DESARROLLO DE APLICACIONES WEB EN ENTORNO CLIENTE

DOCUMENTACIÓN DE LA APLICACIÓN MINDSTONE PILATES CENTER

Resumen de la Aplicación Web:

La aplicación web "MindStone Pilates Center" ha sido creada para gestionar de manera integral un centro de pilates, permitiendo a los usuarios registrarse, adquirir bonos, reservar clases y gestionar su perfil, así como a los administradores controlar coaches, usuarios, clases y reservas.

ANIMACIONES Y FUNCIONES FRONTEND

Sección 1: Inicio

Archivo que agrupa la inicialización de animaciones en la sección principal (hero), el desplazamiento suave entre secciones, y la validación del formulario de contacto utilizando funciones importadas de validación.

Funciones y bloques principales:

- Animaciones de la sección Hero: Se animan el título principal y el botón de acción para aparecer suavemente tras cargar la página.
- Animación de desplazamiento suave: Al hacer clic en el trigger del hero, la página hace un scroll animado hacia la siguiente sección.
- Validación y envío del formulario de contacto:
    - Validación de campos: Al enviar el formulario, se validan los campos de nombre, email, teléfono y mensaje utilizando funciones importadas. Si algún campo es inválido, se muestra su respectivo mensaje de error y se previene el envío.
    - Validación en tiempo real: A medida que el usuario corrige los campos, los mensajes de error se ocultan automáticamente si el dato ingresado es válido.
    - Envío de datos: Si todos los campos son válidos, los datos se envían asíncronamente mediante AJAX (fetch) al controlador PHP, que procesa y envía el mensaje al correo de destino usando PHPMailer.
    - Mensaje de éxito: Tras el envío exitoso, se muestra un mensaje de agradecimiento y el formulario se limpia. El mensaje de éxito se oculta automáticamente después de unos segundos.

Sección 2: FAQ acordeón y scroll suave en página Classes

Este script implementa dos funcionalidades principales en la interfaz web:

- Un acordeón de Preguntas Frecuentes (FAQ) interactivo utilizando jQuery, que permite expandir y contraer respuestas al hacer clic en las preguntas.
- Un desplazamiento suave (scroll suave) a una sección específica de la página al hacer clic en un icono utilizando JavaScript puro.

Sección 3: Slideshow en página Studio

Inicializa el plugin Slick Slider en los elementos con la clase 'lazy' cuando el DOM está listo.

- Configura la galería para mostrar 3 imágenes a la vez y mover una por cada scroll en pantallas grandes.
- Muestra flechas personalizadas para navegar entre las imágenes.
- Ajusta el número de imágenes visibles y la configuración de autoplay/arrows en función del tamaño de pantalla:
    - En pantallas menores a 768px: muestra 2 imágenes.
    - En pantallas menores a 480px: muestra 1 imagen y activa el autoplay.

SECCIÓN DASHBOARD ADMINISTRADOR

Sección 1: Funcionalidades dashboard

Este archivo contiene el código para mostrar información del número de reservas, clases más y menos populares, horas pico, usuarios registrados y visitas al sitio web en el dashboard. Este código se ejecuta al cargar la página y obtiene los datos del backend a través de una solicitud fetch.

Se usa la API de google analytics para obtener el número de visitas al sitio web.

Card de Bookings: Busca la tarjeta de "Bookings" en el dashboard y actualiza: El total de reservas de este mes y el crecimiento porcentual respecto al mes anterior.

Card de clase más popular: Busca la tarjeta de "Most popular class" y muestra el nombre de la clase más reservada del mes.

Card de clase menos popular: Busca la tarjeta de "Least popular class" y muestra el nombre de la clase más reservada del mes.

Card de peak hours: Busca la tarjeta de "Peak hours" y muestra la hora con más reservas del mes y el crecimiento porcentual respecto al mes anterior.

Card de users: Busca la tarjeta de "Users" y muestra lo usuarios registrados este mes y el crecimiento respecto al mes anterior (usuarios y %).

Card de views: Busca la tarjeta de "Views to the website" y muestra la cantidad de visitas este mes y el crecimiento porcentual respecto al mes anterior.

