

## **CARRERA DE INGENIERÍA EN SOFTWARE**



### **APLICACIONES WEB**

#### **Autores:**

Muñiz Rivas Leopoldo Miquel

#### **Tema:**

**Control de asistencia para docentes de la uleam**

#### **Curso:**

4 - A

**Periodo 2025(1)**

## INDICE

1.	Repositorio.....	1
2.	Pagina alojada.....	1
3.	Resumen.....	2
3.1.	Finalidad del sistema y características principales .....	2
3.2.	Función básica.....	2
3.3.	Resumen de la arquitectura del sistema.....	2
3.4.	Patrón de Arquitectura de Componente .....	3
3.5.	Usuarios y casos de uso de destino.....	4
3.6.	Casos de uso primario.....	4
3.7.	Apiladora de la tecnología .....	4
3.8.	Principios clave del diseño.....	5
4.	Arquitectura de sistemas.....	5
4.1.	Finalidad y alcance.....	5
4.2.	Arquitectura general del sistema.....	6
4.3.	Diagrama de capa del sistema.....	8
4.1.3	Patrón de Arquitectura de Componente .....	8
4.2.3	Patrón de Estructura de Componentes.....	8
4.3.3	Ejemplos de ejecución de componentes.....	8
4.4.	Arquitectura de flujo de datos.....	9
4.1.4	Patrón de acceso a los datos .....	9
4.5.	Interacciones básicas del sistema.....	9
4.1.5	Asistencia Gestión de asistencia Flujo .....	10

4.6.	Flujo del sistema de navegación .....	10
4.7.	Principios arquitectónicos clave.....	11
4.1.7	Componente Independencia .....	11
4.2.7	Centralización de la gestión de datos .....	11
4.8.	Evento-Driven Arquitectura.....	12
5.	Gestión de asistencia.....	13
5.1.	Finalidad y alcance.....	13
5.2.	Resumen del Asistencia .....	13
5.3.	Añadir registros de asistencia .....	14
5.1.3	Añadir el componente de Asistencia .....	14
5.2.3	Asistencia Grabación Lógica Flujo .....	15
5.1.3	Funciones clave y operaciones de datos .....	16
5.4.	Mostrando registros de asistencia .....	16
5.1.4	Componente de visualización .....	16
5.1.4	Estructura de plantilla y vinculación de datos .....	17
5.1.4	Generación y actualización de servicios públicos .....	18
5.5.	Flujo e integración de datos .....	19
5.1.5	Integración con Capa de Datos .....	19
5.1.5	Coordinación de Refresca de IU .....	20
5.6.	Componentes de IU y Estilo .....	21
5.1.6	Pantalla con tarjeta.....	21
5.2.6	Añadir botón interactivo .....	22
6.	Gestión de los Horarios.....	23
6.1.	Finalidad del sistema y alcance.....	23
6.2.	Arquitectura de componentes.....	24
6.3.	Estructura de datos y plantilla de refuerzo.....	24

6.1.3	Cuadro de datos de calendario .....	24
6.2.3	Implementación de la función de plantilla .....	25
6.4.	Generación y gestión de calendarios.....	25
6.1.4	Funciones básicas .....	25
6.5.	Diseño visual y diseño .....	27
6.1.5	Estructura de la tarjeta de calendario .....	27
6.6.	Especificaciones de Estilo CSS.....	27
6.7.	Conducta sensible .....	28
6.8.	Integración con Capa de Datos .....	28
6.1.8	Patrón de acceso a los datos .....	28
6.9.	DOM Integración y Gestión de contenedores.....	29
6.1.9	Elementos de contenedores .....	29
6.2.9	Cargando del módulo.....	30
6.10.	Relación con el sistema de asistencia .....	30
7.	Sistema de autenticación.....	30
7.1.	Finalidad y alcance.....	30
7.2.	Autación Flujo.....	31
7.3.	Login de arquitectura de componentes .....	31
7.4.	Forma la generación de campo .....	32
7.5.	Proceso de autenticación.....	33
7.6.	Arquitectura de componentes de sesión.....	33
7.7.	Gestión estatal de Button .....	34
7.8.	Integración con Capa de Datos .....	35

7.9.	Integración de lanzamiento.....	36
8.	Perfil del profesor.....	36
8.1.	Finalidad y alcance.....	36
8.2.	Reseña del componente.....	37
8.3.	Estructura de componentes .....	37
8.4.	Flujo e integración de datos .....	38
8.5.	Proceso de recuperación de datos .....	38
9.	Sistema de navegación.....	39
9.1.	Finalidad y alcance.....	39
9.2.	Estructura de componentes de navegación .....	39
9.1.2	Arquitectura de componentes .....	40
9.3.	Integración de rutas y generación de botones .....	40
9.4.	Flujo de procesamiento de rutas.....	41
9.5.	Aplicación del botón de navegación.....	42
9.6.	Proceso de creación de botones .....	42
9.7.	Diseño visual y posicionamiento .....	43
9.1.7	Especificaciones de diseño .....	43
9.8.	Integración e Inicialización de Componentes .....	43
9.1.8	Secuencia de la inicialización .....	44
10.	Componentes de la interfaz de usuario .....	44
10.1.	Arquitectura de componentes.....	45
10.1.1	Estructura de archivos de componentes.....	45
10.1.1	Patrón de Integración de Componentes .....	47

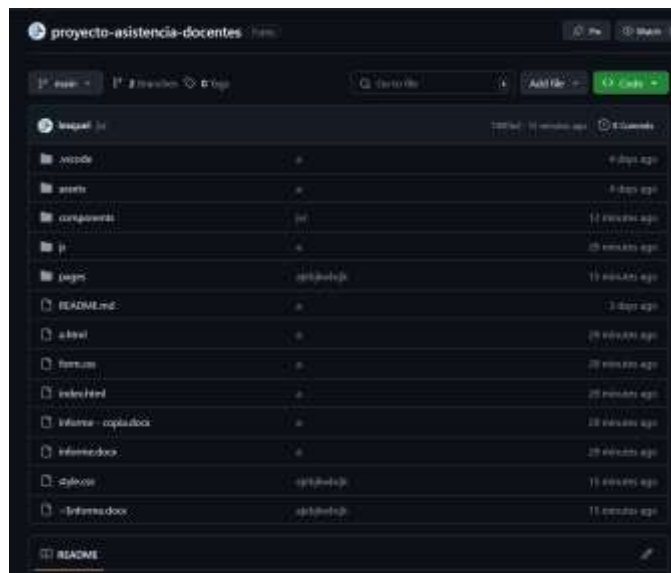
10.2.	Componentes de IU disponibles .....	47
▪	Estructura de componentes de calendario .....	47
10.1.2	Estructura de componentes de logotipo .....	48
10.3.	Características de diseño sensibles .....	49
10.1.3	Calendar la respuesta .....	49
10.4.	Arquitectura de estilo.....	50
10.1.4	Patrón de Organización del CSS .....	50
10.2.4	Esquema de color .....	50
10.5.	Integración de componentes.....	50
11.	Activos y recursos.....	51
○	Estructura de Directorio de Activos.....	51
11.1.	Activos de imagen.....	52
11.2.	Activos de Perfil del Maestro.....	53
11.3.	Navegador y Navegación activos.....	53
11.1.3	Favicon .....	53
11.2.3	Icono de Home .....	53
11.4.	Carga de activos e integración .....	54
11.1.4	Estándas de formato de archivo .....	54
11.2.4	Convenios de nombramiento .....	54
11.5.	Mejores prácticas de la Organización de Activos .....	55
12.	Configuración de desarrollo.....	55
12.1.	Requisitos del entorno para el desarrollo.....	56
12.1.1	Herramientas requeridas .....	56
12.2.	Configuración de servidor de desarrollo.....	57
12.3.	Proyecto de Arquitectura para el Desarrollo .....	57



12.4.	Flujo de trabajo de desarrollo de componentes.....	58
12.5.	Flujo de trabajo para el desarrollo.....	58
12.6.	Consideraciones clave para el desarrollo.....	59
12.7.	Medidas de configuración del entorno para el desarrollo .....	60
12.8.	Requisitos del navegador .....	60

## 1. Repositorio

link del repositorio: <https://github.com/lesquel/proyecto-asistencia-docentes.git>



## 2. Pagina alojada

La página está en github pages. Enlace de la página: <https://lesquel.github.io/proyecto-asistencia-docentes/>





### 3. Resumen

#### 3.1. Finalidad del sistema y características principales

El sistema de gestión de la asistencia al profesorado es una aplicación web del lado del cliente que permite a las instituciones educativas rastrear digitalmente la asistencia al profesorado y gestionar los horarios de enseñanza. El sistema proporciona una interfaz en español para registrar las entradas de asistencia, ver datos históricos de asistencia y gestionar los horarios de enseñanza.

