** Compuesto por módulos, componentes y servicios, organizados para modularidad y reutilización.
- **Componentes Principales y su Funcionalidad:** Los componentes principales para el usuario incluyen, el login para registrarse e iniciar sesión en la página, el componente home, en donde hay información de las clases y eventos de Sattva, y la tienda, que muestra una lista de productos disponibles para comprar.
- **Servicios de Comunicación con el Back-end:** Servicios HTTP para interactuar con la API, manejando operaciones CRUD y autenticación.
- **Integración de Bootstrap para el Diseño:** Utilizado para estilos responsivos y componentes visuales.

Back-end:

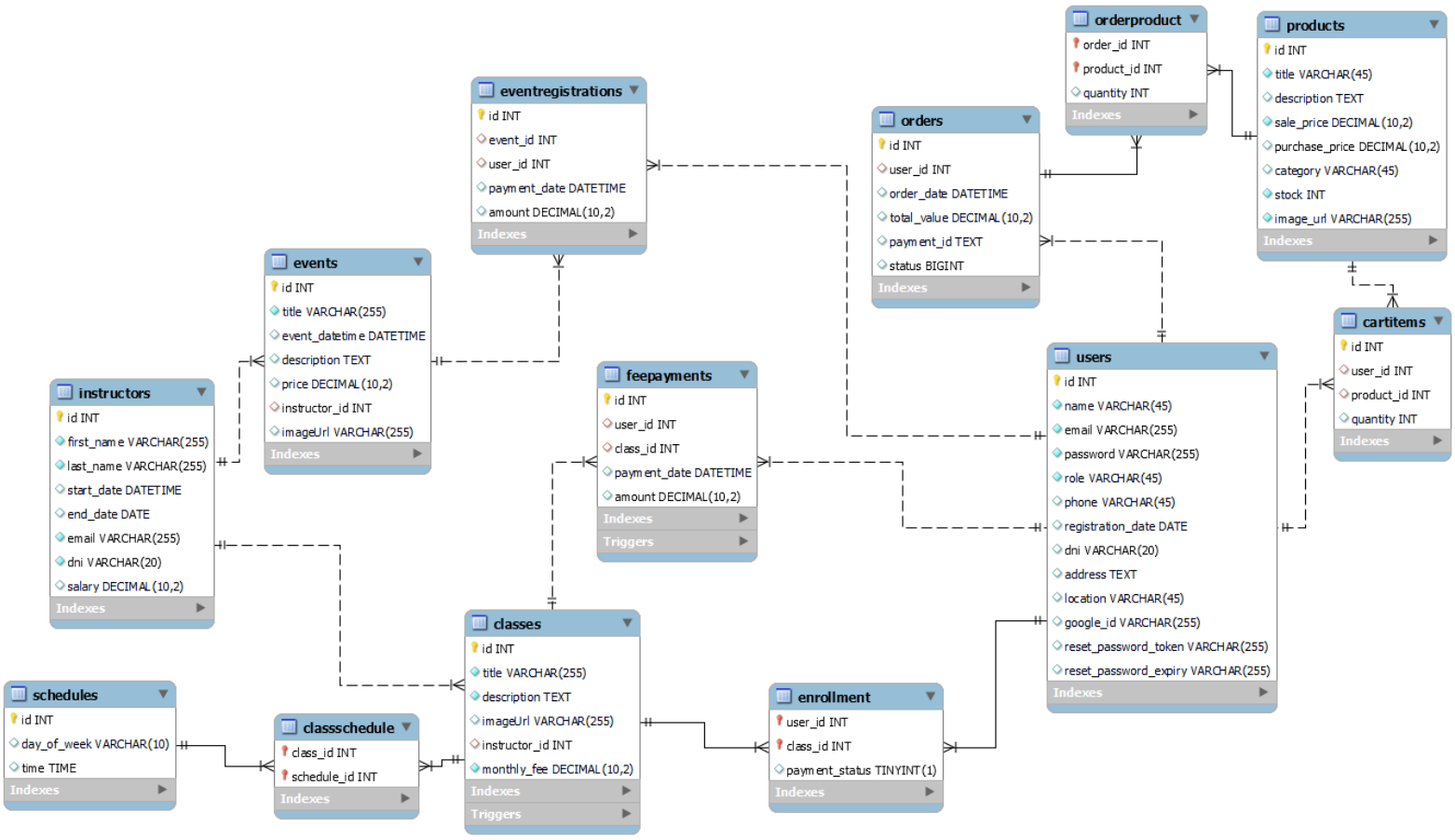
- **Configuración del Servidor con Node.js y Express.js:** Define rutas y maneja solicitudes HTTP.

- **Implementación de Rutas y Controladores:** Controladores procesan lógica de negocio, mientras que las rutas manejan endpoints de la API.
- **Conexión a la Base de Datos MySQL:** Utiliza mysql2, para poder ejecutar operaciones SQL para la gestión de la base de datos dentro de la base de datos MySQL.
- **Middleware y Manejo de Archivos:** Se utiliza Multer para cargar y gestionar archivos de imagen dentro del servidor local. Mientras que en producción se utiliza la base de datos de archivos de vercel (VercelBlob).

Base de Datos:

- **Diseño de la Base de Datos:** Esquema estructurado para manejar, por un lado, productos, usuarios y órdenes, y por otro, usuarios, clases, eventos e instructores con relaciones bien definidas.

Modelo de la base de datos MySQL



Scripts SQL de Creación de Tablas: Scripts para inicializar la base de datos con las tablas necesarias.

CREATE DATABASE IF NOT EXISTS `trainingcentersattva` /*!40100 DEFAULT CHARACTER SET utf8mb4 COLLATE utf8mb4_0900_ai_ci */ /*!80016 DEFAULT ENCRYPTION='N' */;

USE `trainingcentersattva`;

--

-- Table structure for table `cartitems`

--

DROP TABLE IF EXISTS `cartitems`;