Sección 2: Funcionalidades para gestión de clases:

El código está diseñado para mostrar un calendario semanal de clases de pilates, permitiendo alta, edición, borrado y filtrado de clases. La interacción con el backend se realiza principalmente mediante fetch a endpoints PHP. Se utiliza manipulación directa del DOM para renderizar los diferentes componentes de la interfaz. El control de horarios ocupados y seleccionados previene solapamientos y errores en la creación de clases.

normalizeHour(hourStr): Normaliza un string de hora al formato hh:mm. Recorta a los primeros 5 caracteres si la longitud es suficiente. Sirve para mostrar horas uniformemente en la interfaz.

getClassBg(typeName): Determina la clase de color de fondo y borde según el tipo de clase de pilates. Compara el nombre recibido con los del mapa y devuelve la clase CSS correspondiente para estilizar los bloques de clase.

renderCalendarClassList(classes): Agrupa las clases recibidas por día de la semana y renderiza una grilla semanal donde muestra cada clase con su información, incluyendo botones de edición y borrado. Si un día no tiene clases, muestra un mensaje de "No classes".

loadPilatesTypes(): Llama al backend para obtener el listado de tipos de pilates (especialidades) y los almacena en una variable global para ser usados en selects y filtros.

loadCoachesBySpeciality(specialityId, selectElement, selectedValue): Llama al backend para obtener la lista de coaches que tienen la especialidad indicada. Llena el select de coaches y preselecciona uno si corresponde.

fillSelectOptions(select, options, selectedValue, keyId, keyName): Llena un elemento select HTML con opciones de un array, usando la llave de id y nombre. Marca como seleccionada la opción indicada en selectedValue.

renderWeeklyAgenda(): Renderiza un grid con los días de la semana, permitiendo agregar y remover horarios para cada día. Muestra los horarios ocupados y los seleccionados en cada caso.

loadSpecialitiesForCreate(): Carga los tipos de pilates desde el backend y llena el select del formulario de alta de clases.

loadOccupiedHours(): Consulta al backend por los horarios ya ocupados (no disponibles) para la combinación de tipo y coach seleccionados, y actualiza la agenda para evitar solapamientos.

addHour(day, hour): Agrega un horario a la lista de horarios seleccionados para el día indicado y vuelve a renderizar la agenda.

removeHour(day, hour): Elimina un horario de la lista de horarios seleccionados para el día indicado y vuelve a renderizar la agenda.

renderClassFilters(types): Renderiza los botones de filtro para tipo de clase y día de la semana. Permite al usuario filtrar las clases mostradas en el calendario.

applyClassFilters(): Aplica los filtros seleccionados (tipo y día) sobre la lista de todas las clases y re-renderiza el calendario con el resultado.

openEditModal(cls): Abre el modal de edición de clase, llenándolo con los datos de la clase seleccionada, y permite modificar sus datos y guardarlos.

deleteClass(id): Pregunta al usuario si está seguro de borrar una clase, luego llama al backend para eliminarla. Si tiene éxito, actualiza el calendario de clases.

fetchAndRenderClassList(): Hace fetch al backend de la lista de clases y tipos, luego llama a las funciones para renderizar los filtros y el calendario de clases. Es la función central para cargar o recargar la vista principal.

Sección 3: Funcionalidades para gestión de coaches:

El código está diseñado para gestionar coaches (entrenadores), permitiendo alta, edición, eliminación y visualización de la lista, así como la asignación de especialidades a cada coach. Se emplea validación de campos antes del envío de formularios usando funciones importadas para asegurar la integridad de los datos. El flujo contempla mensajes claros de éxito y error, y la recarga automática de la lista de coaches tras cualquier operación relevante.

isValidName, isValidEmail, isValidInternationalPhone: Funciones importadas para validar nombres, emails y teléfonos internacionales. Se usan antes de enviar datos al backend para evitar errores y asegurar la calidad de los datos ingresados.

Mostrar y ocultar el formulario de alta: Al presionar el botón de agregar coach, el formulario se muestra u oculta alternadamente. Se limpia el mensaje previo y se actualiza el texto del botón para indicar el estado.

