Índice de Contenido

[**Base.html** 4](#_Toc184649714)

[1. Estructura 4](#_Toc184649715)

[**Página Alumno.html** 7](#_Toc184649716)

[1. Estructura Base de Django 7](#_Toc184649717)

[2. Bloques Personalizados 8](#_Toc184649718)

[3. Carrusel de Actividades 8](#_Toc184649719)

[4. Botones de Navegación del Carrusel 9](#_Toc184649720)

[**Página historial\_creditos.html** 9](#_Toc184649721)

[1. Indicador de Puntos Totales 9](#_Toc184649722)

[2. Filtro por Cuatrimestres 9](#_Toc184649723)

[3. Contenedor de Semanas 10](#_Toc184649724)

[4. Interactividad Potencial (JavaScript) 12](#_Toc184649725)

[**Script app\_historial.js (JS historial\_creditos.html)** 12](#_Toc184649726)

[1. Variables Globales 12](#_Toc184649727)

[2. Mostrar/Ocultar el Menú Desplegable 12](#_Toc184649728)

[3. Filtrar por Cuatrimestre 13](#_Toc184649729)

[4. Calcular Puntos Obtenidos 14](#_Toc184649730)

[5. Cerrar el Menú Desplegable 14](#_Toc184649731)

[6. Inicialización al Cargar 15](#_Toc184649732)

[**Página créditos\_actuales.html** 15](#_Toc184649733)

[1. Bloque de Contenido Principal 15](#_Toc184649734)

[2. Contenedor del Calendario y el Reloj 15](#_Toc184649735)

[3. Modal de Asistencia 16](#_Toc184649736)

[4. Cierre del Bloque de Contenido 17](#_Toc184649737)

[¿Qué Hace Este Código? 17](#_Toc184649738)

[¿Qué Scripts Soporta Este Código? 18](#_Toc184649739)

[**Script app-creditos.js (JS creditos\_actuales.html)** 18](#_Toc184649740)

[1. Variables Globales 18](#_Toc184649741)

[2. Función: updateClock 19](#_Toc184649742)

[3. Función: generateCalendar 19](#_Toc184649743)

[4. Función: handleDayClick 20](#_Toc184649744)

[5. Función: showModal 20](#_Toc184649745)

[6. Función: closeModal 20](#_Toc184649746)

[7. Evento: DOMContentLoaded 21](#_Toc184649747)

[¿Cómo Funciona Todo Junto? 21](#_Toc184649748)

[**Actividades de Club: ajedrez.html** 22](#_Toc184649749)

[1. Fondo con Título del Club 22](#_Toc184649750)

[2. Contenedor Principal 22](#_Toc184649751)

[3. Sección de Logros 23](#_Toc184649752)

[4. Otras Categorías (Logros y Competencias) 24](#_Toc184649753)

[5. Pendiente y Observaciones 24](#_Toc184649754)

[**Página actividades\_culturales.html** 25](#_Toc184649755)

[1. Formulario 25](#_Toc184649756)

[2. Vista actividades culturales 25](#_Toc184649757)

[3. Modelo Solicitud 27](#_Toc184649758)

[**Página Login.html** 28](#_Toc184649759)

[1. Formulario Login 28](#_Toc184649760)

[2. Vista Login 29](#_Toc184649761)

[3. Vista Logout 30](#_Toc184649762)

[4. Forms.py 31](#_Toc184649763)

[**Página Creditos.html (Administrador)** 31](#_Toc184649764)

[1. Estructura html 31](#_Toc184649765)

[2. Vista de créditos 33](#_Toc184649766)

[3. Objetivo 33](#_Toc184649767)

[4. Pendiente y Observaciones 34](#_Toc184649768)

[**Página Carta\_laboral.html (Administrador)** 34](#_Toc184649769)

[1. Estructura html 34](#_Toc184649770)

[2. Vista carta laboral 35](#_Toc184649771)

[3. Objetivo 35](#_Toc184649772)

[**Página Solicitudes.html (Profesor)** 36](#_Toc184649773)

[1. Estructura html de solicitudes 36](#_Toc184649774)

[2. Vista de solicitudes 37](#_Toc184649775)

[3. Vistas para cambiar el estado 38](#_Toc184649776)

[4. Objetivo 40](#_Toc184649777)

[5. Pendiente y Observaciones 40](#_Toc184649778)

[**Profesor.html (Profesor)** 40](#_Toc184649779)

[Estructura html de solicitudes 40](#_Toc184649780)

[Vista 42](#_Toc184649781)

[Vista/Función para extraer el excel 43](#_Toc184649782)

[Objetivo 45](#_Toc184649783)

[Pendientes y Observaciones 45](#_Toc184649784)

[Archivo Script (script\_profesor.js) 45](#_Toc184649785)

# **Base.html**

### **Estructura**

{% load static %}

<!DOCTYPE html>

<html *lang*="en">

  <head>

    <title>{% block titulo %} {% endblock %}</title>

    <meta *charset*="utf-8" />

    <meta *name*="viewport" *content*="width=device-width, initial-scale=1, shrink-to-fit=no" />

    <link *rel*="stylesheet" *href*="{% static 'css/style.css'%}" />

    <link *rel*="stylesheet" *href*="{% static 'css/diseño-inicio.css'%}" />

    <link *rel*="stylesheet" *href*="{% static 'css/diseño-club.css'%}" />

    <link *rel*="stylesheet" *href*="{% static 'css/diseño-creditos.css'%}" />

    {% block styles %}{% endblock styles %}

    <link *href*="https://fonts.googleapis.com/icon?family=Material+Icons" *rel*="stylesheet" />

    <link *rel*="stylesheet" *href*="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/6.4.0/css/all.min.css" *integrity*="sha512-iecdLmaskl7CVkqkXNQ/ZH/XLlvWZOJyj7Yy7tcenmpD1ypASozpmT/E0iPtmFIB46ZmdtAc9eNBvH0H/ZpiBw==" *crossorigin*="anonymous" *referrerpolicy*="no-referrer"/>

  </head>

  <body>

    <div *class*="menu">

      <ion*-icon* *name*="menu-outline"></ion*-icon*>

      <ion*-icon* *name*="close-outline"></ion*-icon*>

    </div>

    <div *class*="dividir" *style*="display: flex; height: 100vh">

      <aside>

        <div *class*="barra-lateral">

          <div>

            <div *class*="nombre-pagina">

              <img *id*="cloud" *src*="{% static 'assets/Logo-UTS-2023.png' %}" *alt*=""/>

              {% if '/alumno/' in request.path %}

                <span>Mis créditos</span>

              {% elif '/profesor/' in request.path %}

                <span>Docente</span>

              {% endif %}

            </div>

          </div>

          <nav *class*="navegacion">

            {% if '/alumno/' in request.path %}

              <ul>

                <li>

                  <a *href*="{% url 'alumno:inicio' %}">

                    <ion*-icon* *name*="home-outline"></ion*-icon*>

                    <span>Inicio</span>

                  </a>

                </li>

                <li>

                  <a *href*="{% url 'alumno:creditos\_actuales'%}">

                    <ion*-icon* *name*="invert-mode-outline"></ion*-icon*>

                    <span>Créditos actuales</span>

                  </a>

                </li>

                <li>

                  <a *href*="{% url 'alumno:historial\_creditos'%}">

                    <ion*-icon* *name*="clipboard-outline"></ion*-icon*>

                    <span>Historial de créditos</span>

                  </a>

                </li>

                <li>

                  <a *href*="{% url 'alumno:actividades\_culturales'%}">

                    <ion*-icon* *name*="medal-outline"></ion*-icon*>

                    <span>Actividades</span>

                  </a>

                </li>

              </ul>

            {% elif '/profesor/' in request.path%}

              <ul *class*="ps-0">

                <li>

                  <a *href*="{% url 'profesor:inicio' %}">

                    <ion*-icon* *name*="home-outline"></ion*-icon*>

                    <span>Inicio</span>

                  </a>

                </li>

                <li>

                  <a *href*="{% url 'profesor:solicitudes'%}">

                    <ion*-icon* *name*="invert-mode-outline"></ion*-icon*>

                    <span>Ver solicitudes</span>

                  </a>

                </li>

              </ul>

            {% elif '/administrador' in request.path%}

            {% comment %} COPIAR EL UL Y MODIFICAR A TUS URLS {% endcomment %}

            {% endif %}

          </nav>

          <div>

            <!-- <div class="linea"></div> -->

            <div *class*="modo-oscuro">

              <div *class*="info">

                <!-- <ion-icon name="moon-outline"></ion-icon>

                <span>Dark Mode</span> -->

              </div>

              <div *class*="switch">

                <!-- <div class="base">

                  <div class="circulo"></div>

                </div> -->

              </div>

            </div>

            <div *class*="usuario">

              <img *src*="{% static 'assets/profile.jpg' %}" *alt*="" />

              <div *class*="info-usuario">

                <div *class*="nombre-email">

                  <span *class*="nombre">nombre\_apellido</span>

                  <span *class*="email">{{ login }}</span>

                </div>

                <a *href*="{% url 'login:logout' %}">

                  <ion*-icon* *name*="log-out-outline"></ion*-icon*>

                </a>

              </div>

            </div>

          </div>

        </div>

      </aside>

      <div

*class*="contenido"