CREATE TABLE `cartitems` (

```
  `id` int NOT NULL AUTO_INCREMENT,

  `user_id` int DEFAULT NULL,

  `product_id` int DEFAULT NULL,

  `quantity` int DEFAULT NULL,

  PRIMARY KEY (`id`),

  KEY `user_id` (`user_id`),

  KEY `product_id` (`product_id`),

    CONSTRAINT `cartitems_ibfk_1` FOREIGN KEY (`user_id`) REFERENCES `users` (`id`),

    CONSTRAINT `cartitems_ibfk_2` FOREIGN KEY (`product_id`) REFERENCES `products` (`id`)

) ENGINE=InnoDB AUTO_INCREMENT=65 DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci;

--

-- Table structure for table `classes`

--

DROP TABLE IF EXISTS `classes`;

CREATE TABLE `classes` (

  `id` int NOT NULL AUTO_INCREMENT,

  `title` varchar(255) NOT NULL,

  `description` text NOT NULL,

  `imageUrl` varchar(255) DEFAULT NULL,

  `instructor_id` int DEFAULT NULL,

  `monthly_fee` decimal(10,2) NOT NULL,

  PRIMARY KEY (`id`),

  KEY `classes_ibfk_1` (`instructor_id`),

    CONSTRAINT `classes_ibfk_1` FOREIGN KEY (`instructor_id`) REFERENCES `instructors` (`id`) ON DELETE SET NULL
```

```
) ENGINE=InnoDB AUTO_INCREMENT=27 DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci;


DELIMITER ;;

/*!50003 CREATE*/ /*!50017 DEFINER=`root`@`localhost`*/ /*!50003 TRIGGER `classes_BEFORE_DELETE` BEFORE DELETE ON `classes` FOR EACH ROW BEGIN

delete from `trainingcentersattva`.`classschedule` where class_id=OLD.id;

delete from `trainingcentersattva`.`enrollment` where class_id=OLD.id;

update feepayments set class_id = null where class_id=OLD.id;

END */;;

DELIMITER ;


--

-- Table structure for table `classschedule`

--

DROP TABLE IF EXISTS `classschedule`;

CREATE TABLE `classschedule` (

  `class_id` int NOT NULL,

  `schedule_id` int NOT NULL,

  PRIMARY KEY (`class_id`,`schedule_id`),

  KEY `classschedule_ibfk_2` (`schedule_id`),

    CONSTRAINT `classschedule_ibfk_1` FOREIGN KEY (`class_id`) REFERENCES `classes` (`id`) ON DELETE CASCADE,

   CONSTRAINT `classschedule_ibfk_2` FOREIGN KEY (`schedule_id`) REFERENCES `schedules` (`id`) ON DELETE CASCADE

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci;


--
```

```
-- Table structure for table `enrollment`

--

DROP TABLE IF EXISTS `enrollment`;

CREATE TABLE `enrollment` (

  `user_id` int NOT NULL,

  `class_id` int NOT NULL,

  `payment_status` tinyint(1) DEFAULT NULL,

  PRIMARY KEY (`user_id`,`class_id`),

  KEY `class_id` (`class_id`),

   CONSTRAINT `enrollment_ibfk_1` FOREIGN KEY (`user_id`) REFERENCES `users` (`id`),

  CONSTRAINT `enrollment_ibfk_2` FOREIGN KEY (`class_id`) REFERENCES `classes` (`id`)

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci;



--

-- Table structure for table `eventregistrations`

--

DROP TABLE IF EXISTS `eventregistrations`;

CREATE TABLE `eventregistrations` (

  `id` int NOT NULL AUTO_INCREMENT,

  `event_id` int DEFAULT NULL,

  `user_id` int DEFAULT NULL,

  `payment_date` datetime DEFAULT CURRENT_TIMESTAMP,

  `amount` decimal(10,2) DEFAULT NULL,

  PRIMARY KEY (`id`),

  KEY `event_id` (`event_id`),
```

```
  KEY `user_id` (`user_id`),

   CONSTRAINT `eventregistrations_ibfk_1` FOREIGN KEY (`event_id`) REFERENCES `events` (`id`),

   CONSTRAINT `eventregistrations_ibfk_2` FOREIGN KEY (`user_id`) REFERENCES `users` (`id`)

) ENGINE=InnoDB AUTO_INCREMENT=9 DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci;



--

-- Table structure for table `events`

--

DROP TABLE IF EXISTS `events`;

CREATE TABLE `events` (

  `id` int NOT NULL AUTO_INCREMENT,

  `title` varchar(255) NOT NULL,

  `event_datetime` datetime DEFAULT NULL,

  `description` text,

  `price` decimal(10,2) DEFAULT NULL,

  `instructor_id` int DEFAULT NULL,

  `imageUrl` varchar(255) DEFAULT NULL,

  PRIMARY KEY (`id`),

  KEY `events_ibfk_1` (`instructor_id`),

    CONSTRAINT `events_ibfk_1` FOREIGN KEY (`instructor_id`) REFERENCES `instructors` (`id`) ON DELETE SET NULL

) ENGINE=InnoDB AUTO_INCREMENT=8 DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci;



--

-- Table structure for table `feepayments`
```

```
--

DROP TABLE IF EXISTS `feepayments`;

CREATE TABLE `feepayments` (

  `id` int NOT NULL AUTO_INCREMENT,

  `user_id` int DEFAULT NULL,

  `class_id` int DEFAULT NULL,

  `payment_date` datetime DEFAULT CURRENT_TIMESTAMP,

  `amount` decimal(10,2) DEFAULT NULL,

  PRIMARY KEY (`id`),

  KEY `user_id` (`user_id`),

  KEY `class_id` (`class_id`),

   CONSTRAINT `feepayments_ibfk_1` FOREIGN KEY (`user_id`) REFERENCES `users` (`id`),

    CONSTRAINT `feepayments_ibfk_2` FOREIGN KEY (`class_id`) REFERENCES `classes` (`id`)

) ENGINE=InnoDB AUTO_INCREMENT=9 DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci;


DELIMITER ;;

/*!50003 CREATE*/ /*!50017 DEFINER=`root`@`localhost`*/ /*!50003 TRIGGER `feepayments_AFTER_INSERT` AFTER INSERT ON `feepayments` FOR EACH ROW BEGIN

INSERT INTO enrollment (user_id, class_id, payment_status)

VALUES (NEW.user_id, NEW.class_id, true)

ON DUPLICATE KEY UPDATE payment_status = true;

END */;;

DELIMITER ;


--
```

```
-- Table structure for table `instructors`

--

DROP TABLE IF EXISTS `instructors`;

CREATE TABLE `instructors` (

  `id` int NOT NULL AUTO_INCREMENT,

  `first_name` varchar(255) NOT NULL,

  `last_name` varchar(255) NOT NULL,

  `start_date` datetime DEFAULT CURRENT_TIMESTAMP,

  `end_date` date DEFAULT NULL,

  `email` varchar(255) NOT NULL,

  `dni` varchar(20) NOT NULL,

  `salary` decimal(10,2) DEFAULT NULL,

  PRIMARY KEY (`id`),

  UNIQUE KEY `dni` (`dni`)

) ENGINE=InnoDB AUTO_INCREMENT=4 DEFAULT CHARSET=utf8mb4
COLLATE=utf8mb4_0900_ai_ci;



--

-- Table structure for table `orderproduct`

--

DROP TABLE IF EXISTS `orderproduct`;

CREATE TABLE `orderproduct` (

  `order_id` int NOT NULL,

  `product_id` int NOT NULL,

  `quantity` int DEFAULT NULL,

  PRIMARY KEY (`order_id`,`product_id`),

  KEY `product_id` (`product_id`),
```

```
    CONSTRAINT `orderproduct_ibfk_1` FOREIGN KEY (`order_id`) REFERENCES `orders` (`id`),
   CONSTRAINT `orderproduct_ibfk_2` FOREIGN KEY (`product_id`) REFERENCES `products` (`id`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci;


--
-- Table structure for table `orders`
--
DROP TABLE IF EXISTS `orders`;
CREATE TABLE `orders` (
  `id` int NOT NULL AUTO_INCREMENT,
  `user_id` int DEFAULT NULL,
  `order_date` datetime DEFAULT CURRENT_TIMESTAMP,
  `total_value` decimal(10,2) DEFAULT NULL,
  `payment_id` text,
  `status` bigint DEFAULT '0',
  PRIMARY KEY (`id`),
  KEY `user_id` (`user_id`),
  CONSTRAINT `orders_ibfk_1` FOREIGN KEY (`user_id`) REFERENCES `users` (`id`)
) ENGINE=InnoDB AUTO_INCREMENT=87 DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci;


--
-- Table structure for table `products`
--
DROP TABLE IF EXISTS `products`;
CREATE TABLE `products` (
```

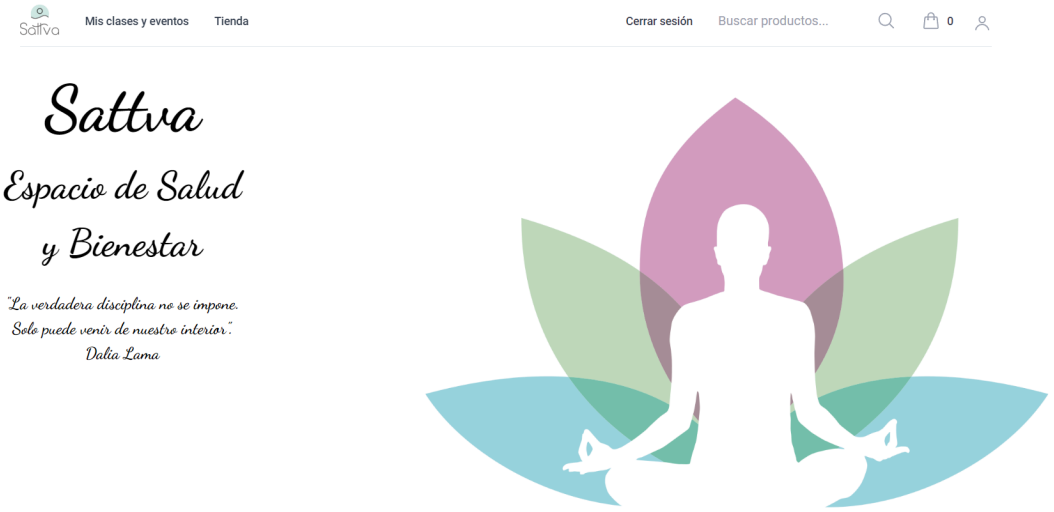
```
  `id` int NOT NULL AUTO_INCREMENT,
  `title` varchar(45) NOT NULL,
  `description` text,
  `sale_price` decimal(10,2) NOT NULL,
  `category` varchar(45) DEFAULT NULL,
  `stock` int NOT NULL,
  `image_url` varchar(255) NOT NULL,
  PRIMARY KEY (`id`)
) ENGINE=InnoDB AUTO_INCREMENT=5 DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci;


--
-- Table structure for table `schedules`
--
DROP TABLE IF EXISTS `schedules`;
CREATE TABLE `schedules` (
  `id` int NOT NULL AUTO_INCREMENT,
  `day_of_week` varchar(10) DEFAULT NULL,
  `time` time DEFAULT NULL,
  PRIMARY KEY (`id`)
) ENGINE=InnoDB AUTO_INCREMENT=37 DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci;


--
-- Table structure for table `users`
--
DROP TABLE IF EXISTS `users`;
```

```
CREATE TABLE `users` (
  `id` int NOT NULL AUTO_INCREMENT,
  `name` varchar(45) NOT NULL,
  `email` varchar(255) NOT NULL,
  `password` varchar(255) NOT NULL,
  `role` varchar(45) NOT NULL DEFAULT 'client',
  `phone` varchar(45) DEFAULT NULL,
  `registration_date` date DEFAULT NULL,
  `dni` varchar(20) DEFAULT NULL,
  `address` text,
  `location` varchar(45) DEFAULT NULL,
  `google_id` varchar(255) DEFAULT NULL,
  `reset_password_token` varchar(255) DEFAULT NULL,
  `reset_password_expiry` varchar(255) DEFAULT NULL,
  PRIMARY KEY (`id`),
  UNIQUE KEY `email` (`email`)
) ENGINE=InnoDB AUTO_INCREMENT=4 DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci;


--
-- Dumping events for database 'trainingcentersattva'
--
/*!50106 DROP EVENT IF EXISTS `DeletePastEvents` */;
DELIMITER ;;
/*!50106 CREATE*/ /*!50117 DEFINER=`root`@`localhost`*/ /*!50106 EVENT `DeletePastEvents` ON SCHEDULE EVERY 1 DAY STARTS '2024-07-26 18:02:18' ON COMPLETION NOT PRESERVE ENABLE DO DELETE FROM Events
  WHERE event_datetime < NOW() */ ;;
```