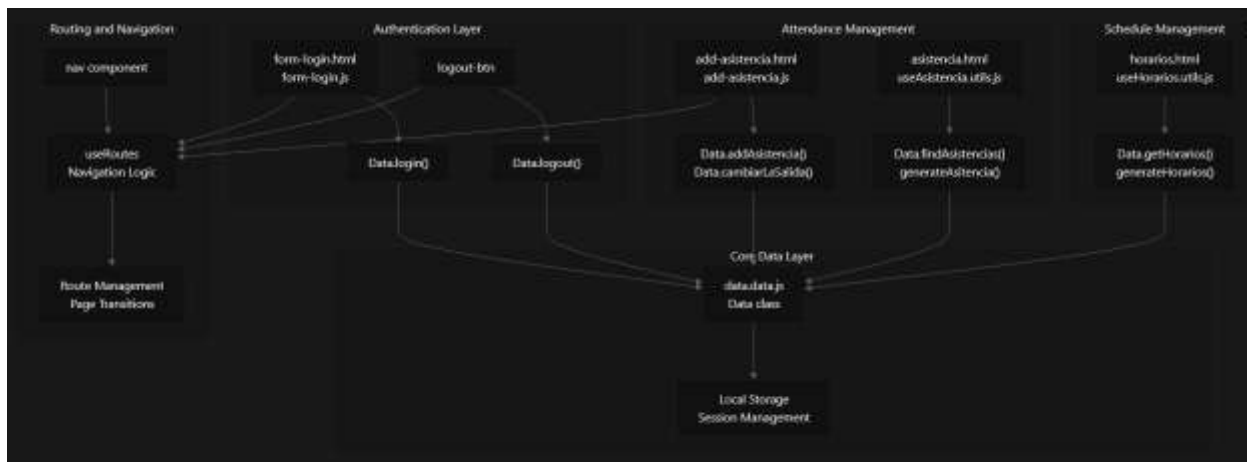
#### 3.2. Función básica

Área de características	Descripción	Componentes primarios
Registro de asistencia	Los profesores pueden entrar y salir, con seguimiento automático de los horarios de entrada y salida	add-asistencia, asistencia
Gestión de los cuadros	Pantalla y gestión de los horarios docentes con integración al seguimiento de asistencia	horarios
Autenticación del usuario	Distribuye funcionalidad de inicio de sesión y logout para profesores	form-login, logout-btn
Gestión del perfil	Mostración de perfil docente y gestión de la información	Docente
Navegación	Enrutamiento de aplicaciones y navegación entre diferentes áreas funcionales	nav, sistema de enrutamiento

#### 3.3. Resumen de la arquitectura del sistema

El siguiente diagrama ilustra la arquitectura de alto nivel y cómo las entidades de código mapean las áreas funcionales:

## Arquitectura de sistemas de alto nivel



### 3.4.Patrón de Arquitectura de Componente

El sistema sigue una arquitectura consistente basada en componentes donde cada área funcional se implementa como un módulo autónomo con archivos dedicados HTML, CSS y JavaScript.

### Estructura de componentes y flujo de datos



### 3.5. Usuarios y casos de uso de destino

El sistema está diseñado para la ULEAM, con la que se dirige específicamente a:

- **Maestros:** Usuarios de primaria que registran su asistencia diaria y ven sus horarios

### 3.6. Casos de uso primario

1. **Grabación diaria de asistencia:** Maestros usan el add-asistenciacomponente para atracar al inicio de su jornada laboral y desgastado al salir
2. **Historia de asistencia Revisión :** Maestros y administradores pueden ver registros de asistencia histórica a través de la asistenciacomponentes
3. **Gestión del horario:** Los usuarios pueden ver los horarios de enseñanza y los próximos compromisos a través de la horarioscomponentes
4. **Gestión de sesiones:** La funcionalidad segura de inicio de sesión y logout garantiza un control de acceso adecuado

### 3.7. Apiladora de la tecnología

El sistema se implementa como una aplicación web del lado del cliente utilizando:

- **Frontend :** HTML5, CSS3, y vainilla JavaScript con módulos ES6
- **Arquitectura :** Arquitectura basada en componentes con generación de IU impulsada por plantillas
- **Gestión de datos:** Almacenamiento y gestión de datos del lado del cliente a través de un Dataclase
- **Ruta :** Sistema de enrutamiento personalizado del lado del cliente (useRoutes)

- **Plantillas** : Elementos de plantilla HTML para la generación de contenido dinámico (trabajación, tarjeta de tarea)

### 3.8.Principios clave del diseño

Principio	Aplicación	Ejemplo
Aislamiento del componente	Cada área de características tiene archivos HTML, CSS y JS dedicados	<a href="#">componentes/add-asistencia/add-asistencia.html1</a> <a href="#">componentes/asistencia/asistencia.html1</a>
IDT de plantilla-diverso	Generación de contenido dinámico con plantillas HTML	<a href="#">componentes/asistencia/asistencia.html5 a 16</a> <a href="#">componentes/horarios/horarios.html2 a 12</a>
Gestión centralizada de los datos	Unico Dataclase maneja todas las operaciones de datos	Referenciada en funciones de utilidad en los componentes
JavaScript modular	Módulos ES6 con clara separación de preocupaciones	<a href="#">componentes/asistencia/asistencia.html21-22</a> <a href="#">componentes/horarios/horarios.html17 a 18</a>

## 4. Arquitectura de sistemas

### 4.1.Finalidad y alcance

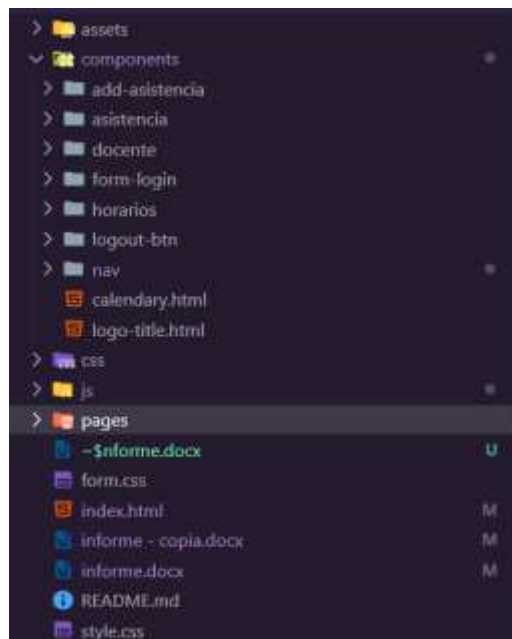
La arquitectura general del sistema del sistema de gestión de la asistencia al profesorado, incluyendo organización de componentes, patrones de flujo de datos e interacciones del sistema. Abarca el diseño estructural de alto nivel y los patrones arquitectónicos utilizados en toda la base de código.

Para obtener información detallada sobre los componentes funcionales específicos, consulte Componentes Core. Para las especificaciones del componente de IU, consulte Componentes de UI. Para la configuración del medio ambiente de desarrollo, vea Development Setup.

## 4.2.Arquitectura general del sistema

El sistema sigue una arquitectura basada en componentes con tres capas primarias:

Componentes Frontend, Infraestructura Básica y Gestión de Datos.

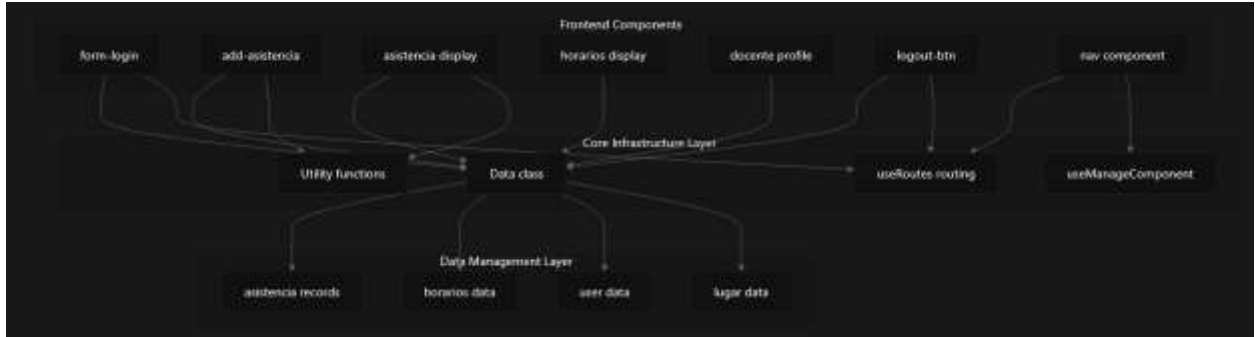


### a) Components/

- **Propósito:** Contiene componentes reutilizables del frontend, como secciones específicas de la interfaz de usuario.
- **Subcarpetas:**
  - add-asistencia/: Puede tener componentes o scripts para agregar asistencia.
  - asistencia/: Relacionado con la visualización o manejo de asistencia.
  - docente/: Lógica o interfaz para los docentes (profesores).
  - form-login/: Componentes para el formulario de inicio de sesión.
  - horarios/: Manejo o visualización de horarios.