*style*="width: 100vw; height: 100vh; overflow: scroll"

      >

        {% block contenido %} {% endblock %}

      </div>

    </div>

    <script

*type*="module"

*src*="https://unpkg.com/ionicons@7.1.0/dist/ionicons/ionicons.esm.js"

    ></script>

    <script

*nomodule*

*src*="https://unpkg.com/ionicons@7.1.0/dist/ionicons/ionicons.js"

    ></script>

    <!-- ARCHIVO SCRIPT -->

    <script *src*="{% static 'js/script.js' %}"></script>

    <script *src*="{% static 'js/app-inicio.js' %}"></script>

    <script *src*="{% static 'js/app-club.js' %}"></script>

    <script *src*="{% static 'js/app-creditos.js' %}"></script>

    {% block scripts %}{% endblock scripts %}

  </body>

</html>

**Propósito:** Ser la estructura base el cuál contiene la barra lateral (Aside) diseño, componente y según cada vista (Alumno, Administrador, Docente) muestra sus respectivas navegaciones. También agrega toda la estructura básica HTML, así como los estilos, scripts

# **Página Alumno.html**

### **Estructura Base de Django**

html

Copiar código

{% extends "pages/base.html" %}

* **Qué hace:** Indica que este archivo HTML extiende (hereda) la estructura definida en pages/base.html. Esto permite reutilizar elementos comunes como el encabezado y el pie de página.
* Centraliza el diseño, facilitando actualizaciones globales.

html

Copiar código

{% block titulo %}Inicio{% endblock %}

* Define el título dinámico de la página como "Inicio". Este bloque se sustituirá en la plantilla base donde corresponda.

### **Bloques Personalizados**

html

Copiar código

{% block inicio\_activo %} active a-extend {% endblock %}

* Marca el apartado de "Inicio" como activo (por ejemplo, resaltándolo en un menú de navegación). Esto se puede combinar con CSS para estilizar elementos activos.

### **Carrusel de Actividades**

html

Copiar código

<div class="slide">

<div class="item" style="background-image: url(../static/assets/voleybol.jpeg)">

<div class="fade-content">

<div class="content">

<div class="name">Voleibol</div>

<div class="des">Mejorar las habilidades técnicas y tácticas...</div>

<a href="{% url 'alumno:voleibol' %}">

<button type="button" class="btn btn-outline-light">Ver más</button>

</a>

</div>

</div>

</div>

<!-- Más items aquí -->

</div>

* **Qué hace:**
  1. **Div .slide:** Contenedor principal del carrusel.
  2. **Div .item:** Cada elemento del carrusel.
     + **style="background-image:** Define la imagen de fondo del elemento.
     + **Subelementos:**
       - .name: Título de la actividad (e.g., "Voleibol").
       - .des: Descripción de la actividad.
       - Botón: Un enlace dinámico generado con la etiqueta de Django {% url 'nombre' %} que redirige a la página correspondiente.
* **Interactividad:** El carrusel muestra múltiples actividades con información y enlaces personalizados.

### **Botones de Navegación del Carrusel**

html

Copiar código

<div class="button">

<button class="btn btn-outline-light prev">

<i class="fa-solid fa-arrow-left"></i>

</button>

<button class="btn btn-outline-light next">

<i class="fa-solid fa-arrow-right"></i>

</button>

</div>

* **Qué hace:**
  + **Botón .prev:** Navega hacia el elemento anterior del carrusel.
  + **Botón .next:** Navega hacia el siguiente elemento del carrusel.
* Utiliza la librería **Font Awesome** para incluir flechas estilizadas.

# **Página historial\_creditos.html**

### **Indicador de Puntos Totales**

html

Copiar código

<div class="indicador-puntos">

<span id="indicador-puntos">0/105</span>

</div>

* **Qué hace:**
  + Muestra el puntaje total actual del usuario (inicialmente 0) sobre un total posible de 105 puntos.
* **Interactividad:** Este valor podría actualizarse dinámicamente usando JavaScript según la lógica de cálculo.

### **Filtro por Cuatrimestres**

html

Copiar código

<div class="filtro">

<div class="dropdown">

<button class="dropdown-btn" onclick="toggleDropdown()">Seleccionar Cuatrimestre</button>

<div id="dropdown-content" class="dropdown-content">

<script>

for (let i = 1; i <= 6; i++) {

document.write(`

<label>

<input type="radio" name="cuatrimestre" value="${i}" onchange="filtrarCuatrimestre()"> Cuatrimestre ${i}

</label>

`);

}

</script>

</div>

</div>

</div>

* **Qué hace:**
  + **Dropdown:** Permite al usuario seleccionar un cuatrimestre del 1 al 6.
  + **document.write:** Genera dinámicamente las opciones de filtro para cada cuatrimestre.
  + **Eventos:**
    - onclick="toggleDropdown()": Controla la apertura/cierre del menú desplegable.
    - onchange="filtrarCuatrimestre()": Filtra las semanas visibles según el cuatrimestre seleccionado.

### **Contenedor de Semanas**

html

Copiar código

<div id="tabla-semanas" class="tabla-flex">

<script>

for (let cuatrimestre = 1; cuatrimestre <= 6; cuatrimestre++) {

const maxPuntosPorSemana = cuatrimestre <= 2 ? 2 : 1;

for (let semana = 1; semana <= 15; semana++) {

const puntosSemana = Math.floor(Math.random() \* maxPuntosPorSemana) + 1;

document.write(`

<div class="semana cuatrimestre-${cuatrimestre}" data-cuatrimestre="${cuatrimestre}" data-puntos="${puntosSemana}" style="display: none;">

<div class="header">

<span>Semana ${semana}</span>

<span>Puntos: ${puntosSemana}</span>

</div>

<div class="contenido">

<div class="dias">

<div>L</div><div>M</div><div>M</div><div>J</div><div>V</div>

</div>

<div class="asistencias">

<div>${Math.floor(Math.random() \* 2)}</div>

<div>${Math.floor(Math.random() \* 2)}</div>

<div>${Math.floor(Math.random() \* 2)}</div>

<div>${Math.floor(Math.random() \* 2)}</div>

<div>${Math.floor(Math.random() \* 2)}</div>

</div>

</div>

</div>

`);

}

}

</script>

</div>

* **Qué hace:**
  1. **Generación Dinámica:**
     + Crea un total de 90 bloques (6 cuatrimestres x 15 semanas).
     + Cada bloque tiene:
       - **Puntos:** Se calculan aleatoriamente, basándose en un máximo determinado por el cuatrimestre (2 puntos/semana para los primeros 2 cuatrimestres y 1 punto/semana para los siguientes).
       - **Asistencias:** Muestra valores aleatorios (0 o 1) para cada día de la semana laboral (Lunes a Viernes).
  2. **Clases y Atributos:**
     + **Clases:** Cada semana recibe una clase que indica el cuatrimestre (cuatrimestre-X).
     + **Atributos:**
       - data-cuatrimestre: Indica el cuatrimestre al que pertenece.
       - data-puntos: Contiene los puntos asignados a la semana.
  3. **Estilo Inicial:** Todos los bloques están ocultos (style="display: none;") hasta que se filtre el cuatrimestre correspondiente.