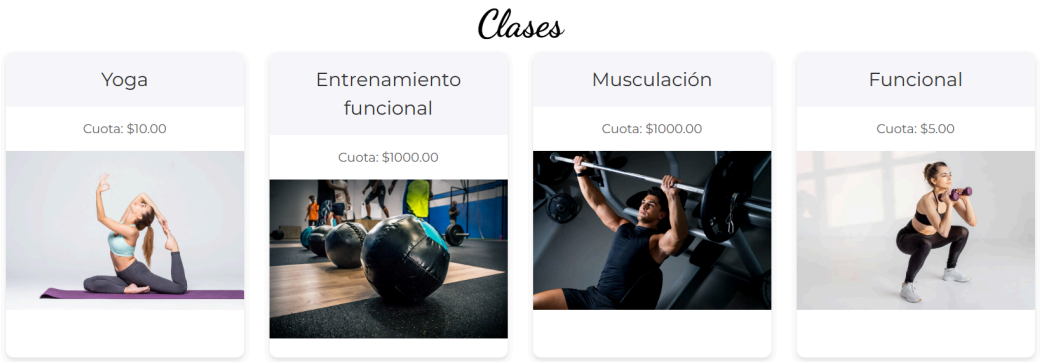
```
/*!50106 DROP EVENT IF EXISTS `UpdatePaymentStatus` */;;

DELIMITER ;;

/*!50106 CREATE*/ /*!50117 DEFINER=`root`@`localhost`*/ /*!50106 EVENT `UpdatePaymentStatus` ON SCHEDULE EVERY 1 DAY STARTS '2024-07-26 17:58:18' ON COMPLETION NOT PRESERVE ENABLE DO UPDATE Enrollment e

  JOIN FeePayments fp ON e.user_id = fp.user_id AND e.class_id = fp.class_id

  SET e.payment_status = FALSE

  WHERE DATEDIFF(CURDATE(), DATE(fp.payment_date)) > 30 */ ;;

DELIMITER ;
```

Diseño Final del proyecto

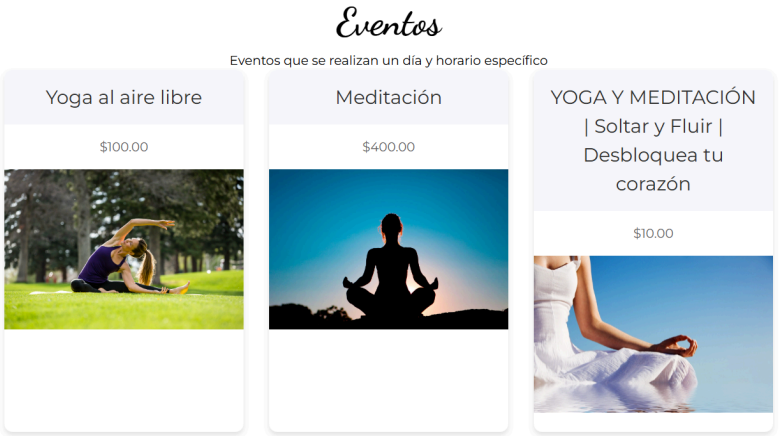
Título (Pantalla de Inicio): En esta pantalla se puede ver un título y una imágen para el diseño estético de la página. Cuenta con una barra superior que nos permite navegar a través de toda la aplicación.



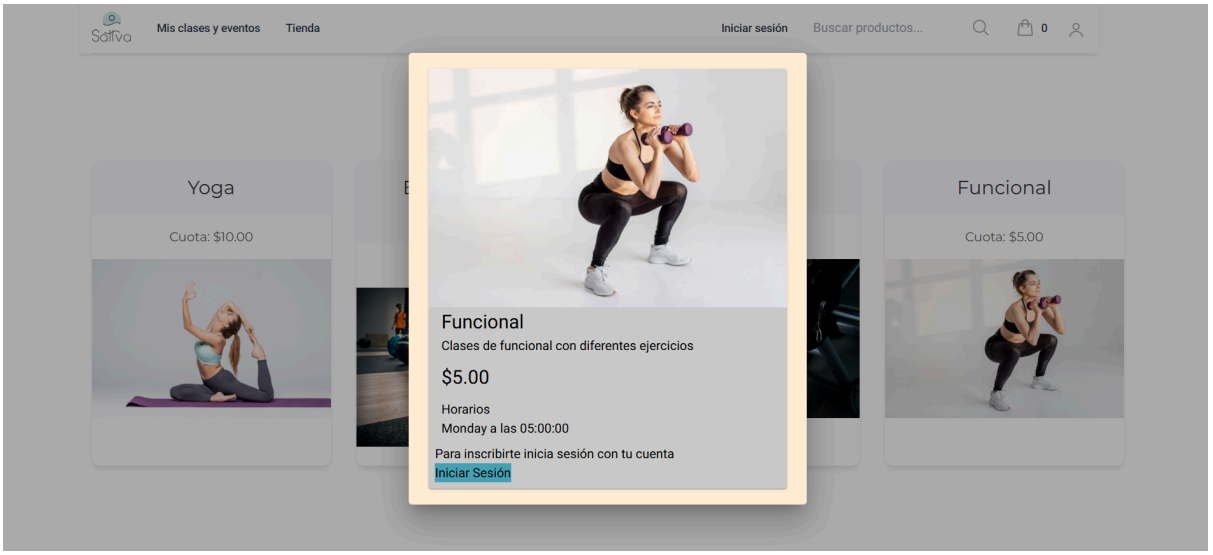
Lista de Clases: Esta consiste en una lista de diferentes tipos de clases disponibles que se ofrecen a los usuarios para que estos puedan inscribirse.



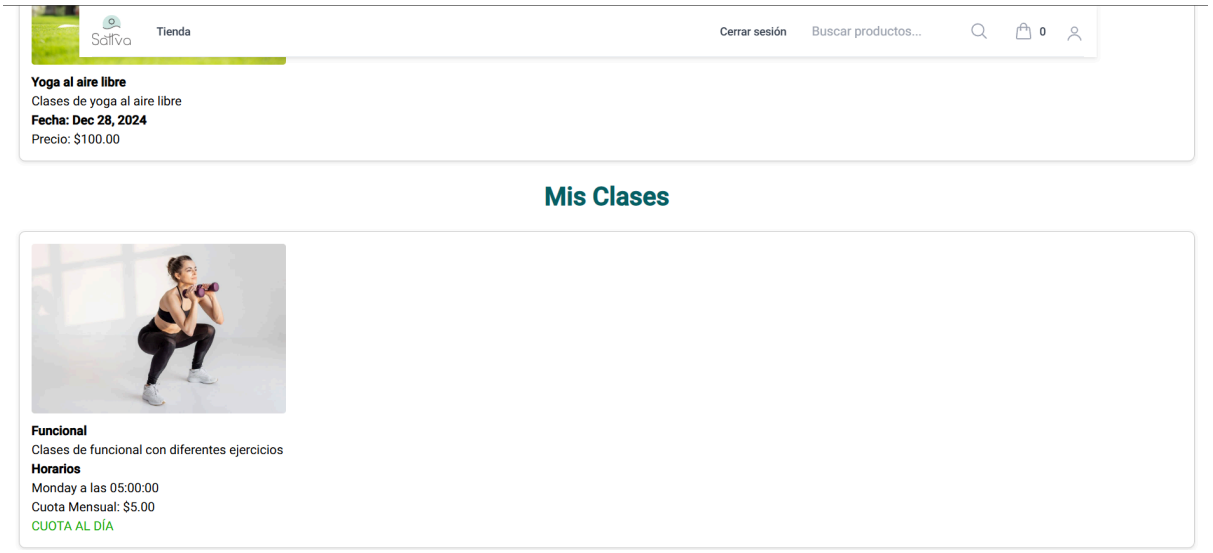
Lista de Eventos: Esta consiste en una lista de diferentes tipos de eventos disponibles que se ofrecen a los usuarios para que estos puedan inscribirse.