Envío del formulario para agregar coach: Recolecta y valida los datos ingresados. Si algún dato no es válido, muestra un mensaje de error. Si todo es correcto, envía los datos por AJAX (fetch) al backend para registrar el coach. Si la respuesta es exitosa, resetea y oculta el formulario y recarga la lista.

showMessage(msg, color, btn): Muestra un mensaje en el formulario de alta, con el color indicado (rojo para error, verde para éxito). Puede reactivar el botón de submit si se pasó como argumento.

loadCoachesList(): Consulta al backend la lista de coaches y la muestra en una tabla. Si no hay datos o ocurre un error, muestra el mensaje correspondiente.

buildCoachesTable(coaches): Genera el HTML de la tabla de coaches, mostrando nombre, email, teléfono, especialidades y botones de acción para editar o eliminar.

renderTags(specialities): Renderiza las especialidades de cada coach como etiquetas de colores para facilitar la visualización e identificación rápida de sus habilidades.

getSpecialityColorClasses(name): Retorna las clases CSS de color para aplicar a las etiquetas y checkboxes, diferenciando visualmente cada especialidad.

renderSpecialities(allSpecialities, coachSpecialitiesIds): En el modal de edición, muestra todas las especialidades disponibles como checkboxes y marca las que ya tiene el coach.

openModal(modal) y closeModal(modal): Agregan o remueven las clases CSS necesarias para mostrar u ocultar el modal de edición en la interfaz.

Edición y eliminación de coaches: Al hacer clic en el botón de editar, se solicita al backend los datos del coach y se abre el modal con el formulario precargado. Al guardar, se valida y se envía la petición de actualización. Al eliminar, pide confirmación y si se acepta, borra el coach y recarga la lista.

Envío del formulario de edición: Recoge y valida los datos del formulario de edición. Si todo es correcto, envía la actualización al backend, muestra un mensaje y recarga la lista tras el éxito.

Carga inicial de coaches: Al cargar la página, se solicita automáticamente la lista de coaches para que el usuario vea siempre la información más actualizada en el dashboard.

Sección 3: Funcionalidades para gestión de usuarios:

El código permite crear, editar y eliminar usuarios desde el dashboard administrativo, asegurando validación de datos, recarga automática de la información y mensajes claros para el usuario. La manipulación del DOM permite mostrar u ocultar formularios, actualizar datos en tiempo real y gestionar la experiencia de usuario con mensajes y confirmaciones.

jsonPost(url,data): Función utilitaria que envía peticiones POST en formato JSON a la URL indicada y retorna la respuesta como objeto JS parseado.

initCreateUserForm(formId,msgId): Inicializa y gestiona el formulario de alta de usuarios. Valida los datos ingresados en el formulario, mostrando mensajes de error si hay campos vacíos o inválidos. Si todo es correcto, envía los datos al backend para registrar un nuevo usuario. En caso de éxito, muestra el mensaje y recarga la página para mostrar los cambios.

Evento de edición de usuario: Escucha el envío del formulario de edición de usuario, recolecta y valida datos, y envía la actualización al backend. Muestra mensajes según el resultado y recarga la página si la edición fue exitosa.

Evento de eliminación de usuario: Escucha clics en los botones de eliminar usuario, solicita confirmación al usuario y, si se acepta, envía la petición de borrado al backend. Muestra alertas según el resultado y recarga la página si la eliminación fue exitosa.

toggleAddUser: Permite mostrar u ocultar el formulario de alta de usuario al hacer clic en el botón correspondiente. Si el formulario debe mostrarse, lo carga dinámicamente desde el backend y reinicializa las validaciones.

En resumen, el archivo implementa todas las operaciones CRUD de usuarios en el dashboard, asegurando validaciones previas, interacción fluida con el backend y una experiencia de usuario consistente y reactiva.

SECCIÓN DASHBOARD USUARIO

Sección 1: Funcionalidades para mostrar el calendario de clases y petición de reservas.

Archivo que carga el calendario de clases de pilates y permite reservar clases desde la página de clases si el usuario está logueado y si no, muestra un modal de login.

Este archivo se carga en la página de clases y en la página de timetable del usuario logueado.

typeColorMap: Es un objeto que asocia cada tipo de clase con una combinación de colores (clases CSS) para estilizar visualmente cada tipo de clase en el calendario.

loadCalendar(): Función que obtiene las clases desde el backend vía AJAX las organiza por día y hora para estructurar una tabla semanal. Genera dinámicamente una tabla HTML con los días de la semana como columnas y las horas como filas.