### **Interactividad Potencial (JavaScript)**

* **toggleDropdown():** Función para mostrar/ocultar el menú desplegable del filtro.
* **filtrarCuatrimestre():** Función que:
  + Identifica qué cuatrimestre fue seleccionado.
  + Muestra solo las semanas correspondientes a ese cuatrimestre.
* **Actualización de Puntos:**
  + Es probable que exista un script adicional que sume los puntos visibles y actualice el indicador total.

## **Script app\_historial.js (JS historial\_creditos.html)**

### **1. Variables Globales**

javascript

Copiar código

let totalPuntos = 105;

let puntosObtenidos = 0;

* **totalPuntos:** Define la cantidad máxima de puntos posibles (105).
* **puntosObtenidos:** Inicializa los puntos acumulados por el usuario en 0. Este valor se actualizará dinámicamente.

### **2. Mostrar/Ocultar el Menú Desplegable**

javascript

Copiar código

function toggleDropdown() {

const dropdown = document.getElementById("dropdown-content");

dropdown.style.display = dropdown.style.display === "block" ? "none" : "block";

}

* **Qué hace:**
  + Alterna la visibilidad del menú desplegable (block para mostrarlo, none para ocultarlo).
  + Utiliza el atributo style.display para cambiar entre estados.
* **Cuándo se llama:** Se ejecuta al hacer clic en el botón "Seleccionar Cuatrimestre".

### **3. Filtrar por Cuatrimestre**

javascript

Copiar código

function filtrarCuatrimestre() {

const cuatrimestres = document.querySelectorAll(".semana");

const seleccionCuatrimestre = document.querySelector('input[name="cuatrimestre"]:checked').value;

// Ocultar todas las semanas inicialmente

cuatrimestres.forEach(semana => {

semana.style.display = "none";

});

// Mostrar semanas del cuatrimestre seleccionado

const semanasSeleccionadas = document.querySelectorAll(`.cuatrimestre-${seleccionCuatrimestre}`);

semanasSeleccionadas.forEach(semana => {

semana.style.display = "flex";

});

calcularPuntos();

closeDropdown();

}

* **Qué hace:**
  1. **Obtiene el cuatrimestre seleccionado:**
     + Busca el input seleccionado en el grupo radio (name="cuatrimestre").
  2. **Oculta todas las semanas:**
     + Cambia el estilo de todas las semanas a display: none.
  3. **Muestra las semanas del cuatrimestre seleccionado:**
     + Cambia el estilo de las semanas con la clase correspondiente (.cuatrimestre-X) a display: flex.
  4. **Actualiza puntos:** Llama a la función calcularPuntos para recalcular los puntos visibles.
  5. **Cierra el menú desplegable:** Llama a closeDropdown.

### **4. Calcular Puntos Obtenidos**

javascript

Copiar código

function calcularPuntos() {

puntosObtenidos = 0;

const semanasVisibles = document.querySelectorAll(".semana[style\*='flex']");

semanasVisibles.forEach(semana => {

puntosObtenidos += parseInt(semana.getAttribute("data-puntos"), 10);

});

document.getElementById("indicador-puntos").textContent = `${puntosObtenidos}/${totalPuntos}`;

}

* **Qué hace:**
  1. **Inicializa los puntos en 0:** Restablece el contador de puntos acumulados.
  2. **Identifica semanas visibles:**
     + Busca las semanas con display: flex (las que pertenecen al cuatrimestre seleccionado).
  3. **Suma los puntos de cada semana visible:**
     + Obtiene el atributo data-puntos de cada semana visible y lo convierte a número entero.
  4. **Actualiza el indicador:** Cambia el contenido del elemento con id="indicador-puntos" para mostrar los puntos acumulados y el total.

### **5. Cerrar el Menú Desplegable**

javascript

Copiar código

function closeDropdown() {

const dropdown = document.getElementById("dropdown-content");

dropdown.style.display = "none";

}

* **Qué hace:**
  + Asegura que el menú desplegable se oculte al realizar una selección.

### **6. Inicialización al Cargar**

javascript

Copiar código

window.onload = calcularPuntos;

* **Qué hace:**
  + Al cargar la página, llama a la función calcularPuntos para mostrar el puntaje inicial. Esto asegura que el indicador de puntos se sincronice con el contenido visible desde el principio.

# **Página créditos\_actuales.html**

### **1. Bloque de Contenido Principal**

html

Copiar código

{% block contenido %}

<div class="body-calendario">

* **{% block contenido %}:** Indica el inicio de un bloque de contenido en un sistema de plantillas como Django. Este bloque se inserta en un diseño base (base.html) en la ubicación del marcador {% block contenido %}.
* **<div class="body-calendario">:**

### **2. Contenedor del Calendario y el Reloj**

html

Copiar código

<div class="container-calendario">

<div id="clock"></div>

<div id="calendar"></div>

</div>

* **<div class="container-calendario">:** Agrupa el reloj y el calendario.
* **<div id="clock"></div>:**
  + Contenedor para mostrar un reloj en tiempo real (el contenido se agregará dinámicamente desde el script).
* **<div id="calendar"></div>:**
  + Contenedor para mostrar un calendario (probablemente generado dinámicamente o por un plugin).

### **3. Modal de Asistencia**

html

Copiar código

<!-- Modal actualizado -->

<div id="attendanceModal" class="modal">

<div class="modal-content">

<span class="close">&times;</span>

<h2 id="modalTitle"></h2>

<p id="modalMessage"></p>

<button id="confirmAttendance">Confirmar</button>

</div>

</div>

* **¿Qué es un Modal?**  
  Es un cuadro de diálogo o ventana emergente superpuesta que requiere la interacción del usuario antes de continuar.

#### Estructura del Modal

1. **Contenedor del Modal:**

html

Copiar código

<div id="attendanceModal" class="modal">

* + id="attendanceModal": Identificador único para este modal.
  + class="modal": Clase CSS para estilizar y posicionar el modal.

1. **Contenido del Modal:**

html

Copiar código

<div class="modal-content">

* + Contiene el contenido principal del modal.

1. **Cerrar el Modal:**

html

Copiar código

<span class="close">&times;</span>

* + Un botón o enlace (<span>) para cerrar el modal, representado por el símbolo &times; (una "X").

1. **Título del Modal:**

html

Copiar código

<h2 id="modalTitle"></h2>

* + Se utiliza para mostrar un título dinámico en el modal.

1. **Mensaje del Modal:**

html

Copiar código

<p id="modalMessage"></p>

* + Un párrafo para mostrar mensajes dinámicos.

1. **Botón de Confirmación:**

html

Copiar código

<button id="confirmAttendance">Confirmar</button>

* + Un botón que el usuario puede presionar para confirmar la acción, como registrar la asistencia.

### **4. Cierre del Bloque de Contenido**

html

Copiar código

</div>

{% endblock %}

* **</div>:** Cierra el contenedor principal (body-calendario).
* **{% endblock %}:** Marca el final del bloque de contenido definido al principio.

### **¿Qué Hace Este Código?**

1. **Reloj (clock):**
   * Proporciona un lugar donde se renderiza un reloj en tiempo real.
2. **Calendario (calendar):**
   * Espacio para mostrar un calendario, posiblemente interactivo, que podría incluir eventos o fechas importantes.
3. **Modal de Asistencia (attendanceModal):**
   * Una ventana emergente para confirmar la asistencia. Esto podría activarse al interactuar con el calendario.

### **¿Qué Scripts Soporta Este Código?**

Este diseño HTML depende de un script para:

1. **Generar y actualizar el reloj (#clock).**
2. **Crear el calendario y gestionar eventos interactivos (#calendar).**
3. **Abrir y cerrar el modal, y manejar acciones como "Confirmar asistencia".**

## **Script app-creditos.js (JS creditos\_actuales.html)**

Este script es una implementación completa que administra un reloj en tiempo real, un calendario interactivo y un sistema de registro de asistencia con un modal para confirmaciones. Aquí está el desglose detallado de su funcionalidad:

### **1. Variables Globales**

javascript

Copiar código

const attendanceRecords = {}; // Objeto para guardar los registros de asistencia

* **attendanceRecords:**  
  Almacena los registros de asistencia en un formato clave-valor.
  + Clave: Fecha en formato YYYY-MM-DD.
  + Valor: Objeto con fecha y hora de asistencia.