- logout-btn/: Botón para cerrar sesión.
- nav/: Barra de navegación u opciones de menú.
- **Archivos:**
  - calendar.html: Posiblemente una vista o widget de calendario.
  - logo-title.html: Un fragmento HTML que muestra el logo o título del sitio/aplicación.
- b) css
- **Propósito:** Almacena hojas de estilo (CSS) utilizadas para dar formato visual a las páginas.
- c) js/
- **Propósito:** Contiene archivos JavaScript que controlan la interacción del usuario, validaciones, eventos, etc.
- d) pages/
- **Propósito:** Aquí podrían estar las páginas completas del sitio web, como vistas independientes.
- **Nota:** No se muestra contenido dentro, pero normalmente incluiría archivos .html.

### 4.3. Diagrama de capa del sistema



Fuentes: [componentes/add-asistencia/add-asistencia.js1 a 8](#)

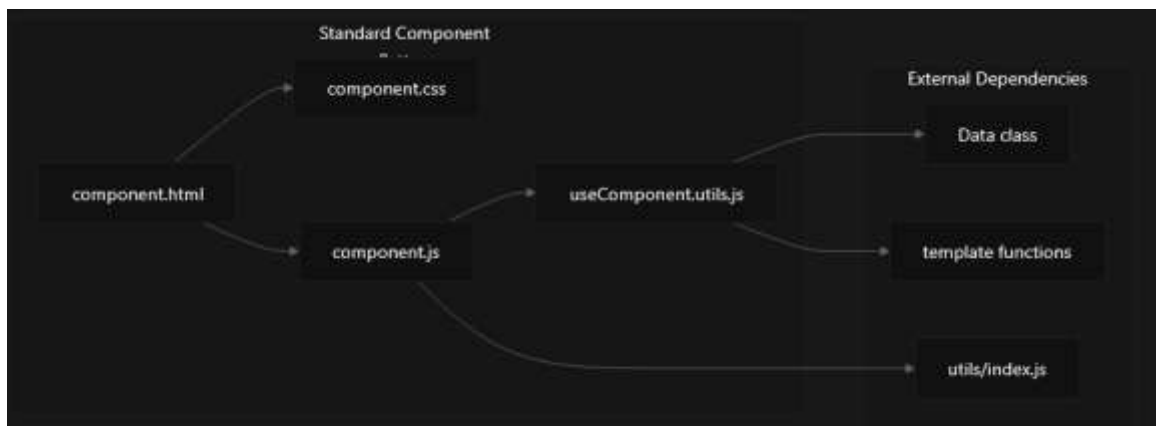
[componentes/asistencia/useAsistencia.utils.js1 a 4](#)

[componentes/horarios/useHorarios.utils.js1 a 4](#) [componentes/nav/nav.js1 a 3](#)

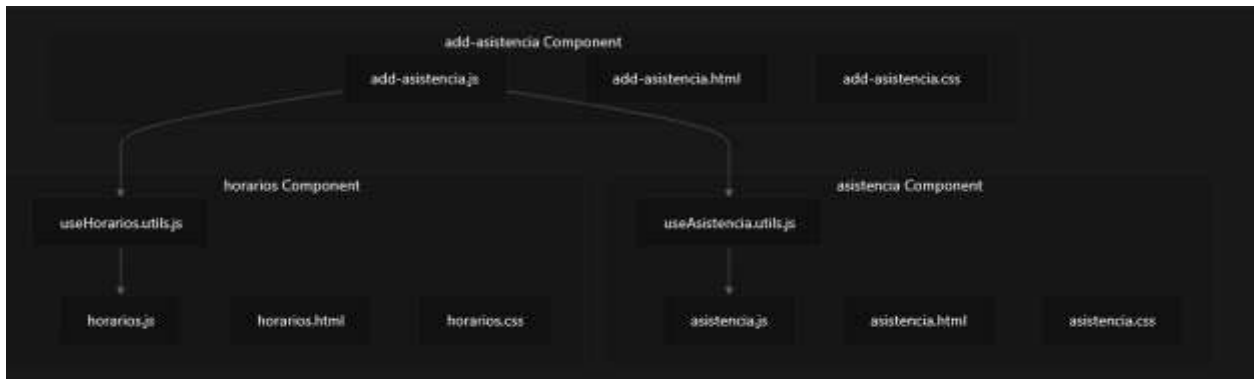
#### 4.1.3 Patrón de Arquitectura de Componente

Cada componente del sistema sigue un patrón estructural consistente con la separación de preocupaciones entre la presentación, el comportamiento y la lógica empresarial.

#### 4.2.3 Patrón de Estructura de Componentes



#### 4.3.3 Ejemplos de ejecución de componentes



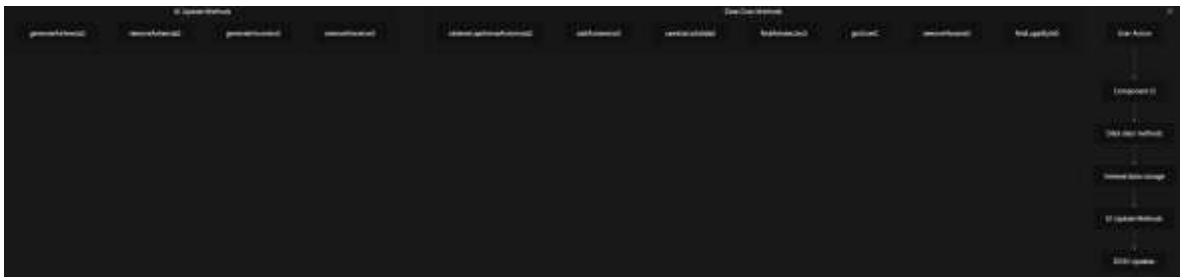
Fuentes: [componentes/add-asistencia/add-asistencia.js](#) 10 a 16

[componentes/asistencia/useAsistencia.utils.js](#) 3 [componentes/horarios/useHorarios.utils.js](#) 3

#### 4.4.Arquitectura de flujo de datos

El sistema aplica un patrón centralizado de gestión de datos en el que todos los componentes interactúan a través de la Dataclase.

##### 4.1.4 Patrón de acceso a los datos



Fuentes: [componentes/add-asistencia/add-asistencia.js](#) 17 a 46

[componentes/asistencia/useAsistencia.utils.js](#) 7 a 26

[componentes/horarios/useHorarios.utils.js](#) 5 a 11

#### 4.5.Interacciones básicas del sistema

La funcionalidad central del sistema gira en torno a la gestión de asistencia con patrones de interacción específicos entre componentes.

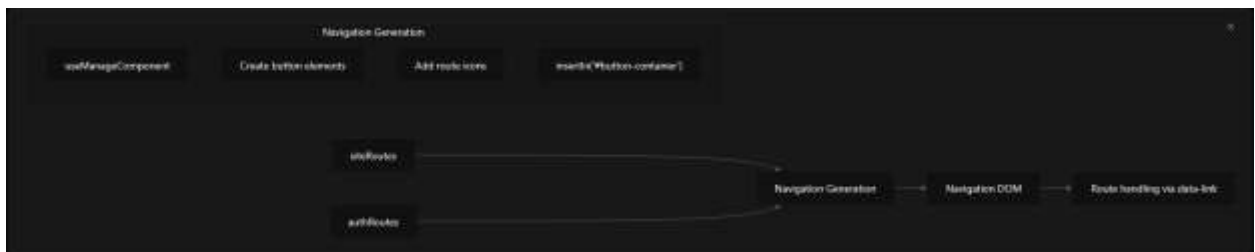


#### 4.1.5 Asistencia Gestión de asistencia Flujo



Fuentes: [componentes/add-asistencia/add-asistencia.js](#) 19 a 52

#### 4.6.Flujo del sistema de navegación



Fuentes: [componentes/nav/nav.js](#) 5 - 66

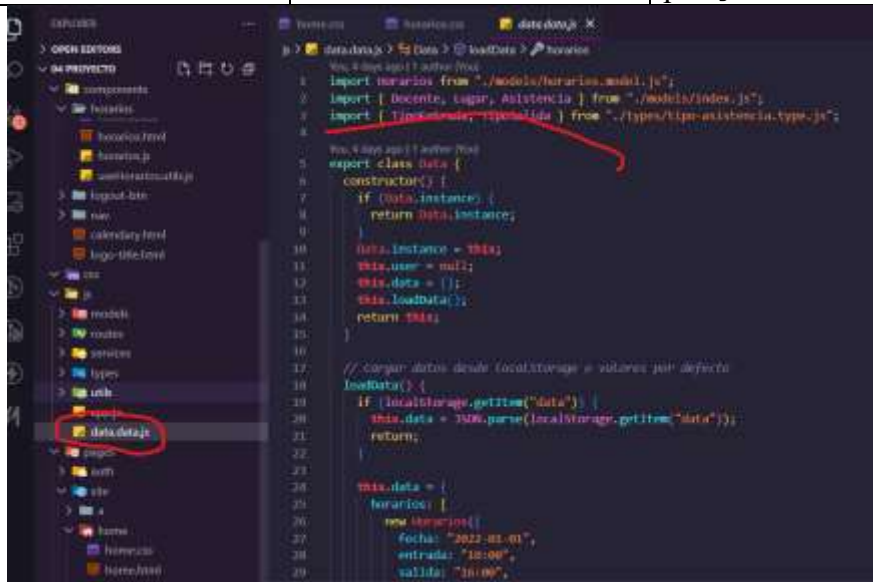
## 4.7. Principios arquitectónicos clave