Permite reservar una clase haciendo click sobre ella. Si el usuario no está logueado, muestra el modal de login. Si está logueado, pide confirmación mediante un toast personalizado y luego realiza la reserva por AJAX.

Usa AOS (Animate On Scroll) para animar las tarjetas del calendario.

Sección 2: Funcionalidades para mostrar los créditos activos del usuario y las reservas hechas por el usuario

loadUserCredits(): Función asíncrona que consulta al backend los créditos del usuario. Si ocurre un error, muestra un mensaje de error. Si la consulta es exitosa, filtra los créditos para mostrar solo los que tienen créditos disponibles y no han expirado. Si no hay créditos activos, muestra un mensaje informativo. Si hay créditos activos, muestra una tarjeta por cada uno, con el total de créditos, los usados y la fecha de expiración, usando un diseño visual agradable y responsivo.

loadReservations(page): Función asíncrona que consulta al backend la lista de reservas del usuario para la página indicada. Si hay reservas, las muestra en filas con sus datos: tipo de clase, coach, fecha, hora, fecha de reserva, estado y botón para cancelar (si aplica).

updatePagination(totalItems, currentPage): Calcula el total de páginas y renderiza los botones de “anterior” y “siguiente” para la navegación, así como la información de la página actual. Los botones se habilitan o deshabilitan automáticamente según la página.

document.addEventListener('click', ...): Escucha clics en el documento y, si el objetivo es el botón de cancelar reserva, muestra un toast de confirmación. Si el usuario confirma, realiza la petición para cancelar la reserva al backend. Tras cancelar, recarga la tabla de reservas y los créditos del usuario, mostrando el mensaje correspondiente.

loadUserCredits(): Función importada que actualiza el resumen de créditos disponibles del usuario. Se llama después de cancelar una reserva para reflejar los cambios en tiempo real.

loadReservations(currentPage): Se ejecuta inicialmente al cargar la página para mostrar la primera página de reservas del usuario.

VALIDACIÓN DE FORMULARIO

Este módulo proporciona funciones para validar distintos tipos de datos de formularios, asegurando que los valores ingresados por el usuario cumplan con los requisitos esperados antes de ser enviados o procesados. Cada función retorna true si el valor cumple la condición y false en caso contrario, permitiendo un control sencillo y robusto en el frontend.

isValidName(name): Verifica si el nombre contiene solo letras (incluyendo acentos), espacios, apóstrofes y guiones. Previene el ingreso de números o caracteres no permitidos en nombres y apellidos.

isValidEmail(email): Valida que el email tenga el formato estándar, es decir, que incluya texto antes y después del símbolo "@" y un dominio con punto.

isValidInternationalPhone(phone): Comprueba que el teléfono comience con el símbolo "+" seguido de entre 6 y 15 dígitos, asegurando que el número sea internacional y tenga la longitud apropiada.

isValidPassword(password): Valida que la contraseña tenga al menos 8 caracteres, contenga al menos una letra mayúscula, un número y un carácter especial, cumpliendo los estándares típicos de seguridad para contraseñas.

LOGIN Y REINICIO DE CONTRASEÑA

Este script controla los modales de login y olvido de contraseña de un sistema. Permite abrir/cerrar los modales, valida los formularios y envía los datos al servidor mediante AJAX. También muestra mensajes de éxito o error según la respuesta del backend o, si no hay respuesta, según las validaciones de frontend.

Apertura y cierre de modales: Permite mostrar y ocultar los modales de login y recuperación de contraseña según las acciones del usuario (click en botones correspondientes).

Validación y envío de formulario de login: Al enviar el formulario de login, se valida que el email y la contraseña estén completos y en el formato correcto. Si la validación es correcta, se envían los datos al backend por AJAX. Se muestra el mensaje de éxito o error que devuelve el backend, y si el login es exitoso, se redirige al usuario según su rol.

Validación y envío de formulario de recuperación: Al enviar el formulario de recuperación, se valida el email. Si es válido, se envía al backend por AJAX. Se muestra el mensaje que retorna el backend o, si hay error, un mensaje genérico.

Prevención de múltiples envíos: Se usa una bandera interna para evitar que los eventos de submit se registren más de una vez por formulario.

Gestión de mensajes: Los mensajes de error y éxito se muestran en la interfaz, priorizando los que devuelve el backend.