### **2. Función:** updateClock

javascript

Copiar código

function updateClock() {

const clock = document.getElementById('clock');

const now = new Date();

const timeString = now.toLocaleTimeString();

clock.textContent = `Hora actual: ${timeString}`;

}

* Actualiza el contenido del elemento #clock con la hora actual.
* La hora se actualiza cada segundo mediante setInterval.

### **3. Función:** generateCalendar

javascript

Copiar código

function generateCalendar() {

// Código para generar el calendario

}

Genera un calendario para el mes actual:

1. **Nombres de días:**  
   Crea elementos para los días de la semana (Dom, Lun, etc.).
2. **Días vacíos:**  
   Agrega espacios vacíos para alinear los días del mes con su posición en la semana.
3. **Días del mes:**  
   Crea elementos para cada día del mes y agrega un evento click para interactuar con cada día.

#### Destacado:

* Resalta el día actual con la clase today.
* Cada día tiene un evento click para manejar el registro de asistencia.

### **4. Función:** handleDayClick

javascript

Copiar código

function handleDayClick(day, month, year, dayElement) {

// Código para manejar el click en un día

}

Se ejecuta cuando el usuario selecciona un día en el calendario:

1. **Revisa si ya se registró asistencia:**  
   Si el día ya tiene la clase attended, muestra el registro existente.
2. **Verifica si es fin de semana:**  
   Bloquea los registros en sábados y domingos.
3. **Abre el modal:**  
   Solicita confirmar el registro de asistencia para el día seleccionado.

### **5. Función:** showModal

javascript

Copiar código

function showModal(title, message, confirmCallback = null) {

// Código para mostrar el modal

}

1. **Título (title) y mensaje (message):**  
   Muestra la información relevante en el modal.
2. **Botón de confirmación (confirmCallback):**  
   Si se pasa una función, se muestra el botón **Confirmar** y se asigna el evento onclick.

### **6. Función:** closeModal

javascript

Copiar código

function closeModal() {

const modal = document.getElementById('attendanceModal');

modal.style.display = 'none';

}

Oculta el modal cuando el usuario cierra la ventana o hace clic fuera de ella.

### **7. Evento:** DOMContentLoaded

javascript

Copiar código

document.addEventListener('DOMContentLoaded', () => {

// Inicialización de funciones y eventos

});

Se ejecuta una vez que el DOM está completamente cargado:

1. **Inicializa el reloj y el calendario.**
2. **Configura eventos para cerrar el modal.**
3. **Actualiza el reloj cada segundo.**

### **¿Cómo Funciona Todo Junto?**

1. **Carga Inicial:**
   * Se genera el calendario y se inicia el reloj.
2. **Interacción con el Calendario:**
   * El usuario hace clic en un día.
   * Si el día es válido, se abre un modal para confirmar el registro.
   * Al confirmar, el día se marca como attended y se guarda el registro.
3. **Modal:**
   * Se usa para mostrar información o confirmar acciones.

# **Actividades de Club: ajedrez.html**

El siguiente contenedor o página, es básicamente la base para todas las páginas de los clubes, así que todos llevan la misma “plantilla” por así decirlo.

### **1. Fondo con Título del Club**

html

Copiar código

<div class="wallpaper" style="background-image: url(../../../static/assets/ajedrez.jpeg);">

<div class="fade-wall">

<div class="name-wall">

<div class="name-club">Ajedrez</div>

</div>

</div>

</div>

* **Propósito**:
  + Crear un encabezado visualmente atractivo para la página del Club de Ajedrez.
* **Estructura**:
  + wallpaper: Aplica una imagen de fondo (en este caso, ajedrez.jpeg).
  + fade-wall: Superpone un efecto de desvanecimiento sobre el fondo para destacar el texto.
  + name-club: Contiene el nombre del club mostrado con tipografía destacada.
* **Estilo**:
  + Usa CSS para establecer la imagen como fondo y agregar un efecto de degradado.

### **2. Contenedor Principal**

html

Copiar código

<div class="contenedor">

<div class="contenedor-texto">

El Club de Ajedrez Universitario "Estrategia en Movimiento" es un lugar cálido y estimulante para estudiantes de distintas carreras que comparten la pasión por el ajedrez...

</div>

</div>

* **Propósito**:
  + Proveer una breve descripción del club, incluyendo su misión y actividades.
* **Estructura**:
  + contenedor: Enmarca todo el contenido principal.
  + contenedor-texto: Contiene el texto descriptivo, explicando los valores y actividades del club.

### **3. Sección de Logros**

html

Copiar código

<div class="contenedor-logros">

<!-- Trofeos -->

<div class="box" id="trofeos" onclick="toggleBox('trofeos')">

<i class="material-icons">emoji\_events</i> <!-- Icono de trofeo -->

<span>Trofeos</span>

<div class="box-content">

<h2>Trofeos de Futbol</h2>

<p>Listado de competiciones de danza ganadas...</p>

</div>

</div>

</div>

* **Propósito**:
  + Mostrar información sobre los logros alcanzados por el club, como trofeos y competiciones.
* **Estructura**:
  + Cada sección (Trofeos, Logros, Competencias) es un contenedor con clases específicas:
    - box: Representa un bloque desplegable.
    - box-content: Contiene la descripción y listado de logros.
  + Usa un icono (emoji\_events) para indicar visualmente el contenido del bloque.
* **Interactividad**:
  + Usa la función toggleBox('trofeos') para mostrar u ocultar el contenido cuando el usuario hace clic en el bloque.

### **4. Otras Categorías (Logros y Competencias)**

Cada bloque sigue el mismo esquema que **Trofeos**, pero con información diferente:

1. **Logros**:
   * Muestra récords o hitos destacados alcanzados por el club.
2. **Competencias**:
   * Lista las competencias en las que ha participado el club.

### **5. Pendiente y Observaciones**

1. **Inconsistencias**:
   * El contenido de Trofeos, Logros, y Competencias hace referencia a otros clubes (fútbol, danza). Esto parece ser un error que requiere corrección.
2. **Diseño Flexible**:
   * El uso de bloques desplegables (box) permite agregar más categorías en el futuro sin alterar significativamente el diseño.
3. **Interactividad Pendiente**:
   * Requiere una implementación o revisión del script toggleBox para que las secciones sean dinámicas.

# **Página actividades\_culturales.html**

## **Formulario**

<div class="mb-3 row">

<div class="mb-3 col-md-6">

<label for="nombre" class="form-label">Nombre</label>

<input disabled type="text" id="nombre" class="form-control" placeholder="Nombre" value="{{nombre }}">

</div>

<div class="col-md-6">

<label for="apellidos" class="form-label">Apellidos</label>

<input disabled type="text" id="apellidos" class="form-control" placeholder="Apellidos" value="{{apellidos }}">

</div>

</div>

<div class="mb-3 row">

<div class="mb-3 col-md-6">

<label for="carrera" class="form-label">Carrera</label>

<input disabled type="text" id="carrera" class="form-control" placeholder="Carrera" value="{{nombre\_carrera}}">

</div>

<div class="mb-3 col-md-3">

<label for="matricula" class="form-label">Matrícula</label>

<input disabled type="text" id="matricula" class="form-control" value="{{matricula }}">

</div>

<div class="col-md-3">

<label for="matricula" class="form-label">Grado y Grupo</label>

<input disabled type="text" id="grado" class="form-control" value="{{nombre\_grupo }}">

</div>

</div>

* **Proposito :**
  + Autorellenar los campos como Nombre, Apellidos, Carrera, Matricul, Grado y Grupo segun el alumno que inicio sesioon
  + Tener varios select que permitan al alummno elegir el Club, Dia a la semana y Horas a las que quiere asistir
  + Y tener un boton para mandar el formulario

## **Vista actividades culturales**

def actividades\_culturales(request):

matricula = request.user.login

solicitud\_exitosa = False

if request.method == 'POST':

form = SolicitudForm(request.POST)

if form.is\_valid ():

solicitud = form.save(commit=False)

horario = solicitud.id\_horario\_club

if horario.capacidad > 0:

horario.capacidad -= 1

horario.save()

solicitud.matricula = matricula

solicitud.save()

solicitud\_exitosa = True

messages.success(request, 'Tu solicitud ha sido enviada exitosamente.')