### 4.1.7 Componente Independencia

Principio	Aplicación	Ejemplo
<b>Separación de las preocupaciones</b>	Cada componente tiene archivos HTML, CSS y JS distintos	add-asistencia.html, add-asistencia.css, add-asistencia.js
<b>Separación de utilidad</b>	Lógica de negocio dividida en useComponent.utils.js Archivos	useAsistencia.utils.js, useHorarios.utils.js
<b>Renderización basada en la plantilla</b>	Generación HTML a través de funciones de plantilla	templateAsistencia(), templateHorarios()

### 4.2.7 Centralización de la gestión de datos

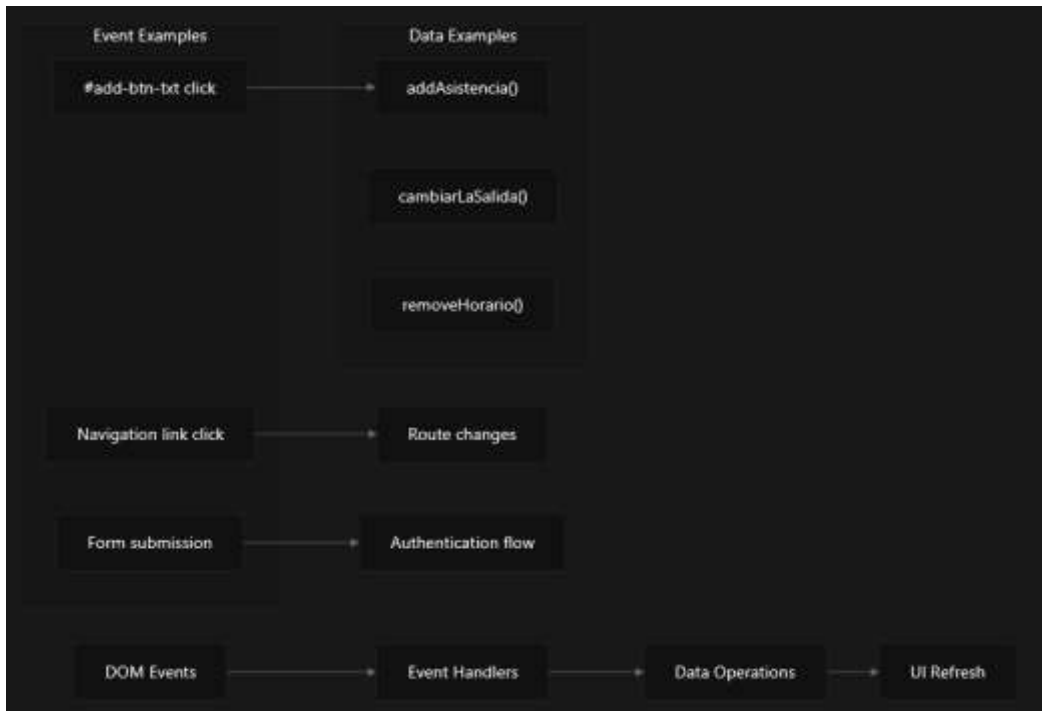
Patrón	Aplicación	Referencia del código
<b>Fuente de datos única</b>	Todos los componentes utilizan una sola Data instancia de clase	const data = new Data()
<b>Acceso de datos consistente</b>	Nombre de método estandarizado para operaciones de la CRUD	findAsistencias(), addAsistencia(), removeHorario()
<b>Actualizaciones de IU reactivas</b>	Regeneración de IU tras cambios de datos	removeAsistencia() seguidos por generateAsistencia()



La clase data es un singleton lo que permite conservar los datos en toda la aplicación

#### 4.8.Evento-Driven Arquitectura

El sistema utiliza manejadores de eventos DOM para desencadenar la lógica del negocio, manteniendo una clara separación entre las interacciones de los usuarios y las operaciones de datos.



Fuentes: [componentes/add-asistencia/add-asistencia.js](#) 19

[componentes/nav/nav.js](#) A 23

Esta arquitectura proporciona una separación sostenible de preocupaciones, al tiempo que permite un flujo de datos eficiente y un manejo de la interacción de los usuarios a lo largo del sistema de gestión de la asistencia al profesorado.

## 5. Gestión de asistencia

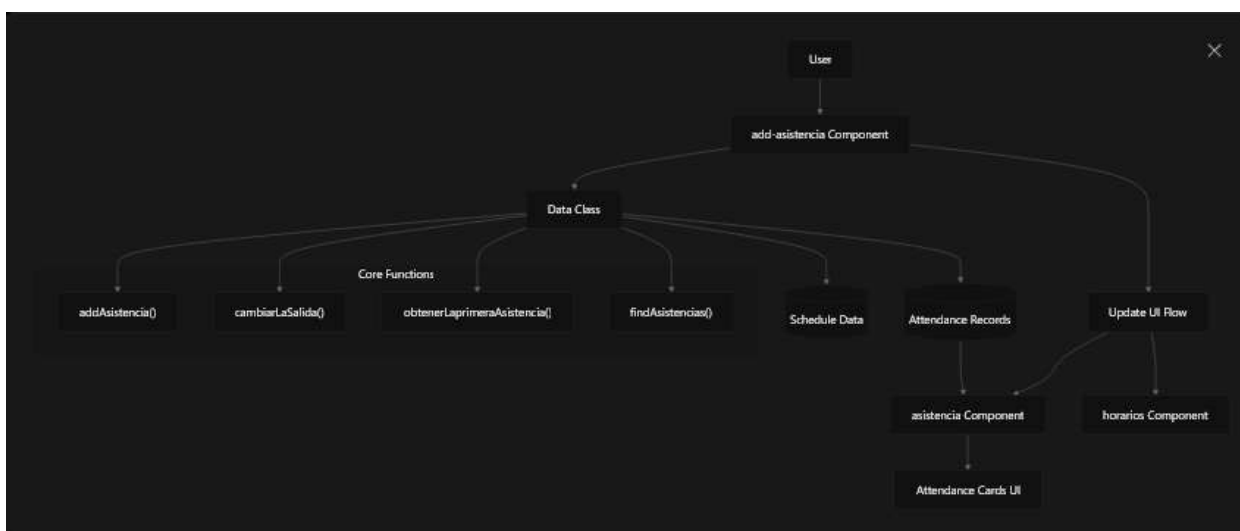
### 5.1. Finalidad y alcance

La funcionalidad de grabación y visualización de asistencia dentro del sistema de gestión de asistencia al profesorado. El subsistema de gestión de asistencia se encarga de la creación de nuevas entradas de asistencia, actualizando los tiempos de salida para las entradas existentes y mostrando registros de asistencia a los usuarios a través de una interfaz basada en tarjetas.

Para obtener información sobre la gestión de horarios y su relación con la asistencia, consulte Gestión de Listas. Para más detalles sobre la autenticación del usuario y la gestión de sesiones, consulte Sistema de autenticación.

### 5.2. Resumen del Asistencia

El subsistema de gestión de asistencia consta de dos componentes principales que trabajan juntos para proporcionar funcionalidad completa de seguimiento de asistencia:



```

    },
    asistencias: [
      new Asistencia({
        id: 0,
        idDocente: 0,
        idLugar: 0,
        entrada: {
          tipoEntrada: TipoEntrada.entrada,
          horaEntrada: "10:00",
        },
        salida: {
          tipoSalida: TipoSalida.salida,
          horaSalida: "10:00",
        },
        fecha: "2022-01-01",
      })
    ]
  });

```

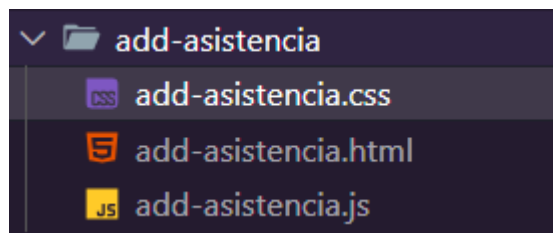
Fuentes: [componentes/add-asistencia/add-asistencia.js1-52](#)

[componentes/asistencia/useAsistencia.utils.js1 a 31](#) [componentes/asistencia/asistencia.js1-50](#)

### 5.3. Añadir registros de asistencia

#### 5.1.3 Añadir el componente de Asistencia

El add-asistencia componente proporciona la interfaz principal para registrar entradas y salidas de asistencia. El componente cuenta con un botón circular que se expande en flota para mostrar el texto de acción.