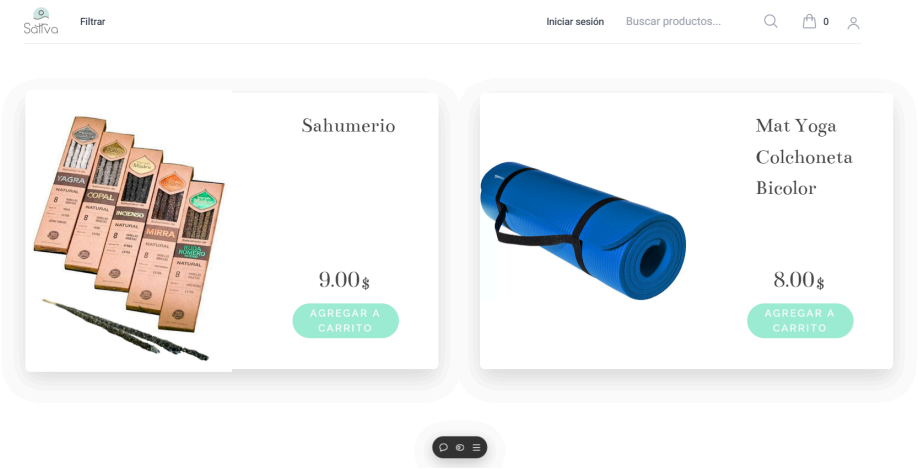
Detalles de clase/evento: Esta pantalla se muestra cuando se presiona una clase o evento, mostrando una descripción de esta y la posibilidad de inscribirse si se está registrado en la página.



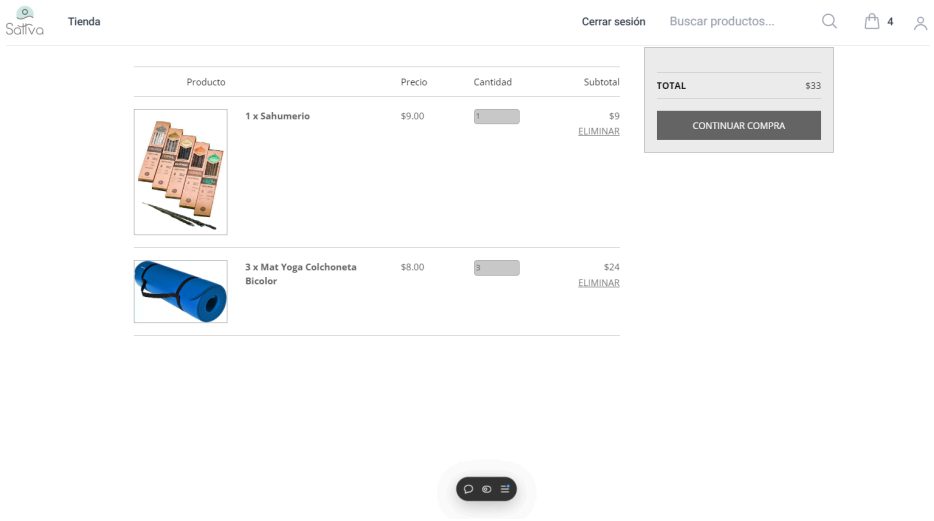
Lista clases y eventos inscriptos: Consiste en una lista de clases y eventos a los que el usuario que está utilizando la página está inscripto.



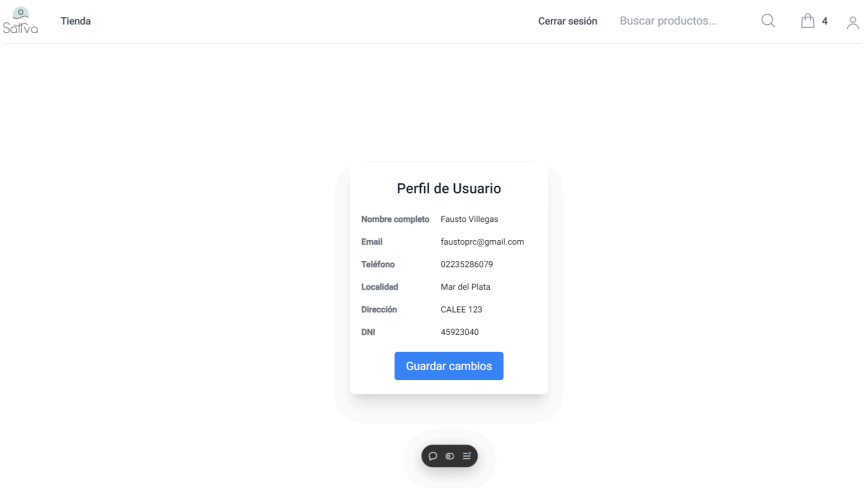
Tienda: Esta pantalla muestra una lista de productos disponibles para que los usuarios que deseen puedan agregarlos al carrito y continuar con su compra.



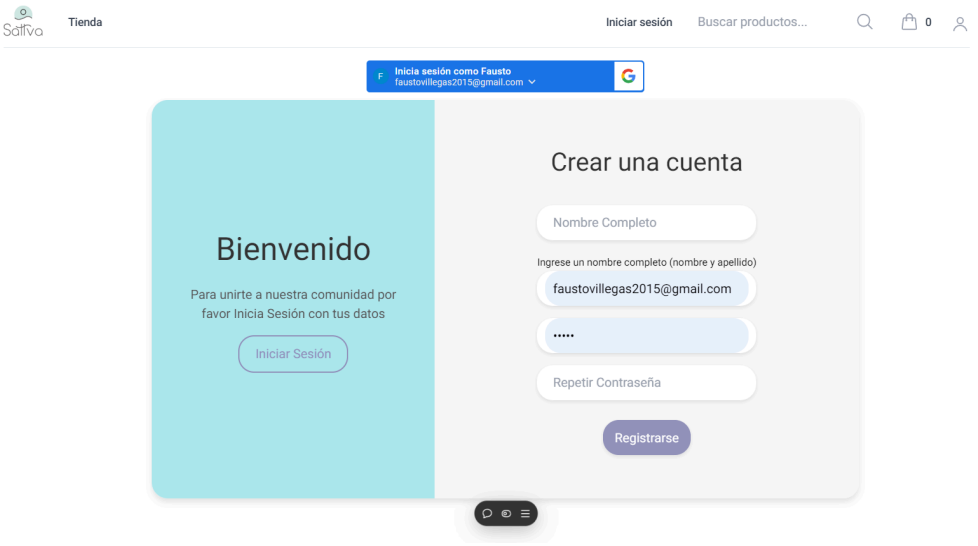
Carrito: En esta pantalla se muestran los productos que el usuario decidió agregar al carrito para poder continuar con su compra.