else:

messages.error(request, 'El horario seleccionado ya no tiene capacidad.')

else:

messages.error(request, 'Por favor, corrige los errores en el formulario.')

else:

form = SolicitudForm()

try:

usuario\_acceso = UsuarioAcceso.objects.get(login=request.user)

persona = Persona.objects.get(cve\_persona=usuario\_acceso.cve\_persona)

nombre = f"{persona.nombre}"

apellidos = f"{persona.apellido\_paterno} {persona.apellido\_materno}"

alumno = Alumno.objects.get(matricula=matricula)

alumno\_grupo = (

AlumnoGrupo.objects.filter(matricula=alumno.matricula)

.select\_related('cve\_grupo\_\_cve\_carrera')

.order\_by('-cve\_grupo')

.first()

)

if alumno\_grupo:

grupo = alumno\_grupo.cve\_grupo

carrera = grupo.cve\_carrera

nombre\_carrera = carrera.nombre

nombre\_grupo = grupo.nombre

else:

nombre\_carrera = "Grupo no encontrado"

nombre\_grupo = "Grupo no encontrado"

# Pasa los datos al contexto

horarios\_disponibles = Horario.objects.filter(capacidad\_\_gt=0)

clubes = Club.objects.all()

dias = Dia.objects.all()

horas = Hora.objects.all()

context = {

'form': form,

'horarios\_disponibles': horarios\_disponibles,

'clubes': clubes,

'dias': dias,

'horas': horas,

'nombre': nombre,

'apellidos': apellidos,

'matricula': matricula,

'nombre\_carrera': nombre\_carrera,

'nombre\_grupo': nombre\_grupo ,

'solicitud\_exitosa': solicitud\_exitosa,

}

return render (request, 'pages/actividades\_culturales.html', context)

* **Proposito:** 
  + Generar una nueva solicitud con los campos que se seleccionaron
  + Con la matricula del alumno que inicio sesión se hacen consultas con otros modelos para obtener el nombre, carrera, matricula, grado y grupo
  + Manda los datos en un contexto hacia actividades\_culturales.html para usar las variables

## **Modelo Solicitud**

class Solicitud(models.Model):

id\_solicitud = models.AutoField(primary\_key=True)

id\_horario\_club = models.ForeignKey(Horario, on\_delete=models.CASCADE, db\_column='id\_horario\_club') # Relación con Horario

matricula = models.IntegerField()

fecha\_inscripcion = models.DateTimeField(auto\_now\_add=True)

ESTADOS = [

('Pendiente', 'Pendiente'),

('Aceptada', 'Aceptada'),

('Rechazada', 'Rechazada'),

]

estado = models.CharField(max\_length=20, choices=ESTADOS, default='Pendiente')

class Meta:

managed = False

db\_table = 'Solicitud'

def \_\_str\_\_(self):

return f"{self.matricula} - {self.estado}"

def get\_nombre\_completo(self):

try:

usuario = Usuario.objects.using('default').get(login=self.matricula)

persona = Persona.objects.using('default').get(cve\_persona=usuario.cve\_persona)

return f"{persona.nombre} {persona.apellido\_paterno} {persona.apellido\_materno}"

except Usuario.DoesNotExist:

return "Usuario no encontrado"

except Persona.DoesNotExist:

return "Persona no encontrada"

def get\_nombre\_carrera(self):

try:

alumno\_grupo = AlumnoGrupo.objects.using('default').filter(matricula=self.matricula).first()

if alumno\_grupo:

grupo = alumno\_grupo.cve\_grupo

return f"{grupo.nombre}"

else:

return "Grupo no encontrado"

except AlumnoGrupo.DoesNotExist:

return "Grupo no encontrado"

* **Proposito:** 
  + Se cre un modelo en base a la tabla Solicitud de la base de datos, para poder realizar consultas en base al modelo y no directamente hacia la base de datos
  + Se crearon funciones dentro del modelo para tener un codigo mas limpio en la vista.

# **Página Login.html**

## **Formulario Login**

<form method="post" action="{% url 'login:login' %}">

{% csrf\_token %}

<!-- Campo Matrícula -->

<div class="form-group mb-3">

<label class="font-weight-bold">Matrícula<span class="text-danger">\*</span></label>

<input

type="text"

name="login"

class="form-control"

placeholder="Ingresa tu matrícula"

required

/>

</div>

{% if form.login.errors %}

<div class="text-danger ml-3">

<ul class="mb-0">

{% for error in form.login.errors %}

<li>{{ error }}</li>

{% endfor %}

</ul>

</div>

{% endif %}

<div class="form-group mb-3 mt-3">

<label class="font-weight-bold">Contraseña<span class="text-danger">\*</span></label>

<input

type="password"

name="password"

class="form-control"

placeholder="Ingresa una contraseña"

required

/>

</div>

<button class="btn btn-outline-danger width-100" type="submit">

Iniciar Sesión

</button>

</form>

* **Propósito:** 
  + Muestra los campos para poner la matricula y contraseña.
  + Si algun campo esta mal muestra un mensaje de error.
  + Manda la matricula y la contraseña a la vista para ser validada.

## **Vista Login**

def login\_view(request):

if request.user.is\_authenticated:

return redirect('alumno:inicio')

form = LoginForm(request.POST or None)

if request.method == 'POST' and form.is\_valid():

login = form.cleaned\_data['login']

password = form.cleaned\_data['password']

usuario\_existente = Usuario.objects.filter(login=login, password=password).first()

if usuario\_existente:

sistema\_usuario = UsuarioAcceso.objects.filter(login=usuario\_existente.login).first()

if sistema\_usuario:

if not check\_password(usuario\_existente.password, sistema\_usuario.password):

sistema\_usuario.set\_password(usuario\_existente.password)

sistema\_usuario.save()

usuario = authenticate(request, username=login, password=password)

if usuario:

auth\_login(request, usuario)

return redirect(request.GET.get('next', 'alumno:inicio'))

else:

messages.error(request, "Por favor introduzca un nombre de usuario y contraseña correctos.")

return redirect('login:login')

else:

sistema\_usuario = UsuarioAcceso.objects.create(

cve\_persona=usuario\_existente.cve\_persona,

login=usuario\_existente.login,

activo=True,

staff=True

)

sistema\_usuario.set\_password(password)

sistema\_usuario.save()

usuario\_grupo\_seguridad = UsuarioGrupoSeguridad.objects.filter(cve\_persona=usuario\_existente.cve\_persona)

for grupo in usuario\_grupo\_seguridad:

group, created = Group.objects.get\_or\_create(name=grupo.cve\_grupo\_seguridad)

sistema\_usuario.groups.add(group)

auth\_login(request, sistema\_usuario)

return redirect(request.GET.get('next', 'alumno:inicio'))

else:

messages.error(request, "Por favor introduzca un nombre de usuario y contraseña correctos.")

return redirect('login:login')

return render(request, 'Login.html', {'form': form})

* **Propósito:**
  + Valida que según los datos recibidos coincidan con algún usuario de la base de datos.
  + En caso de que no exista el usuario crea un nuevo en sistema\_usuario.
  + Si el usuario esta autentificado lo redirige a la página siguiente según su grupo de seguridad.

## **Vista Logout**

def logout\_view(request):

auth\_logout(request)

request.session.flush()

return render(request, 'Login.html')

* **Propsotito:** 
  + **Cierra la sesión del usuario y lo redirige a la vista del login.**
* **Como funciona:** 
  + En el SideBar hay un botón que manda a llamar la ruta login:logout de las URLs de la app Login, lo cual manda a llamar esta vista que cierra la sesión.