Archivo de componentes	Objeto
<p>add-asistencia.html</p> <pre> &lt;link rel="stylesheet" href="components/add-asistencia/add-asistencia.css"&gt;  &lt;div class="add-container-div"&gt;   &lt;header class="header-div"&gt;     &lt;div class="title"&gt;       &lt;span style="font-size: .9em;" class="title-main"&gt;Asistencias/&lt;/span&gt;       &lt;span style="font-size: .9em;" class="title-highlight"&gt; / Salidas/&lt;/span&gt;     &lt;/div&gt;   &lt;/header&gt;   &lt;button class="btn btn-add btn" id="add-btn"&gt;     &lt;div class="btn-icon"&gt;&lt;/div&gt;     &lt;div class="btn-txt"&gt;Agregar asistencia / Salidas/&lt;/div&gt;   &lt;/button&gt; &lt;/div&gt;  &lt;script type="module" src="components/add-asistencia/add-asistencia.js"&gt;&lt;/script&gt; </pre>	<p>Estructura de botones y diseño</p>

<p>add-asistencia.css</p> <pre> .add-container-div {   background-color: var(--color-light);   border: 1px solid #000;   border-radius: 20px;   padding: 2.5em 1.5em;   width: 100%;   display: flex;   flex-direction: column;   align-items: center;   justify-content: center;   box-shadow: 0 12px 24px #000;   transition: all 0.3s ease; } </pre>	<p>Estilo de botón animado con efectos flotantes</p>
<p>add-asistencia.js</p> <pre> \$("add-btn-tel").addEventListener("click", () =&gt; {   const asistencia = data.obtenerAsistencia();    if (     asistencia.entrada        asistencia.salida        asistencia.tipoSalida === 'modified'   ) {     data.cambiarSalida(asistencia.id, {       tipoSalida: obtenerNuevoTipoSalida(),       horaSalida: obtenerHoraActual(),     });   } } </pre>	<p>Lógica de grabación de asistencia básica</p>

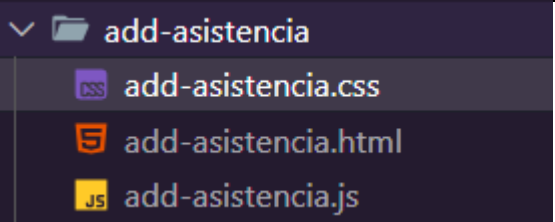
### 5.2.3 Asistencia Grabación Lógica Flujo



Fuentes: [componentes/add-asistencia/add-asistencia.js](#) 19 a 52

### 5.1.3 Funciones clave y operaciones de datos

El proceso de registro de asistencia utiliza varias funciones de utilidad y operaciones de datos:

Función	Objeto	Ubicación
<code>obtenerLaPrimeraAsistencia()</code>	Obtiene la primera entrada de asistencia incompleta	Método de la clase de datos
<code>cambiarLaSalida(id, salidaData)</code>	Actualiza el tiempo de salida para la entrada existente	Método de la clase de datos
<code>addAsistencia(attendanceData)</code>	Crea nuevo récord de asistencia	Método de la clase de datos
<code>removeHorario(index)</code>	Elimina la franja horaria tras la creación de asistencia	Método de la clase de datos
<code>obtenerRandonTipoEntrada()</code>	Genera tipo de entrada aleatorio	Función de utilidad
<code>obtenerRandonTipoSalida()</code>	Genera tipo de salida aleatorio	Función de utilidad
<code>obtenerHoraActual()</code>	Consigue el tiempo actual	Función de utilidad
		

Fuentes: [componentes/add-asistencia/add-asistencia.js1 a 8](#) [componentes/add-asistencia/add-asistencia.js20 a 1945](#)

## 5.4. Mostrando registros de asistencia

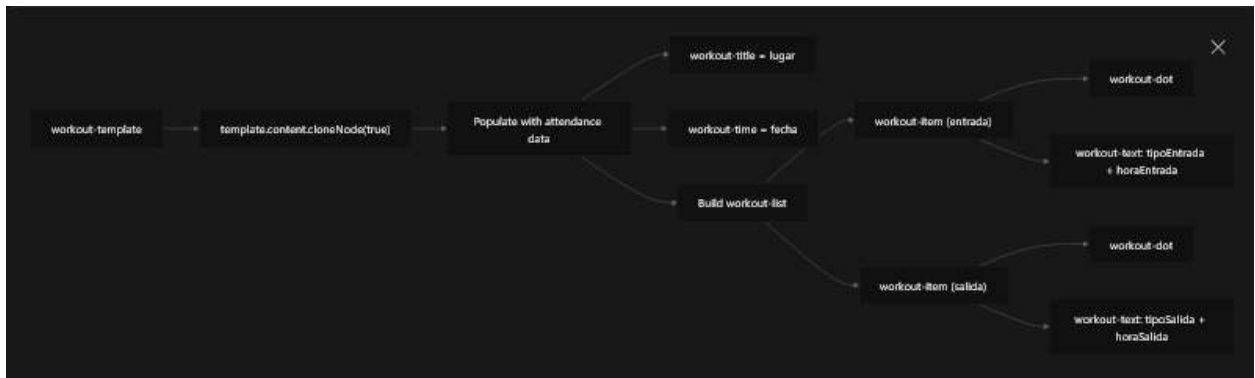
### 5.1.4 Componente de visualización

El `asistencia`El componente realiza registros de asistencia utilizando un diseño basado en tarjetas con un sistema de plantillas. Cada récord de asistencia se muestra como una tarjeta de estilo que muestra información de entrada y salida.



```
export const templateAsistencia = ({ id, lugar, fecha, salida, entrada }) => {  
  const template = $(".workout-template")[0];  
  console.log(template);  
  const clone = template.content.cloneNode(true);  
  
  clone.querySelector(".workout-title").textContent = lugar;  
  clone.querySelector(".workout-time").textContent = fecha;  
  const list = clone.querySelector(".workout-list");
```

```
✓ export const generateAsistencia = () => {  
  const asistencias = data.findAsistencias(user); // aquí estaba mal el nombre: findasistencias  
  asistencias.forEach((asistencia) => {  
    const lugar = data.findLugarById(asistencia.idLugar);  
    const card = templateAsistencia({  
      id: asistencia.id,  
      lugar: lugar.nombre + " (" + lugar.direccion + ")",  
      fecha: asistencia.fecha,  
      salida: {  
        tipoSalida: asistencia.salida.tipoSalida,
```



Fuentes: [componentes/asistencia/asistencia.html](#) 5 a 16 [componentes/asistencia/asistencia.js](#) 4 a 49

### 5.1.4 Estructura de plantilla y vinculación de datos

El `templateAsistencia` función crea tarjetas de asistencia mediante la clonación de la plantilla HTML y poblarla con datos:

Elemento de la plantilla	Fuente de datos	Descripción
--------------------------	-----------------	-------------



.workout-title	lugar.nombre + direccion	Nombre de la ubicación y dirección
.workout-time	asistencia.fecha	Fecha de asistencia
.workout-list	Elementos de entrada y salida	Contenedor para detalles de asistencia
.workout- item(entrada)	entrada.tipoEntrada + horaEntrada	Tipo de entrada y tiempo
.workout- item(salida)	salida.tipoSalida + horaSalida	Tipo y hora de salida

```

const template = $(".workout-template")[0];
console.log(template);
const clone = template.content.cloneNode(true);

clone.querySelector(".workout-title").textContent = lugar;
clone.querySelector(".workout-time").textContent = fecha;
const list = clone.querySelector(".workout-list");

// Crear div para entrada
const entradaDiv = document.createElement("div");
entradaDiv.className = "workout-item"; // mejor clase para contener punto + texto

const entradaDot = document.createElement("div");
entradaDot.className = "workout-dot";

const entradaText = document.createElement("div");
entradaText.className = "workout-text";
const textoEntrada = entrada.horaEntrada ? entrada.horaEntrada : "";
entradaText.textContent = entrada.tipoEntrada + ", " + textoEntrada; //(entrada.horaEntrada

entradaDiv.appendChild(entradaDot);
entradaDiv.appendChild(entradaText);

// Crear div para salida
const salidaDiv = document.createElement("div");
salidaDiv.className = "workout-item";

```

Fuentes: [componentes/asistencia/asistencia.js9 a 1945](#)

#### 5.1.4 Generación y actualización de servicios públicos

El useAsistencia.utils.js file proporciona funciones para la gestión del ciclo de vida de visualización de asistencia:



Fuentes: [componentes/asistencia/useAsistencia.utils.js7 a 30](#)

## 5.5. Flujo e integración de datos

### 5.1.5 Integración con Capa de Datos

El sistema de gestión de la asistencia se integra estrechamente con el `Dataclase` para todas las operaciones de persistencia:

Método de datos	Usage in Asistencia	Parámetros
<code>getUser()</code>	Obtenga el usuario actual para los registros de asistencia	Ninguno
<code>findAsistencias(user)</code>	Recupera los registros de asistencia del usuario	Objeto de usuario
<code>findLugarById(id)</code>	Obtenga detalles de ubicación para la visualización	Identificación de ubicación
<code>addAsistencia(data)</code>	Crear nueva entrada de asistencia	Objeto de asistencia

cambiarLaSalida(id, salida)	Actualizar la información de salida	ID, objeto de datos de salida
obtenerLaPrimeraAsistencia()	Comprobar las entradas incompletas	Ninguno
removeHorario(index)	Eliminar programar ranura después de la entrada	Indice de la lista

```

if (
  asistencia &&
  asistencia.entrada &&
  asistencia.salida.tipoSalida === undefined
) {
  data.cambiarLaSalida(asistencia.id, {
    tipoSalida: obtenerRadoMtipoSalida(),
    horaSalida: obtenerHoraActual(),
  });
} else {
  if (data.data.horarios.length === 0) {
    alert("No hay horarios disponibles");
    return;
  }
  data.addAsistencia({
    idDocente: data.getUser().id,
    idLugar: 0,
    entrada: {
      tipoEntrada: obtenerRadoMtipoEntrada(),
      horaEntrada: obtenerHoraActual(),
    },
    fecha: obtenerDiaMesAñoActual(),
  });
}

```

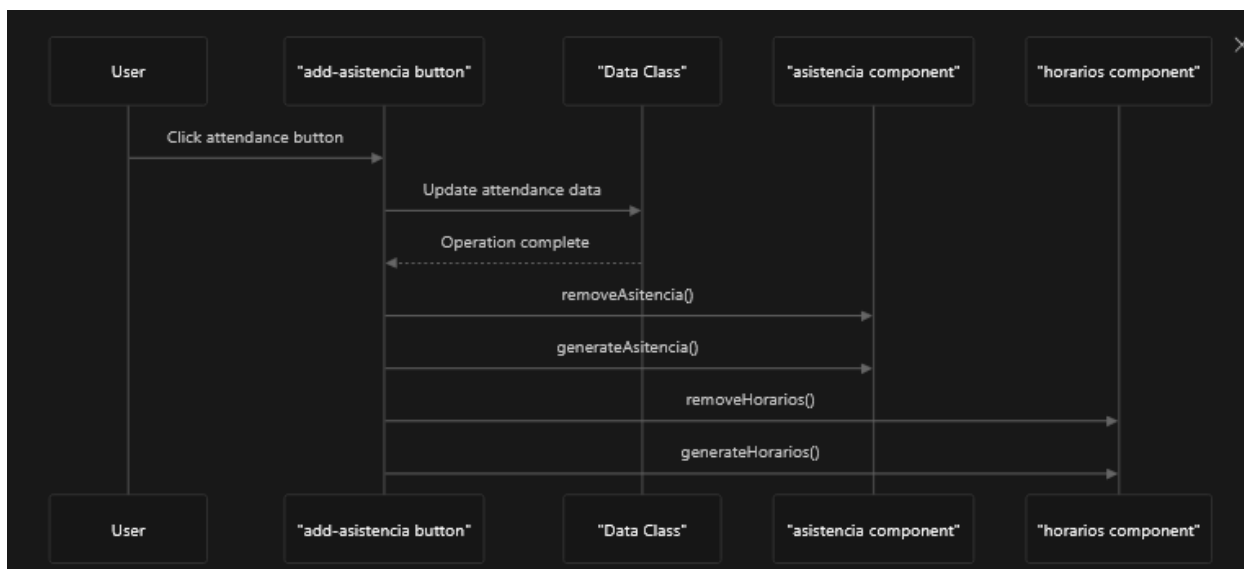
Fuentes:

[componentes/add-asistencia/add-asistencia.js](#) 17 a 46

[componentes/asistencia/useAsistencia.utils.js](#) 4 a 25

### 5.1.5 Coordinación de Refresca de IU

Las operaciones de asistencia desencadenan actualizaciones coordinadas de la interfaz de usuario en múltiples componentes:



Fuentes: [componentes/add-asistencia/add-asistencia.js48 - 51](#)

## 5.6.Componentes de IU y Estilo

### 5.1.6 Pantalla con tarjeta

La pantalla de asistencia utiliza un diseño basado en tarjetas con las siguientes clases de CSS:

Clase CSS	Objeto
.workout-card	Contensión de tarjeta principal con tamaño sensible
.workout-bg	Semi-transparente superposición de fondo
.workout-title	Mostrar nombre de ubicación
.workout-time	Fecha de visualización posicionada de arriba-derecha
.workout-icon-wrapper	Contenedor de icono con pantalla de emoji
.workout-list	Contenedor para artículos de entrada/salida
.workout-item	Registro individual de entrada o salida
.workout-dot	Indicador circular pequeño
.workout-text	Contenido de texto para los detalles de entrada/salida

```
.workout-card {
  width: 100%;
  max-width: 465px;
  height: auto;
  min-height: 150px;
  position: relative;
  overflow: hidden;
  font-family: -apple-system, BlinkMacSystemFont, "Segoe UI", system-ui,
  sans-serif;
  padding: 1.5rem;
  box-sizing: border-box;
  border-radius: 20px;
  transition: all 0.3s cubic-bezier(0.4, 0, 0.2, 1);
}

.workout-card:hover {
  transform: translateY(-4px);
}

.workout-bg {
  position: absolute;
  inset: 0;
  background: linear-gradient(
```

Fuentes:

[componentes/asistencia/asistencia.css1-144](#)

### 5.2.6 Añadir botón interactivo

El botón de asistencia de añadir características de efectos de flotante animados:

Clase CSS	Efecto
<code>.icon-btn</code>	Base de estilo de botones circulares
<code>.add-btn:hover</code>	Amplía ancho para mostrar texto
<code>.btn-txt</code>	Texto oculto que aparece en flotante
<code>.add-icon::before, .add-icon::after</code>	Crea icono de "o" con CSS

```
.icon-btn {
  position: relative;
  width: 60px;
  height: 60px;
  border: 1px solid rgba(0, 0, 0, 0.88);
  background: #fff;
  border-radius: 50%;
  font-family: inherit;
  font-weight: 500;
  transition: all 0.3s cubic-bezier(0.175, 0.885, 0.32, 1.275);
  overflow: hidden;
  cursor: pointer;
  box-shadow: 0 4px 12px rgba(0, 0, 0, 0.88);
}

.icon-btn:focus {
  outline: none;
  box-shadow: 0 0 4px rgba(42, 92, 154, 0.3);
}

.add-btn:hover {
  width: 150px;
  border-radius: 30px;
  background-color: var(--color-light);
}
```

Fuentes:

[componentes/add-asistencia/add-asistencia.css14 - 111](#)

## 6. Gestión de los Horarios

Este documento cubre la visualización de horarios y la funcionalidad de gestión dentro del sistema de asistencia al profesorado. El componente de gestión del horario es responsable de renderizar las tarjetas de horario del maestro y manejar la eliminación dinámica de las franjas horarias de horarios cuando se registra la asistencia.

Para obtener información sobre cómo interactúan los horarios con la grabación de asistencia, consulte Gestión de asistencia. Para más detalles sobre la navegación general del sistema, consulte Sistema de navegación.

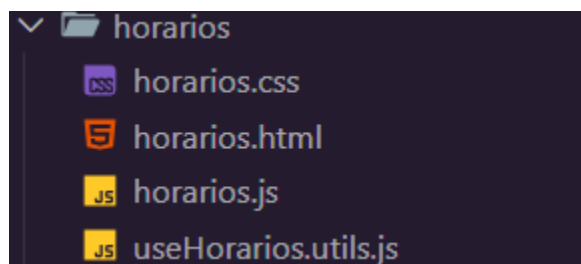
### 6.1.Finalidad del sistema y alcance

El sistema de gestión de horarios muestra los horarios de trabajo del profesorado como tarjetas visuales que muestran la fecha, la hora de entrada y la información del tiempo de salida. El sistema se integra con el seguimiento de asistencia eliminando automáticamente las

frangas horarias del horario cuando los profesores completan su asistencia para ese período de tiempo.