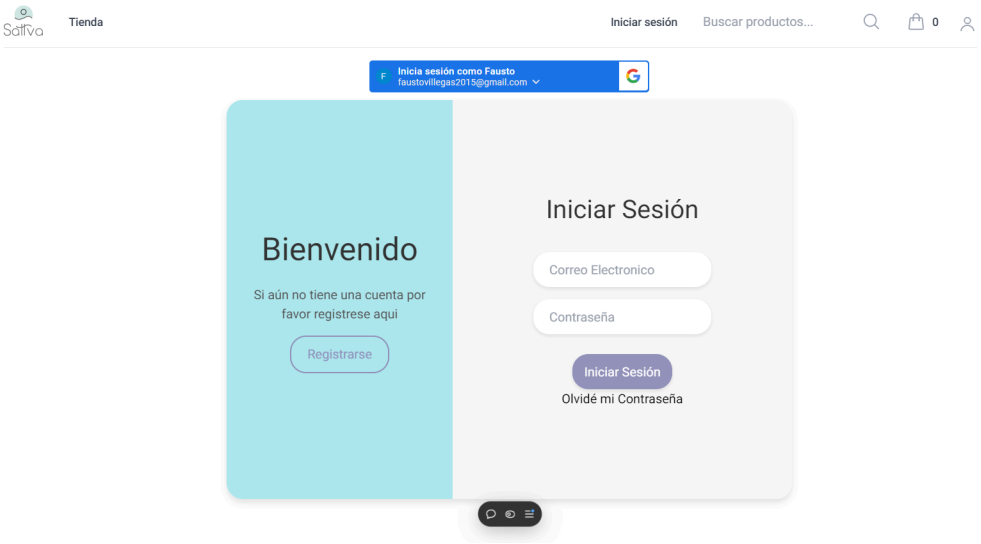


Perfil: Muestra información básica del perfil



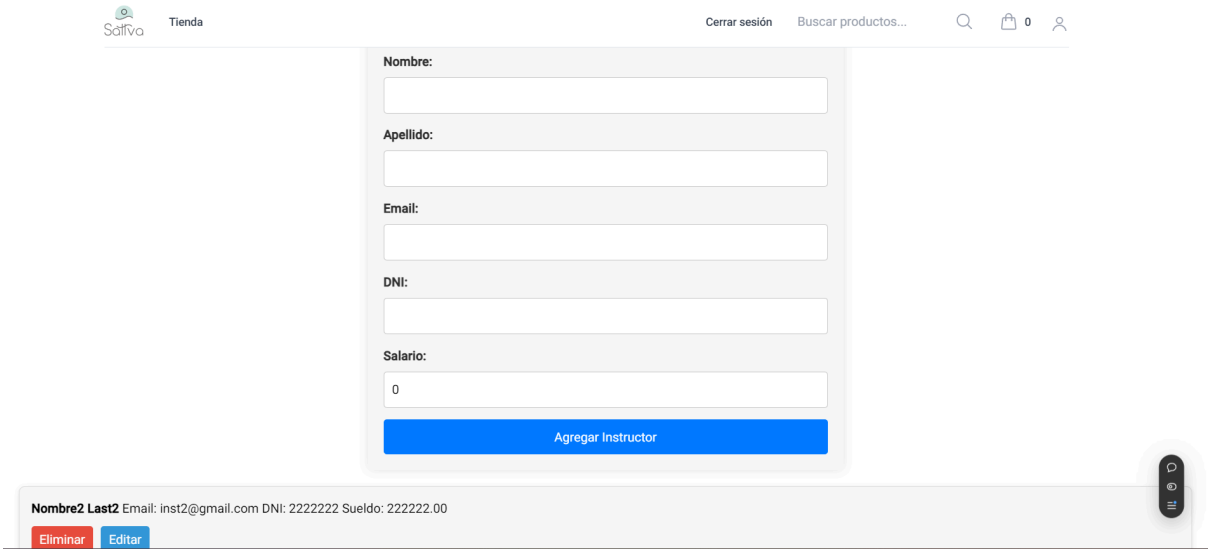
Login: Pantalla que permite al usuario Iniciar Sesión o Registrarse en la aplicación

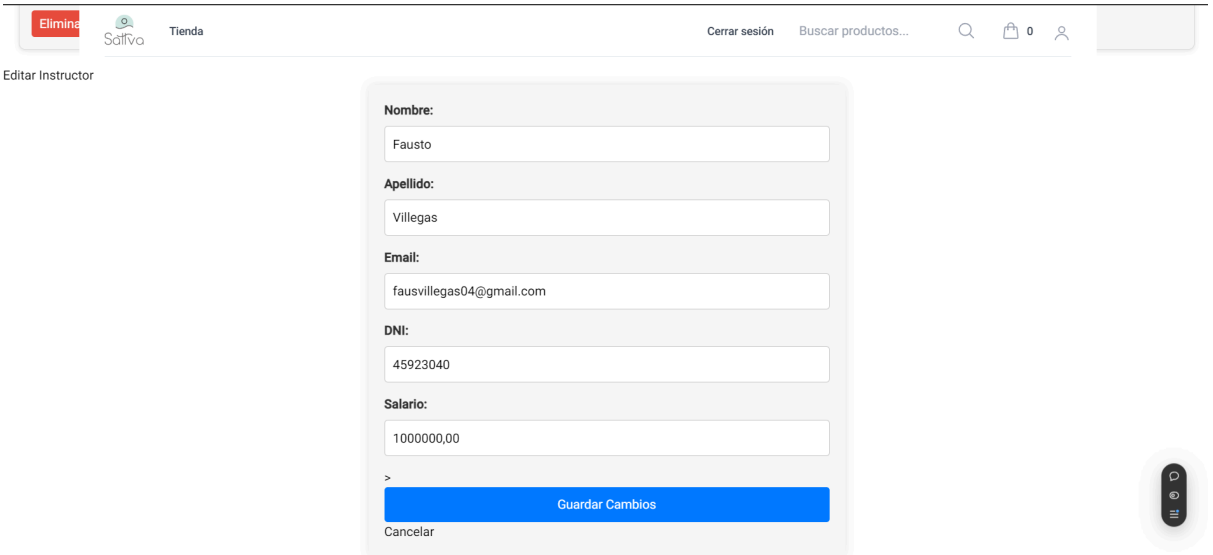




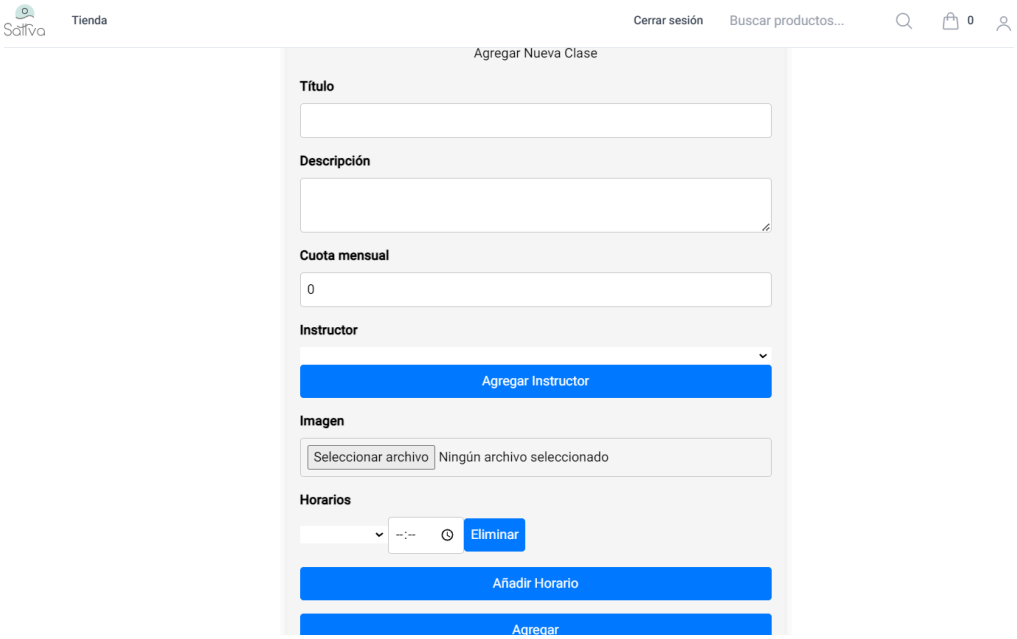
Vista de administrador

Agregar, eliminar y modificar datos de instructores:

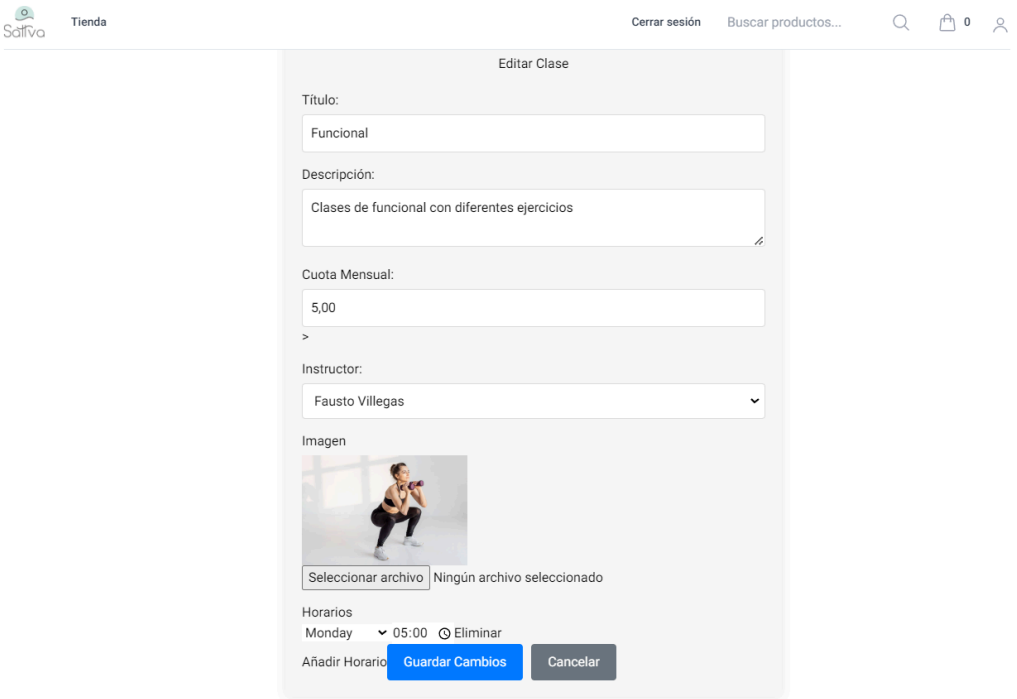




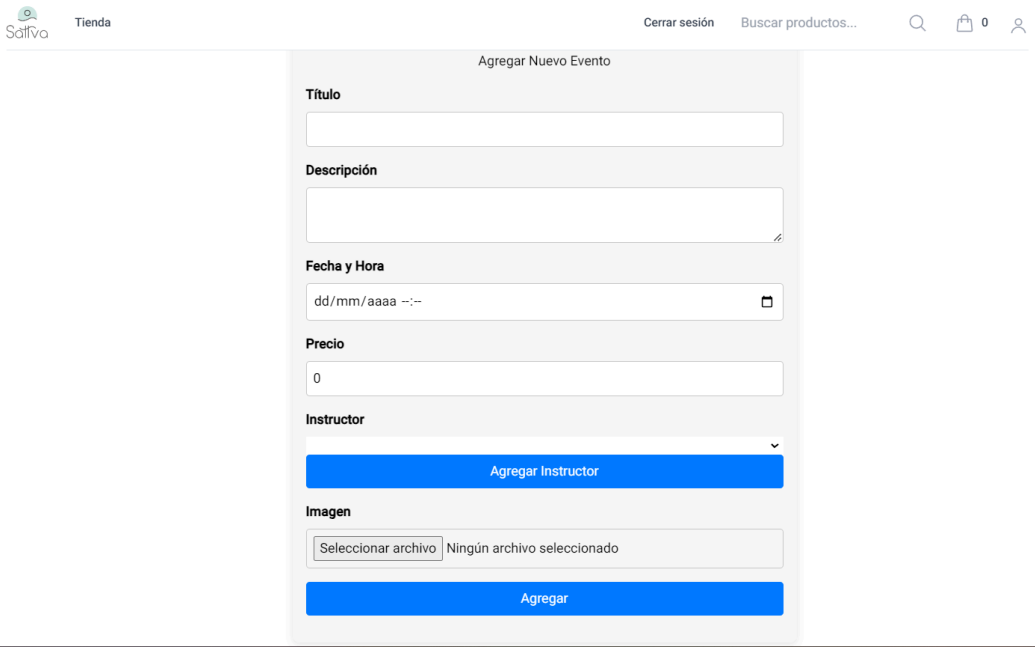
Agregar nueva clase:



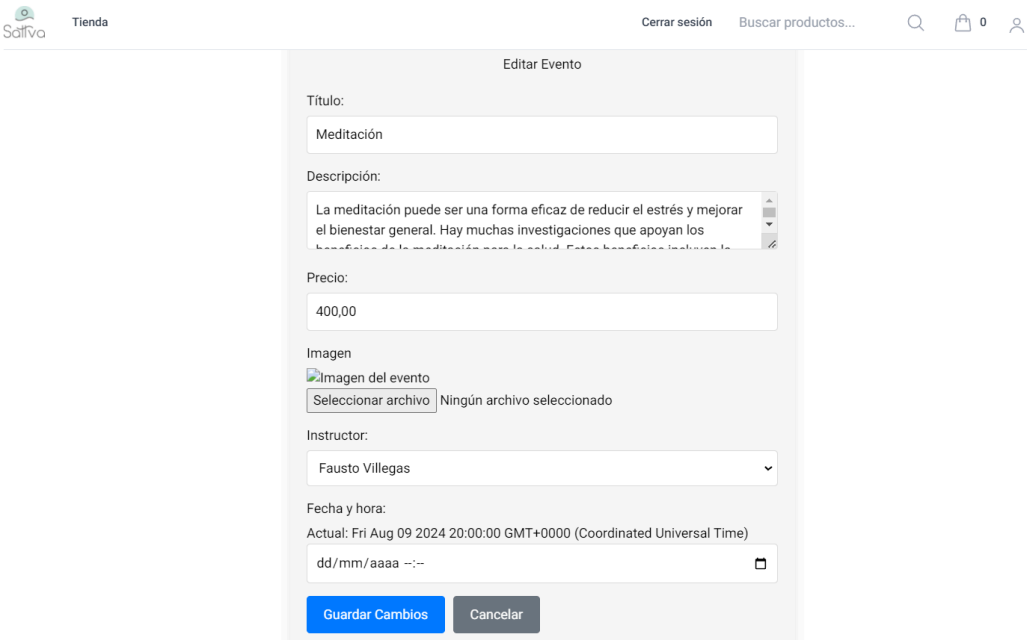
Modificar datos de clase:



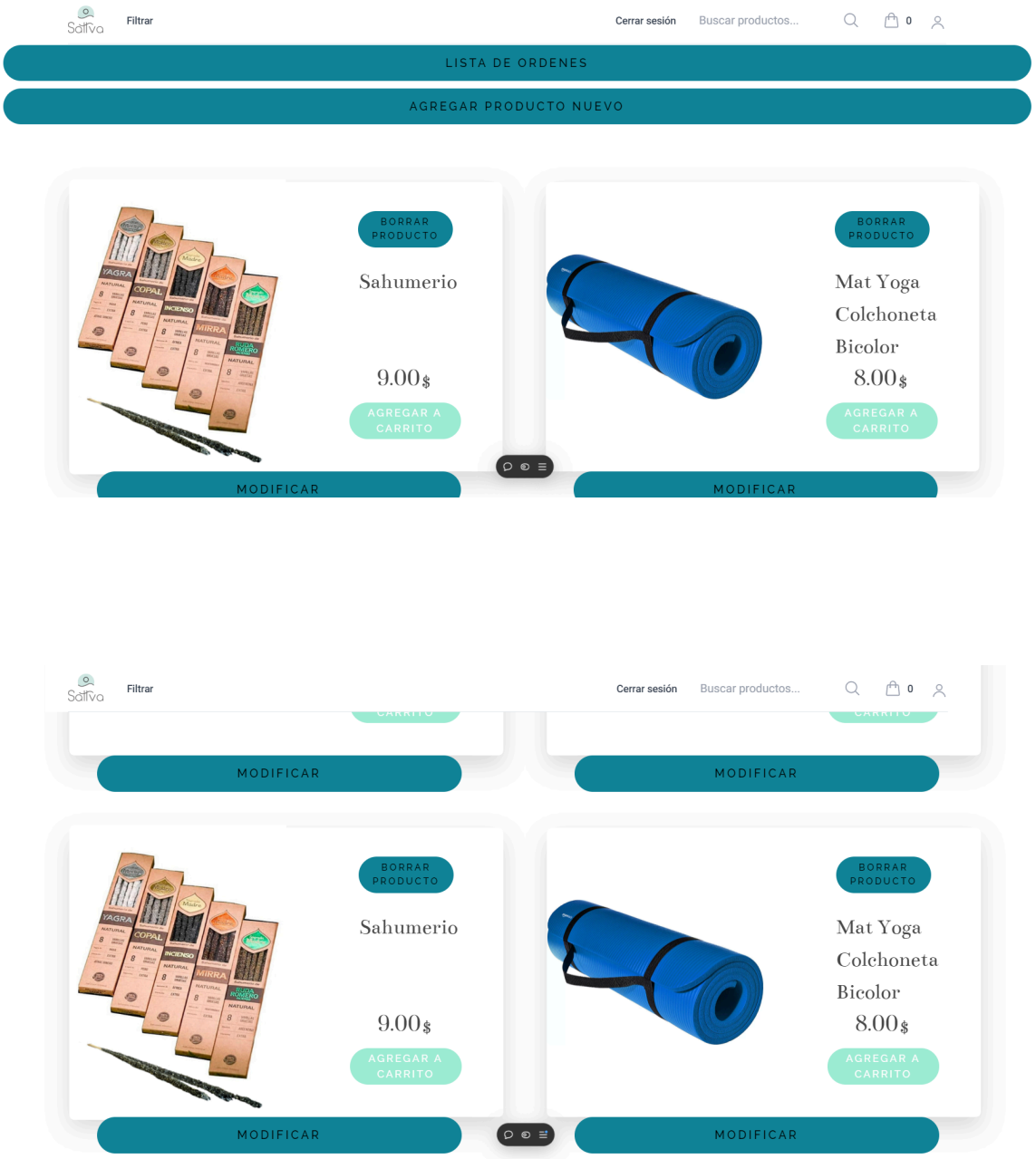
Agregar nuevo evento:



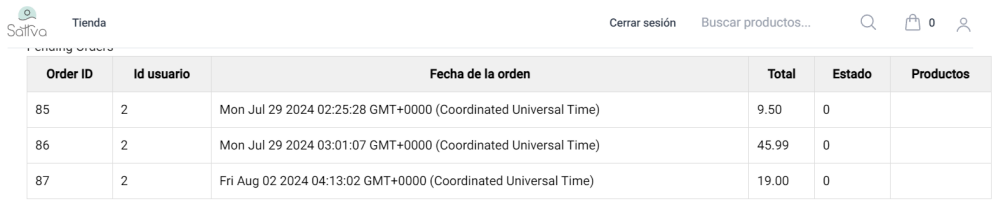
Editar evento:



Tienda:

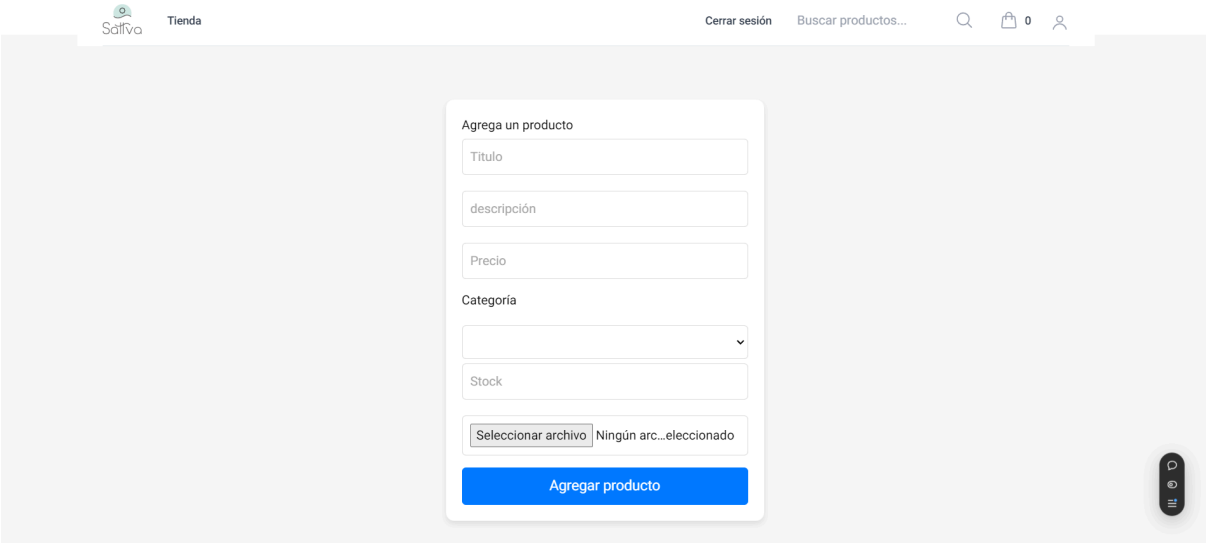

Sattva
Filtrar
Cerrar sesión
Buscar productos...
0
LISTA DE ORDENES
AGREGAR PRODUCTO NUEVO
BORRAR PRODUCTO
Sahumerio
9.00$
AGREGAR A CARRITO
MODIFICAR
BORRAR PRODUCTO
Mat Yoga
Colchoneta
Bicolor
8.00$
AGREGAR A CARRITO
MODIFICAR
Sattva
Filtrar
Cerrar sesión
Buscar productos...
0
MODIFICAR
MODIFICAR
BORRAR PRODUCTO
Sahumerio
9.00$
AGREGAR A CARRITO
MODIFICAR
BORRAR PRODUCTO
Mat Yoga
Colchoneta
Bicolor
8.00$
AGREGAR A CARRITO
MODIFICAR

Lista de órdenes de productos:

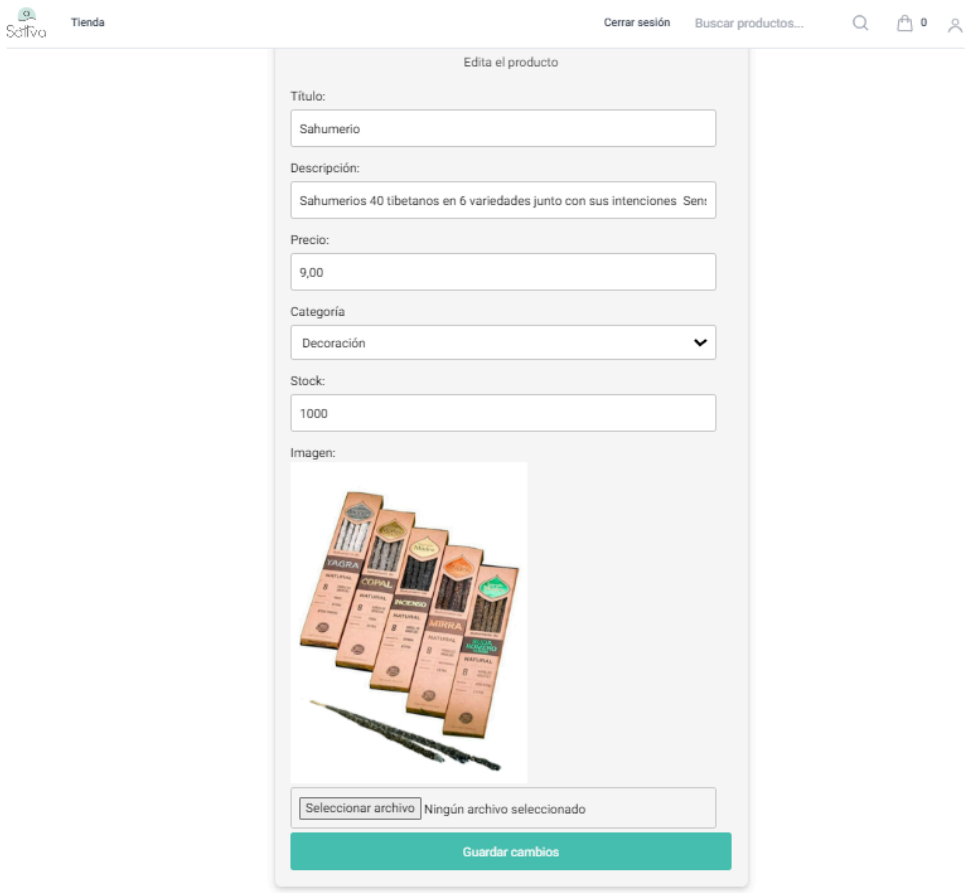


Order ID	Id usuario	Fecha de la orden	Total	Estado	Productos
85	2	Mon Jul 29 2024 02:25:28 GMT+0000 (Coordinated Universal Time)	9.50	0	
86	2	Mon Jul 29 2024 03:01:07 GMT+0000 (Coordinated Universal Time)	45.99	0	
87	2	Fri Aug 02 2024 04:13:02 GMT+0000 (Coordinated Universal Time)	19.00	0	

Agregar nuevo producto



Modificar datos producto:



Para continuar viendo ingresar a: https://web-sattva.vercel.app/

Pruebas

Para las pruebas de las funciones de la página se crearon diferentes usuarios y se realizaron con estos todas las opciones que un usuario podría realizar en una práctica real, mediante estos usuarios se probó realizar el registro de su cuenta, el inicio de sesión, la inscripción a clases pagando una cuota, la inscripción a un evento, la compra de una orden de diversos productos ficticios.

Se creó un usuario administrador para que esté, al iniciar sesión, pueda realizar diversas modificaciones a los datos y acceder a información que un usuario regular no puede. Este usuario administrador puede crear, eliminar y modificar la información de productos, clases y eventos. Además podrá acceder a información como los instructores que están registrados, podrá eliminar, agregar y modificar a estos, y podrá acceder a una lista de órdenes que realizaron los usuarios que dichas órdenes no fueron entregadas aún (status = false).

Para las pruebas de pagos mediante Mercado Pago se utilizaron cuentas de prueba o test, un comprador y un vendedor. En donde el vendedor asemeja al dueño de Sattva y el comprador a cualquier usuario registrado que desea realizar una compra.

Seguridad

Medidas de seguridad implementadas.

- **Encriptación de contraseñas con bcryptjs**: Para proteger la información sensible de los usuarios, las contraseñas se almacenan de forma segura utilizando bcryptjs. Este método aplica un hash a las contraseñas, lo que las hace difíciles de descifrar si se produce una violación de datos.

- **Validación de Datos con express-validator**: Para garantizar que los datos ingresados por los usuarios sean seguros y válidos, se utiliza express-validator. Esta librería ayuda a sanitizar y validar los datos, previniendo posibles ataques como inyecciones SQL y XSS.

Consideraciones de Seguridad en el Almacenamiento de Archivos

Se toman medidas para proteger los archivos subidos, como el uso de nombres de archivo únicos y la validación del tipo de archivo. Esto ayuda a prevenir ataques que podrían comprometer el sistema a través de archivos maliciosos.

Resultados y Discusión

Análisis de los resultados obtenidos.

El proyecto "WebsiteSattva" ha logrado crear una plataforma funcional para la gestión de un centro de entrenamiento. Se implementan funcionalidades clave como la administración de productos, clases, eventos, instructores y usuarios, así como la gestión de órdenes y pagos.

Comparación con los objetivos planteados.

Los objetivos principales de desarrollar un sistema de gestión integral y una interfaz de usuario amigable se han cumplido. Sin embargo, áreas como la seguridad y la optimización de rendimiento podrían mejorarse.

Ventajas y limitaciones del sistema desarrollado.

Ventajas:

- Facilidad de uso y gestión centralizada.
- Eficiencia en el manejo de datos de usuarios, clases y órdenes.
- Capacidad de edición de productos, clases, eventos e instructores por parte de los usuario de tipo Administrador.
- Facilidad en la gestión de usuarios y administradores.