## **Forms.py**

from django import forms

class LoginForm(forms.Form):

login = forms.CharField(max\_length=10, label='Nombre de usuario')

password = forms.CharField(widget=forms.PasswordInput, label='Contraseña')

* **Propósito:** 
  + Genera la estructura del formulario, ósea los campos que debe recibir y el tipo de datos.
  + Se puede usar también para mostrar los inputs en el formulario o solo mandar la información.
* **Como funciona:** 
  + En la vista de login, se manda a llamar el Formulario y si existe un método “POST” la información que se mando se valida con los campos y tipo de dato que se indicó en el Form.py

# **Página Creditos.html (Administrador)**

## **Estructura html**

div class="container-fluid mt-3">

<div class="row">

<!-- filtro de clubes -->

<div class="col-md-3">

<div class="card">

<div class="card-header text-center">

<h5>Filtrar por club</h5>

</div>

<div class="list-group list-group-flush p-3">

<a href="?" class="btn mb-2 {% if not selected\_club %}btn-success{% else %}btn-secondary{% endif %}">VER TODOS</a>

<a href="?club=AJEDREZ" class="btn mb-2 {% if selected\_club == 'AJEDREZ' %}btn-success{% else %}btn-secondary{% endif %}">AJEDREZ</a>

<a href="?club=RÍTMICA" class="btn mb-2 {% if selected\_club == 'RÍTMICA' %}btn-success{% else %}btn-secondary{% endif %}">RÍTMICA</a>

<a href="?club=TAEKWONDO" class="btn mb-2 {% if selected\_club == 'TAEKWONDO' %}btn-success{% else %}btn-secondary{% endif %}">TAEKWONDO</a>

<a href="?club=BAILE%20MODERNO" class="btn mb-2 {% if selected\_club == 'BAILE MODERNO' %}btn-success{% else %}btn-secondary{% endif %}">BAILE MODERNO</a>

<a href="?club=CARTA%20LABORAL" class="btn {% if selected\_club == 'CARTA LABORAL' %}btn-success{% else %}btn-secondary{% endif %}">CARTA LABORAL</a>

</div>

</div>

</div>

<!-- Tabla de créditos -->

<div class="col-md-9">

<div class="d-flex justify-content-between align-items-center mb-3">

<h3>{{ selected\_club|default:"Todos los alumnos " }}</h3>

<input type="text" class="form-control w-25" placeholder="Buscar alumnos...">

</div>

<table class="table table-striped text-center">

<thead>

<tr>

<th>Matrícula</th>

<th>Nombre</th>

<th>Carrera</th>

<th>Club</th>

<th>Créditos</th>

<th>Acciones</th>

</tr>

</thead>

<tbody>

{% for solicitud in solicitudes %}

<tr>

<td>{{ solicitud.solicitud.matricula }}</td>

<td>{{ solicitud.nombre\_completo }}</td>

<td>{{ solicitud.grupo}}</td>

<td>{{ "FALTA" }}</td>

<td>{{ "FALTA" }}</td>

<td>

<button class="btn btn-primary btn-sm">Créditos</button>

<button class="btn btn-danger btn-sm">Eliminar</button>

</td>

</tr>

</tbody>

<tfoot>

{% empty %}

<tr>

<td colspan="5">No hay solicitudes aceptadas para este club.</td>

</tr>

{% endfor %}

</tfoot>

</table>

<button class="btn btn-danger">Eliminar todo</button>

</div>

</div>

</div>

* **Propósito:**
  + En el lado izqueirdo hay botones con el nombre de los clubes que permiten filtrar a los alumnos segun el club
  + Del lado derecho se muestra una tabla con todos los alumnos, mostrando la matricula, nombre, carrera, club, creditos y las acciones que son editar los creditos y eliminar al alumno.

## **Vista de créditos**

def creditos(request):

selected\_club = request.GET.get('club', '')

solicitudes = Solicitud.objects.filter(estado='Aceptada')

if selected\_club:

solicitudes = solicitudes.filter(id\_horario\_club\_\_id\_club\_\_nombre\_club=selected\_club)

solicitudes = [

{

'solicitud': solicitud,

'nombre\_completo': solicitud.get\_nombre\_completo(),

'grupo': solicitud.get\_nombre\_carrera(),

}

for solicitud in solicitudes

]

return render(request, 'pages/creditos.html', {

'selected\_club': selected\_club,

'solicitudes': solicitudes,

})

* **Proósito:**
  + Filtra las solicitudes en base al id\_club.
  + Obtien el nombre de los alumnos, grado y grupo con la matricula.

## **Objetivo**

El administrador debe poder ver todos los alumnos que están dentro de un club, esto con la finalidad de poder asignar créditos en caso de que hayan realizado actividades o llevados materiales a cambio de créditos. El administrador debe poder filtrar por club, nombre, grado o grupo

## **Pendiente y Observaciones**

Actualmente solo se muestran los alumnos en base a la tabla de Solicitudes que tienen un estado de aceptadas. Se debe hacer el cambio de que cuando se acepten los alumnos se envíen a una tabla que contenga a los alumnos y los clubes a los que pertenecen.

# **Página Carta\_laboral.html (Administrador)**

## **Estructura html**

<div class="container-fluid">

<div class="row mt-4">

<div class="col-md-9 ">

<h3 class="mb-3">Socilitudes</h3>

<table class="table text-center">

<thead>

<tr>

<th></th>

<th>MATRICULA</th>

<th>NOMBRE</th>

<th>GRADO</th>

<th>GRUPO</th>

<th>DETALLES</th>

</tr>

</thead>

<tbody>

<tr>

<td><input type="checkbox" checked></td>

<td>612110393</td>

<td>Angel Fernando Muñoz Castorena</td>

<td>TIS</td>

<td>4C</td>

<td> <button class="btn btn-primary">VER</button></td>

</tr>

<tr>

<td><input type="checkbox" checked></td>

<td>612110393</td>

<td>Bryan Rodrigo Gomez Coronado</td>

<td>TIS</td>

<td>2B</td>

<td> <button class="btn btn-primary">VER</button></td>

</tr>

<tr>

<td><input type="checkbox" checked></td>

<td>612110393</td>

<td>Omar Lirachi Ramirez</td>

<td>TIS</td>

<td>3A</td>

<td> <button class="btn btn-primary">VER</button></td>

</tr>

</tbody>

</table>

</div>

<div class="col-md-3 rounded-4 border">

<h3 class="text-center">Detalles del Alumno</h3>

<div class="p-4">

<strong>CARTA LABORAL</strong>

<p class="bg-dark p-2 text-white rounded text-center mt-2" style="opacity: 25%;">Archivo Cargado</p>

<strong>INE DEL JEFE:</strong>

<p class="bg-dark p-2 text-white rounded text-center mt-2" style="opacity: 25%;">Archivo Cargado</p>

</div>

<div class="d-flex flex-column pl-4 pr-4 mb-2">

<button class="btn btn-danger mb-2">CANCELAR SOLICITUD</button>

<button class="btn btn-success mb-2">AGREGAR ALUMNO</button>

</div>

</div>

</div>

</div>

* **Propósito:**
  + Mostrar una bandeja con las cartas laborales del alumno.
  + Al seleccionar una solicitud se deberan ver dos documentos que el alumno alla cargado.

## **Vista carta laboral**

def carta\_laboral(request):

return render(request, 'pages/carta\_laboral.html')

* **Propósito:**
  + Renderiza el archivo html de la carta\_laboral

## **Objetivo**

El objetivo es consultar un modelo que contenga todas las cartas laborales enviadas por el alumno, debe contener la matricula para relacionarla con los modelos que tengan el nombre del alumno, carrera, grado, grupo, la carta laboral, el INE del jefe del trabajo, la fecha y el estado de la solicitud. El administrador debe poder ver los documentos y cambiar el estado de la solicitud a aceptado o rechazado. Para luego agregarlo a una tabla como un club de carta laboral. Y avisarle al alumno si fue aceptado o rechazado.

# **Página Solicitudes.html (Profesor)**

## **Estructura html de solicitudes**

<tr data-id="{{ dato.id }}" class="solicitud">

<td>{{ dato.matricula }}</td>

<td>{{ dato.nombre }}</td>

<td>{{ dato.carrera }}</td>

<td>{{ dato.grupo }}</td>

<td id="estado-{{ dato.id }}">{{ dato.estatus }}</td>

<td>{{ dato.fecha }}</td>

<td>{{ dato.comentarios }}</td>

<td>

<div class="d-flex gap-2 justify-content-center align-items-center">

{% if dato.estatus == "Pendiente" %}

<form method="POST" action="{% url 'profesor:aceptar\_solicitud' dato.id %}">

{% csrf\_token %}

<button class="btn btn-success btn-sm mr-2" type="submit">Aceptar</button>

</form>

<form method="POST" action="{% url 'profesor:rechazar\_solicitud' dato.id %}">

{% csrf\_token %}

<button class="btn btn-danger btn-sm" type="submit">Rechazar</button>

</form>

{% elif dato.estatus == "Aceptada" %}

<form method="POST" >

{% csrf\_token %}

<button class="btn btn-danger btn-sm " type="submit">Eliminar</button>

</form>

{% elif dato.estatus == "Rechazada" %}

<form method="POST" action="{% url 'profesor:revertir\_solicitud' dato.id %}">

{% csrf\_token %}

<button class="btn btn-warning btn-sm" type="submit">Revertir</button>

</form>

{% endif %}

</div>

</td>

</tr>

* **Proósito:**
  + Se muestra la tabla con la informacion de las solicitudes.
  + Por cada solicitud hay botones para aceptar o rechazar, segun el estado se muestra un boton de revertir o eliminar.