## 6.2.Arquitectura de componentes

El sistema de gestión de horarios sigue un patrón modular de componentes con archivos separados para estructura, estilo y comportamiento:



**Fuentes:** [componentes/horarios/horarios.html](#) 1 a 19 [componentes/horarios/horarios.js](#) 1 a 10 [componentes/horarios/useHorarios.utils.js](#) 1 a 16

## 6.3.Estructura de datos y plantilla de refuerzo

### 6.1.3 Cuadro de datos de calendario

Cada lista de entrada (`horario`) contiene las siguientes propiedades:

Bienes	Tipo	Descripción
fecha	Cadena	Fecha de la sesión prevista

entrada	Cadena	Tiempo de entrada/inicio
salida	Cadena	Tiempo de salida/fin

```
export default class Horarios {
  constructor({fecha, salida, entrada}) {
    this.fecha = fecha;
    this.salida = salida;
    this.entrada = entrada;
  }
}
```

### 6.2.3 Implementación de la función de plantilla

El `templateHorarios` función crea tarjetas de horario a partir de datos de plantilla:



Fuentes: [componentes/horarios/horarios.js1-9](#) [componentes/horarios/horarios.html2 a 12](#)

## 6.4. Generación y gestión de calendarios

### 6.1.4 Funciones básicas

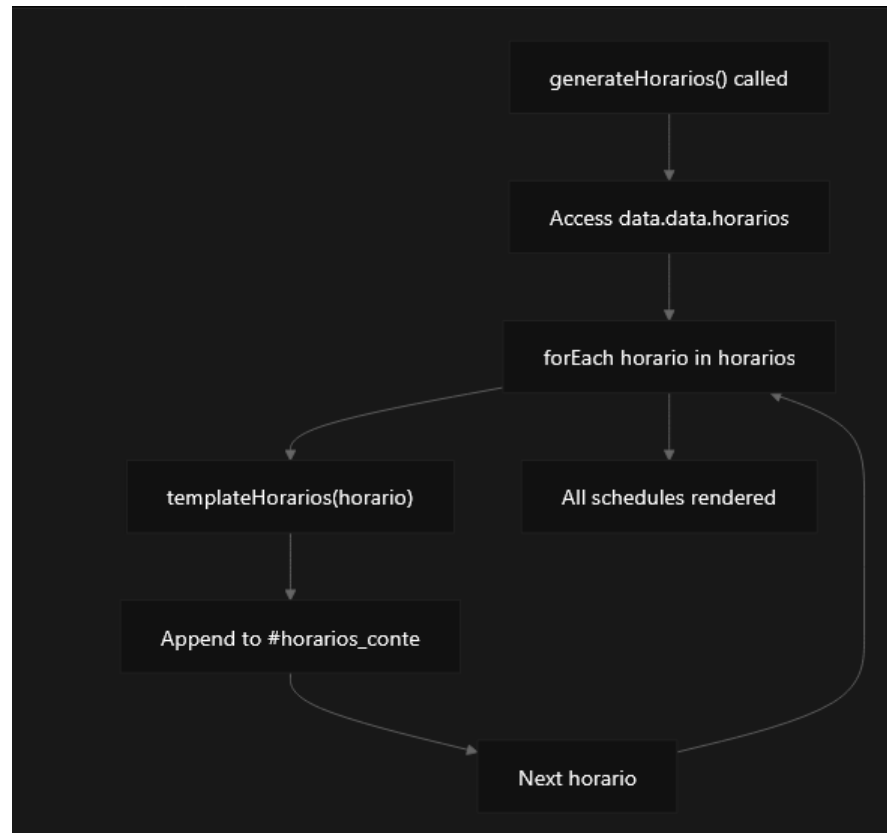
El sistema de gestión de horarios se desempeña dos funciones principales:

`generarHorarios ()`

```
export const generateHorarios = () => {
  const horarios = data.data.horarios;
  horarios.forEach((horario, index) => {
    const card = templateHorarios(horario);
    $("#horarios_conte").appendChild(card);
  });
};
```

Renderiza todas las entradas de horarios de la capa de datos en el contenedor DOM:





### `removeHorarios()`

```
export const removeHorarios = () => {  
  $("#horarios_conte").innerHTML = "";  
};
```

Borra todas las tarjetas de horario del contenedor de visualización mediante la configuración `innerHTML` a la cuerda vacía.

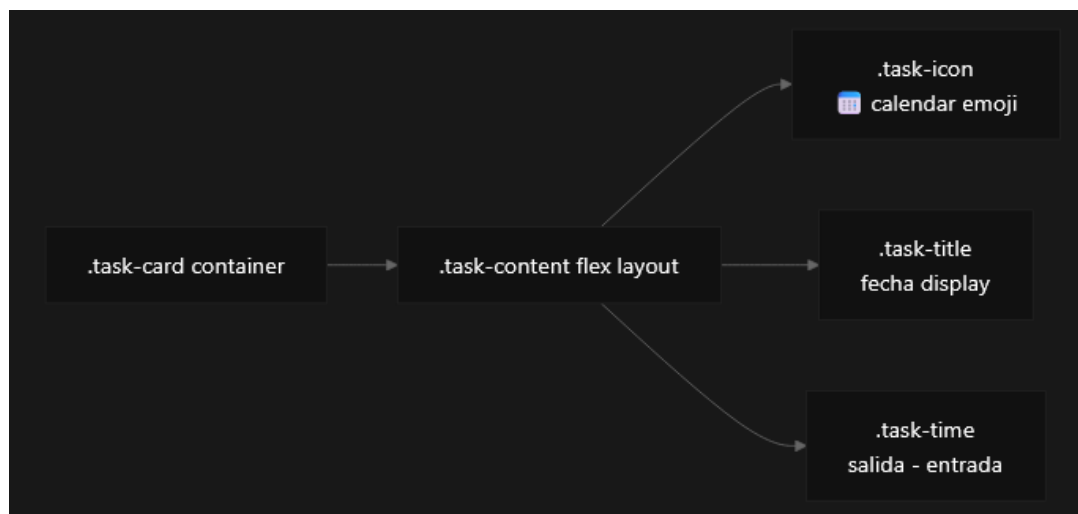
### Fuentes:

[componentes/horarios/useHorarios.utils.js5 a 15](#)

## 6.5. Diseño visual y diseño

### 6.1.5 Estructura de la tarjeta de calendario

Cada tarjeta de horario sigue un diseño visual consistente:



```

<template id="horarios-template">
  <article class="task-card">
    <div class="task-content">
      <div class="task-icon">
        <span class="task-emoji" aria-hidden="true">📅</span>
      </div>
      <h2 class="task-title"></h2>
      <time class="task-time"></time>
    </div>
  </article>
</template>
  
```

## 6.6. Especificaciones de Estilo CSS

Elemento	Dimensiones	Antecedentes	Propiedades clave
.task-card	465px . 75px	F8d57e	borde-radio: 20px, diseño de flexbox
.task-icon	35px 35px	"ffffff"	Sector fronterizo: 10px, emoji centrado
.task-title	flexible	transparente	Fuente de Gilroy-Bold, color n.b2442
.task-time	Alineado por la derecha	transparente	Gilroy-Medium font, tamaño 14px



```
.task-card {  
  width: 485px;  
  height: 75px;  
  overflow: hidden;  
  background: linear-gradient(135deg, var(--color-light) 0%, #ffffff 100%);  
  border-radius: 16px;  
  display: flex;  
  align-items: center;  
  padding: 0 26px;  
  box-shadow: 0 4px 20px #00000004, 0 1px 4px #00000004;  
  border: 1px solid #cccccc04;  
  transition: all 0.3s cubic-bezier(0.4, 0, 0.2, 1);  
  position: relative;  
}
```

## 6.7. Conducta sensible

El sistema incluye un estilo móvil que responde al móvil para pantallas de ancho de 480px:

- Ancho de la tarjeta se ajusta al 100%
- Reducido el acolchado (15px vs 31px)
- márgenes de icono más pequeños (10px vs 17px)
- Tamaños de fuente reducidos para el título (14px) y tiempo (12px)

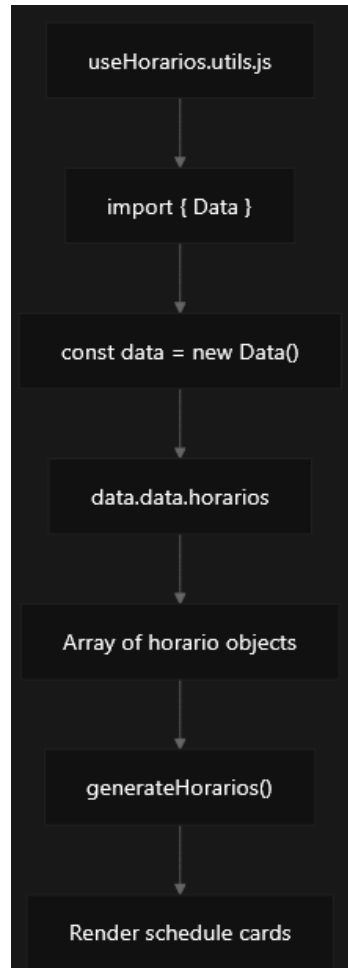
**Fuentes:**

[componentes/horarios/horarios.css77-144](#)

## 6.8. Integración con Capa de Datos

### 6.1.8 Patrón de acceso a los datos

El sistema de gestión de horarios se integra con la capa de datos centralizada:



Fuentes: [componentes/horarios/useHorarios.utils.js1 a 4](#)

## 6.9.DOM Integración y Gestión de contenedores

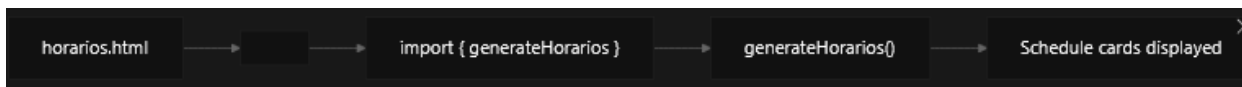
### 6.1.9 Elementos de contenedores

Tarjetas de horario se entregan en el `#horarios_conteelemento` contenedor, que utiliza:

- Clases: `flex-container-grid`
- Estilo: `margin: 1em 0`
- Finalidad: contenedor de diseño de cuadrícula para tarjetas de horario

### 6.2.9 Cargando del módulo

El componente de horario utiliza las importaciones de módulos ES6 y se inicializa a través de la plantilla HTML:



**Fuentes:** [componentes/horarios/horarios.html](#) 14 a 19

### 6.10. Relación con el sistema de asistencia

El sistema de gestión de los horarios está estrechamente integrado con el seguimiento de la asistencia. Cuando los profesores registran asistencia, se eliminan las franjas horarias de horario completadas de la pantalla para evitar entradas duplicadas y mantener la consistencia de los datos. Esta integración se gestiona a través de la capa de datos compartida y funciones de utilidad coordinadas.