Limitaciones:

- Falta de pruebas extensivas de seguridad y escalabilidad.
- Necesidad de mejorar la interfaz de usuario para una mejor experiencia.

Estas observaciones destacan la efectividad del sistema desarrollado y proporcionan una base sólida para futuras mejoras y optimizaciones.

Manual del desarrollador de instalación y despliegue

Este manual proporciona una guía detallada para desplegar la aplicación "Website Sattva", tanto el frontend como el backend. Siguiendo estos pasos, podrás configurar y ejecutar la aplicación en tu entorno local y desplegarla en producción usando Vercel y Railway.

1. **Requisitos Previos**

Node.js (v14 o superior)

npm (v6 o superior)

Angular CLI (v16 o superior)

Cuenta en Vercel

Cuenta en Railway

2. **Configuración del Entorno**

Clona el repositorio del proyecto:

- git clone https://github.com/tu-usuario/website-sattva.git
- cd website-sattva

3. **Despliegue del Backend**

    a. **Instalación de Dependencias del Backend**

Navega al directorio del backend e instala las dependencias:

- cd backend npm install

    b. **Configuración del Archivo .env**

Crea un archivo .env en el directorio del backend y agrega las variables de entorno necesarias, por ejemplo:

- PORT=3000
- DATABASE_URL=mysql://user:password@host:port/database
- JWT_SECRET=your_secret_key

    c. **Ejecutar el Servidor**

Inicia el servidor en modo desarrollo:

- npm run dev

4. **Despliegue del Frontend**

    a. **Instalación de Dependencias del Frontend**

Navega al directorio del frontend e instala las dependencias:

- cd ../frontend
- npm install

    b. **Construir la Aplicación**

Construye la aplicación Angular:

- npm run build

c. Servir la Aplicación

Inicia el servidor de desarrollo de Angular:

- npm start

Manual de uso de usuario

Este ayudará a los usuarios a navegar y utilizar todas las funcionalidades del sitio web.

Registro e Inicio de Sesión

Registro de usuario

1. Acceda a la página principal de Web Sattva.
2. Haga clic en "Registrarse" en la esquina superior derecha.
3. Complete el formulario con su nombre, correo electrónico y una contraseña segura.
4. Haga clic en "Crear cuenta".

Inicio de Sesión

1. Acceda a la página principal de Web Sattva.
2. Haga clic en "Iniciar sesión" en la esquina superior derecha.
3. Ingresar a el formulario de inicio de sesión presionando Iniciar Sesion
4. Ingrese su correo electrónico y contraseña.

5. Haga clic en "Iniciar Sesión".

Navegación por el Sitio

El sitio está organizado en varias secciones principales:

- **Inicio**: Página principal con las diferentes clases y eventos dispoinbles.
- **Productos**: Catálogo completo de productos.
- **Carrito**: Productos seleccionados para la compra.
- **Cuenta**: Información y configuración de su cuenta.

Inscripción a clases y eventos

Dentro de la página de Inicio se verán unas listas de clases y eventos disponibles.

1. Seleccionar la clase/evento que se desea inscribir.
2. Seleccionar el botón “Inscribirse y Pagar”.
3. Aparecerá un botón de Mercado Pago, seleccione este botón y lo dirigirá a una interfaz de pago de Mercado Pago.

Compra de productos

Para acceder al catálogo de productos disponibles presiona en el botón ‘Tienda’ de la barra superior.

1. Dentro de cada producto tendrá un botón “Agregar a carrito” que agrega un producto al carrito.
2. Seleccionar un producto para ver sus detalles y agrega al carrito la cantidad que deseas de ese producto.
3. Acceder al carrito mediante el botón “cart” de la barra superior.
4. Presionar continuar compra.
5. Aparecerá un botón de Mercado Pago, seleccione este botón y lo dirigirá a una interfaz de pago de Mercado Pago.

Conclusiones

Este proyecto fue un proyecto desafiante ya que fue mi primer proyecto Full-stack en el que se debe desarrollar tanto frontend, backend y diseñar la base de datos. Fui aprendiendo el manejo de muchas de las tecnologías que fueron utilizadas y estos conocimientos podrán ser muy útiles para proyectos futuros.

A pesar de poder realizar mejoras al proyecto, considero que el estado de este es considerablemente completo y con una complejidad suficiente en relación a los conocimientos adquiridos durante la carrera de Tecnicatura en programación.

Algunas de las posibles mejoras del proyecto son un uso mejor de las variables y organización en el proyecto, funciones más completas y con una interfaz más llamativa. Para futuros proyectos se tendrá una mejor organización de las tareas que se deben realizar y un mayorn conocimiento de cómo utilizarlas desde un principio.

Al finalizar este proyecto noté que mis habilidades en desarrollo tanto de frontend como de backend mejoraron, ya que tuve que investigar y aprender nuevas tecnologías.

Anexos

Código fuente principal.

El código fuente del proyecto está disponible en el repositorio de GitHub (https://github.com/FausVillegas/WebSattva.git), donde se pueden revisar los archivos principales del frontend y backend.

Diagramas adicionales.

En una primera instancia se diseñó el modelo relacional para este proyecto cuando solo se estaba realizando el primer análisis, luego con el tiempo fui adaptando y modificando este diseño para satisfacer las necesidades que me iban surgiendo.

Modelo Relacional

R' = id-pedido, id-producto, id-pago-cuota, id-evento

F = vacio

CC = id-pedido, id-producto, id-pago-cuota, id-evento

R1 = id-usuario, nombre-apellido, correo-electronico, contraseña, fecha-registro, documento, domicilio.

F = {

id-usuario --> nombre-apellido, correo-electronico, contraseña, fecha-registro, documento, domicilio;

}

CC = id-usuario

R2 = id-clase, nombre-clase, descripción-clase, id-dia-horario, id-profesor/a;

F = {

id-clase --> nombre-clase, descripción-clase, id-dia-horario, id-profesor/a;

}

CC= id-clase

R3 = id-dia-horario, dia-semanal, horario;

F = {

id-dia-horario --> dia-semanal, horario;

}

CC = id-dia-horario

R4 = id-profesor/a, nombre-apellido-prof, fecha-ingreso, fecha-egreso, email-prof, dni-prof, sueldo-prof;

F = {

id-profesor/a --> nombre-apellido-prof, fecha-ingreso, fecha-egreso, email-prof, dni-prof, sueldo-prof;

}

CC = id-profesor/a

R5 = id-pedido, fecha-pedido, valor-total-pedido, id-usuario;

F = {

id-pedido --> fecha-pedido, valor-total-pedido, id-usuario;

}

CC = id-pedido

R6 = id-producto, titulo-prod, descripción-prod, precio-venta-prod, categoría-prod, stock-prod, precio-compra-prod.

F = {

id-producto --> titulo-prod, descripción-prod, precio-venta-prod, categoría-prod, stock-prod, precio-compra-prod;

}

CC = id-producto

R7 = id-pedido, id-producto, cantidad-producto.

F = {

id-pedido, id-producto --> cantidad-producto;

}

CC = id-pedido, id-producto.

R8 = id-usuario, id-clase, cuota-al-día.

F = {

id-usuario, id-clase --> cuota-al-día;

}

CC = id-usuario, id-clase

R9 = id-pago-cuota, id-usuario, id-clase, fecha-pago, importe-pagado, id-pago-cuota, importe-pagado.

F = {

id-pago-cuota --> id-usuario, id-clase, fecha-pago, importe-pagado;

id-usuario, id-clase, fecha-pago --> id-pago-cuota, importe-pagado;

}

CC1 = id-pago-cuota

CC2 = id-usuario, id-clase, fecha-pago

R10 = id-evento → nombre-evento, dia-evento, horario-evento, descripción, precio, id-profesor/a;

F = {

id-evento → nombre-evento, dia-evento, horario-evento, descripción, precio, id-profesor/a;

}

CC = id-evento

R ~= R' |x| R2 |x| R3 |x| R4 |x| R5 |x| R6 |x| R7 |x| R8 |x| R9 |x| R10 |x| R1

Referencias

Lista de algunas de las fuentes bibliográficas y electrónicas consultadas.

- Angular Documentation: https://docs.angular.lat/docs
- Node.js Documentation: https://nodejs.org
- MySQL Documentation: https://dev.mysql.com/doc
- Mercado Pago API: https://mercadopago.com/developers
- Vercel Documentation: https://vercel.com/docs
- Railway Documentation: https://docs.railway.app/
- Stack Overflow: https://stackoverflow.com/
- MDN Web Docs: https://developer.mozilla.org/es/docs/Learn
- Tailwind: https://www.creative-tim.com/twcomponents/components
- Integración de Mercado Pago: https://www.youtube.com/watch?v=vEXwN9-tKcs
- W3School: https://www.w3schools.com/mysql/mysql_datatypes.asp
- MySQL YA: https://mysqlya.com.ar/programacion/
- Free Code Camp: https://www.freecodecamp.org/espanol/news/