## **Vista de solicitudes**

def solicitudes(request):

loginuser = request.session.get('loginuser')

solicitudes = Solicitud.objects.all()

datos\_completos = []

mensaje\_modal = request.session.pop('mensaje\_modal', None) # Tomar y eliminar el mensaje de la sesión

for solicitud in solicitudes:

usuario\_acceso = UsuarioAcceso.objects.filter(login=solicitud.matricula).first()

if usuario\_acceso:

persona = Persona.objects.filter(cve\_persona=usuario\_acceso.cve\_persona).first()

if persona:

alumno = Alumno.objects.filter(matricula=solicitud.matricula).first()

if alumno:

alumno\_grupo = (

AlumnoGrupo.objects.filter(matricula=alumno.matricula)

.select\_related('cve\_grupo\_\_cve\_carrera')

.order\_by('-cve\_grupo')

.first()

)

if alumno\_grupo:

grupo = alumno\_grupo.cve\_grupo

carrera = grupo.cve\_carrera

nombre\_grupo = grupo.nombre

nombre\_carrera = carrera.nombre

else:

nombre\_grupo = "Grupo no encontrado"

nombre\_carrera = "Carrera no encontrada"

fechaFormateada = solicitud.fecha\_inscripcion.strftime('%d/%m/%Y')

datos\_completos.append({

'matricula': solicitud.matricula,

'nombre': f"{persona.nombre} {persona.apellido\_paterno} {persona.apellido\_materno}",

'carrera': nombre\_carrera,

'grupo': nombre\_grupo,

'estatus': solicitud.estado,

'fecha': fechaFormateada,

'id' : solicitud.id\_solicitud,

'loginuser': loginuser

})

return render(request, 'pages/solicitudes.html', {'datos\_completos': datos\_completos, 'mensaje\_modal': mensaje\_modal})

* **Propósito:**
  + Se obtienen las solicitudes
  + Se obtienen los datos extras en base a la matricula de las solicitudes.

## **Vistas para cambiar el estado**

def aceptar\_solicitud(request, solicitud\_id):

if request.method == 'POST':

solicitud = get\_object\_or\_404(Solicitud, id\_solicitud=solicitud\_id)

solicitud.estado = 'Aceptada'

solicitud.save()

request.session['mensaje\_modal'] = 'La solicitud ha sido aceptada.'

return redirect('profesor:solicitudes')

def rechazar\_solicitud(request, solicitud\_id):

if request.method == 'POST':

solicitud = get\_object\_or\_404(Solicitud, id\_solicitud=solicitud\_id)

solicitud.estado = 'Rechazada'

solicitud.save()

request.session['mensaje\_modal'] = 'La solicitud ha sido rechazada.'

return redirect('profesor:solicitudes')

def revertir\_solicitud(request, solicitud\_id):

if request.method == 'POST':

solicitud = get\_object\_or\_404(Solicitud, id\_solicitud=solicitud\_id)

solicitud.estado = 'Pendiente'

solicitud.save()

request.session['mensaje\_modal'] = 'La solicitud ha sido revertida.'

return redirect('profesor:solicitudes')

def eliminar\_solicitud(request, solicitud\_id):

if request.method == 'POST':

solicitud = get\_object\_or\_404(Solicitud, id\_solicitud=solicitud\_id)

solicitud.delete()

request.session['mensaje\_modal'] = 'La solicitud ha sido eliminada.'

return redirect('profesor:solicitudes')

* **Propósito:** 
  + Se tiene la vista acpetar solcitud, cambia el estado de la solicitud a aceptada y devuelve un mensaje que se muestra en una tarjeta en la parte inferior derecha de la página.
  + Se tiene la vista de rechazar solicitud, cambia el estado y devuelve un mensaje que se muestra en una tarjeta.
  + Se tiene la vista revertir que cambia el estado de la solicitud.
  + Se tiene la vista eliminar que cambia el estado de la solicitud.

## **Objetivo**

El profesor podrá ver las solicitudes que tengan el club que el imparte, estas solicitudes se verán en forma de tabla que contendrán la matricula del alumno que envió la solicitud, la carrera, grupo, estatus, fecha de la solicitud, comentarios de si les faltan créditos y botones para aceptar la solicitud, rechazarla, revertir en caso de haberla rechazado o eliminar. Además de poder filtrar las solicitudes por el estado, el nombre o fecha.

## **Pendiente y Observaciones**

Falta que se pueda filtrar por la fecha, se pueden cambiar los estados de la solicitud correctamente.

# **Profesor.html (Profesor)**

## **Estructura html de solicitudes**

  <div *class*="container-fluid" *id*="barra">

    <div *class*="row align-items-center justify-content-center bg-primary">

      <div *class*="col d-flex justify-content-center align-items-center">

        <b>Alumnos:</b>

        <p *class*="m-0 ps-1" *id*="alumnos"></p>

      </div>

      <div *class*="col d-flex justify-content-center align-items-center">

        <b>Actividad:</b>

        <p *class*="m-0"> -----</p>

      </div>

      <div *class*="col d-flex justify-content-center align-items-center">

        <a *href*="#" *class*="btn" *role*="button" *data-bs-toggle*="button">

          <i *class*="bi bi-pencil fs-5" *data-bs-toggle*="tooltip" *data-bs-title*="Editar asistencias"></i>

        </a>

        <a *href*="#" *class*="btn">

          <i *class*="bi bi-file-earmark-arrow-down fs-5" *data-bs-toggle*="tooltip" *data-bs-title*="Descargar reporte"></i>

        </a>

      </div>

      <div *class*="col">

        <fieldset>

          <div *class*="">

            <select *name*="dia" *id*="dia" *class*="form-select form-select-sm">

              <option *selected*="selected" *value*="Seleccionar día">Seleccionar día</option>

              <option *value*="GENERAL">General</option>

              {% for dia in dias %}

              <option *value*="{{ dia }}">{{ dia }}</option>

              {% endfor %}

            </select>

          </div>

        </fieldset>

      </div>

      <div *class*="col">

        <fieldset>

          <div *class*="">

            <select *name*="hora" *id*="hora" *class*="form-select form-select-sm">

              {% for hora in horas %}

              <option *value*="{{ hora }}">{{ hora }}</option>

              {% endfor %}

            </select>

          </div>

        </fieldset>

      </div>

    </div>

  </div>

  <div *id*="resultado" *class*="p-3 overflow-y-auto">

    <div *class*="container-fluid d-flex flex-column justify-content-center align-items-center">

      <h4>Bienvenido docente, selecciona un día para ver la asistencia.</h4>

      <img *class*="border rounded"  *width*="300" *alt*="">

    </div>

  </div>

* **Propósito:**
  + Muestra la barra que contiene información como cantidad de Alumnos, seleccionar el día y horario para filtrar.
  + El contenedor resultado es lo que se le imprime según el día seleccionado (también debería ser por horario pero aún no se establece)
  + Profesor.html es la página principal de inicio para la vista de profesor

## **Vista**

# Create your views here.

@profesor\_required

def inicio(*request*):

    loginuser = *request*.session.get('loginuser')

    horas = Hora.objects.all()

    dia = Dia.objects.all()

    archivo\_path = os.path.join(os.getcwd(), 'Profesor', 'static', 'assets', 'asistencia.xlsx')

    if not os.path.exists(archivo\_path):

        raise FileNotFoundError(f"El archivo no se encontró: {archivo\_path}")

    # SOLICITUD AJAX

    if *request*.headers.get('x-requested-with') == 'XMLHttpRequest':

        dia = *request*.GET.get('dia', '')

        excel, cantidad\_alumnos = extraer\_excel(dia, archivo\_path)

        html\_stream = io.StringIO(excel)

        tablas = pd.read\_html(html\_stream)

        print(f'TABLAS: \n {tablas}')

        df = tablas[0]

        return JsonResponse( {'html': excel, 'alumnos': cantidad\_alumnos})

    if *request*.method == 'POST':

        form = ArchivoExcelForm(*request*.POST, *request*.FILES)

        if form.is\_valid():

            archivo\_subido = form.cleaned\_data['archivo']

            try:

                archivo\_existente = ArchivoExcel.objects.get(*usuario*=*request*.user)

                if default\_storage.exists(archivo\_existente.documento.name):

                    archivo\_existente.documento.delete()

                archivo\_existente.documento = archivo\_subido

                archivo\_existente.save()

            except ArchivoExcel.DoesNotExist:

                ArchivoExcel.objects.create(

*usuario*=*request*.user,

*documento*=archivo\_subido

                )

            return render(*request*, 'profesor.html', {

                'loginuser': loginuser,

                'form': form,

                'message': 'Archivo subido y actualizado correctamente'

            })

    context= {

        'loginuser': loginuser,

        'horas': horas,

        'dias': dia,

        'mapache\_docente': static('assets/mapache\_docente.jpeg')

    }

    return render(*request*, 'profesor.html', context)

* **Propósito:**
  + Se obtiene el usuario, horas, días y una imagen
  + Se realiza la solicitud Ajax que detecta cambios cuando se seleccionar un día diferente (Se complementa con código JavaScript)
  + Imprime en la página la tabla según los datos tomados en el archivo Excel dentro de Static/assets/asistencias.xlsx

## **Vista/Función para extraer el excel**

def extraer\_excel(*dia*, *archivo*):

    excluir = []

    excluir2 = ['Unnamed: 0']

*dia* = *dia*.upper()

    print(f'dia: {*dia*}')

    for i in range(1, 31):

        excluir.append(i)

    # OBTENER TABLA DEL EXCEL

    df\_alumnos = pd.read\_excel(

*archivo*,

*sheet\_name*=*dia*.upper(),

*header*=7,

*index\_col*=0,

*usecols*=lambda *col*: *col* not in excluir and *col* not in excluir2,

    )

    # RENOMBRAR COLUMNAS

    df\_alumnos.rename(*columns*={'Unnamed: 38': 'ESTATUS'}, *inplace*=True)

    df\_alumnos.rename(*columns*={'T': 'HORAS'}, *inplace*=True)

    # CONCATENAR COLUMNAS CARRERA, GRUPO Y GRADO

    # OBTENER EL GRADO PARA QUITAR VALORES FLOAT

    df\_alumnos['GRADO'] = df\_alumnos[['GRADO']].fillna('').astype(str)

    df\_alumnos['GRADO'] = df\_alumnos['GRADO'].apply(lambda *x*: *x*[0] if len(*x*) > 0 else '')

    df\_alumnos['HORAS'] = df\_alumnos['HORAS'].fillna(0).astype(int)

    # QUITAR ESPACIOS EN BLANCO A CARRERA

    df\_alumnos['CARRERA'] = df\_alumnos['CARRERA'].apply(lambda *x*: str(*x*).strip() if isinstance(*x*, str) else str(*x*).strip())

    df\_alumnos['CARRERA'] = df\_alumnos['CARRERA'].apply(lambda *x*: str(*x*).upper())

    # ESTABLECER LOS NOMBRES EN MAYUSCULA

    df\_alumnos['NOMBRE(S)'] = df\_alumnos['NOMBRE(S)'].apply(lambda  *x*: str(*x*).rstrip().upper() if isinstance(*x*, str) else str(*x*).rstrip().upper())

    # PARA SOLO MOSTRAR LOS ALUMNOS QUE ESTEN EN EL DOCUMENTO (HASTA DONDE MATRICULA NO ES IGUAL A NaN)

    df\_matricula= df\_alumnos[['MATRICULA']].fillna(0).astype(int) # ESTABLECE EN CERO LOS VALORES NaN

    df\_matricula = df\_matricula[df\_matricula['MATRICULA'] != 0] # FILTRA TODOS LOS QUE SEAN DIFERENTE DE 0

    df\_matricula = df\_matricula.loc[df\_matricula.index]

    df\_info = df\_alumnos.loc[df\_matricula.index]

    df\_alumnos.drop(*columns*=['MATRICULA'], *inplace*=True)

    df\_info.drop(*columns*=['MATRICULA'], *inplace*=True)

    df\_info = pd.concat([df\_matricula.reset\_index(*drop*=True), df\_info.reset\_index(*drop*=True)], *axis*=1)

    print(df\_info)

    # CONCATENA ASISTENCIA Y MATRICULA

    html\_table = df\_info.to\_html(*classes*='table table-sm table-hover', *index*=False, *escape*=False)

    html\_table = html\_table.replace('<td>', '<td contenteditable="true">')

    cantidad\_inscritos = df\_info.shape[0]

    return html\_table, cantidad\_inscritos

**Propósito:**

**Entrada:**

* **dia: Nombre de la hoja del Excel a procesar.**
* **archivo: Archivo Excel con datos de alumnos.**

**Procesamiento:**

* **Excluye columnas innecesarias por nombre o índice.**
* **Carga la hoja del Excel especificada con encabezado y columnas válidas.**
* **Renombra columnas (Unnamed: 38 → ESTATUS, T → HORAS).**

**Limpia datos:**

* **Formatea GRADO, CARRERA y NOMBRE(S) (quita espacios, normaliza texto en mayúsculas).**
* **Convierte valores NaN a predeterminados (0 o cadenas vacías).**
* **Filtra filas con matrículas válidas y elimina columnas innecesarias.**

**Salida:**

* **Tabla HTML: Editable y estilizada con Bootstrap.**
* **Cantidad de inscritos: Número de registros válidos.**

## **Objetivo**

El docente verá la página de inicio donde podrá filtrar por día y horario, así como manejar a través de archivos Excel según el formato establecido, así como realizar y anotar las horas realizadas según por Alumno.

## **Pendientes y Observaciones**

No está terminado, faltan establecer detalles.

## **Archivo Script (script\_profesor.js)**

document.addEventListener("DOMContentLoaded", function () {

    const dia = document.getElementById("dia");

    const contenedor\_resultado = document.getElementById("resultado");

    const alumnos = document.getElementById('alumnos');

    function actualizarResultado() {

        const diaSeleccionado = dia.value;

        if (diaSeleccionado !== "Seleccionar día") {

            alumnos.innerHTML = '...';

            contenedor\_resultado.innerHTML = `<div class="text-muted text-center">Cargando datos...</div>`;

            fetch(`?dia=${diaSeleccionado}`, {

                headers: {

                    'X-Requested-With': 'XMLHttpRequest'

                }

            })

                .then(*response* => {

                    if (!*response*.ok) {

                        throw **new** *Error*(`HTTP error! Status: ${*response*.status}`);

                    }

                    return *response*.json();  // Cambiar a .json() para respuesta JSON

                })

                .then(*data* => {

                    console.log('JSON recibido:', *data*);

                    contenedor\_resultado.innerHTML = *data*.html;  // Inserta el HTML

                    alumnos.innerHTML = *data*.alumnos;           // Actualiza el contador

                })

                .catch(*error* => {

                    console.error('Hubo un problema con la solicitud AJAX:', *error*);

                    contenedor\_resultado.innerHTML = `

                    <div class="alert alert-danger text-center">

                        <strong>Error:</strong> No se pudieron cargar los datos. Intenta más tarde.

                    </div>`;

                });

        } else {

            contenedor\_resultado.innerHTML = `

            <div class="container d-flex flex-column justify-content-center align-items-center">

                <h4>Por favor, selecciona un día para cargar la asistencia.</h4>

                <img class="border rounded" src="{{ mapache\_docente }}" width="300" alt="Docente">

            </div>`;

            alumnos.innerHTML = "---";

        }

    }

    dia.addEventListener("change", actualizarResultado);

});

**FUNCIONA PARA SEGÚN EL VALOR DEL DIA SELECCIONADO MUESTRE Y AGREGUE LA INFORMACIÓN PARA SER MOSTRADA EN EL HTML**