## 7. Sistema de autenticación

### 7.1. Finalidad y alcance

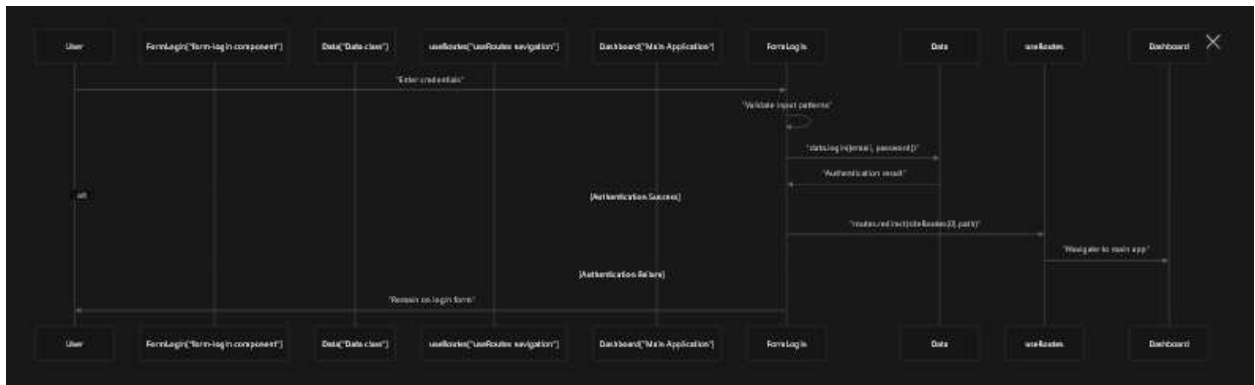
Este documento cubre el sistema de autenticación que gestiona la funcionalidad de inicio de sesión de usuario y logout en la aplicación de gestión de asistencia al profesor. El sistema proporciona un control seguro del acceso mediante autenticación basada en la forma y la gestión de sesiones. Para obtener información sobre la navegación entre zonas autenticadas y no autenticadas, consulte [Sistema de navegación](#).

El sistema de autenticación consta de dos componentes primarios: el formulario de inicio de sesión para la autenticación del usuario y el botón de inicio de sesión para la terminación de

sesión. Ambos componentes se integran con la capa central de datos y el sistema de enrutamiento para proporcionar experiencia de usuario sin problemas.

## 7.2. Autación Flujo

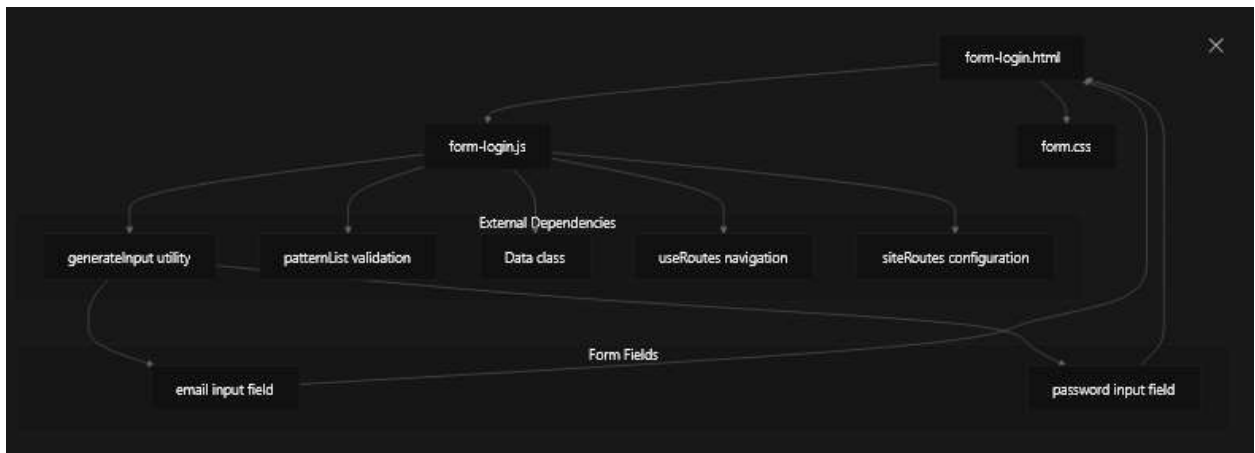
El sistema de autenticación sigue un flujo de aplicación web estándar en el que los usuarios deben autenticarse antes de acceder a las principales características de aplicación.



**Fuentes:** [componentes/form-login/form-login.js](#)41 a 56 [componentes/logout-btn/logout-btn.js](#)10 a 21

### 7.3.Login de arquitectura de componentes

El componente de forma de inicio de sesión implementa un enfoque modular con preocupaciones separadas para la presentación, validación y lógica de negocio.



```
login.html X
pages > auth > login > login.html > script
You, 4 days ago | 1 author (You)
1 <div data-component="./components/form-login/form-login"></div>
2
3 <script type="module" src="pages/auth/login/login.js"></script>
```

**Fuentes:** [componentes/form-login/form-login.html](#) 1 a 16 [componentes/form-login/form-login.js](#) 1-57

#### 7.4.Forma la generación de campo

La forma de inicio de sesión genera dinámicamente campos de entrada utilizando el `generateInput` función de utilidad con patrones de validación específicos:

Campo	Tipo	Patrón de validación	Aferradero de la Coloc
Correo electrónico	email	patternList.email	"Correo"
Contraseña	password	patternList.numberAndLetters	"Contraseña"

```
const email = generateInput({
  className: "input-field",
  attributes: {
    type: "email",
    placeholder: "Correo",
  },
  expression: patternList.email,
  labelText: "Correo",
  insertIn: "#form-login",
  labelText: ""
});

const password = generateInput({
  className: "input-field",
  attributes: {
    type: "password",
    placeholder: "Contraseña",
  },
  expression: patternList.numberAndLetters,
  labelText: "Contraseña",
  insertIn: "#form-login",
  labelText: ""
});
```

Ambos campos están configurados con la misma clase CSS `input-field` y se insertan en el `#form-login` contenedor. El formulario implementa validación del lado del cliente mediante la coincidencia de patrones antes de la presentación.

**Fuentes:** [componentes/form-login/form-login.js10 a 38](#)

### 7.5. Proceso de autenticación

El proceso de inicio de sesión sigue estos pasos:

1. **Formulario de envío Manejo** : El evento de envío se impide el comportamiento predeterminado del navegador
2. **Validación** : Cada campo de entrada llama `showError()` para mostrar los mensajes de validación
3. **Auténtico Intento** : El `Data.login()` método se llama con valores de correo electrónico y contraseña
4. **Redirección en Success**: Si la autenticación tiene éxito, el usuario es redirigido a la primera ruta del sitio

**Fuentes:** [componentes/form-login/form-login.js41 a 56](#)

### 7.6. Arquitectura de componentes de sesión

El componente de logout proporciona un botón dinámico que cambia el comportamiento basado en el estado de autenticación del usuario.





```

const listInputs = {email, password};
$("#form-login").addEventListener("submit", (e) => {
  e.preventDefault();

  listInputs.forEach(element => {
    element.showError();
  });

  const login = data.login({
    email: email.getValue(),
    password: password.getValue(),
  });

  if (login) {
    location.reload();
  }
});

```

**Fuentes:** [componentes/logout-btn/logout-btn.html](#) 1 a 6 [componentes/logout-btn/logout-btn.js](#) 1 a 22

## 7.7.Gestión estatal de Button

El botón de inicio de sesión implementa la gestión de estado inteligente:

- **Estado autentnticoted:** Muestra texto "Logout" y realiza la operación de logout cuando se hace clic

- **Estado noautizado** : Muestra texto "Entrar" texto y redirecciones a la ruta de autenticación cuando se hace clic

El Estado se determina llamando `data.getUser()` que devuelve la información actual de la sesión de usuario.

```
const routes = new useRoutes();
const data = new Data();
const logout = $("#logout");
if (data.getUser()) {
  logout.textContent = "Logout";
  logout.addEventListener("click", () => {
    data.logout();
    routes.redirect(authRoutes[0].path);
  });
} else {
  logout.textContent = "Login";
  logout.addEventListener("click", () => {
    routes.redirect(authRoutes[0].path);
  });
}
```

**Fuentes:** [componentes/logout-btn/logout-btn.js](#) 10 a 21

## 7.8.Integración con Capa de Datos

Ambos componentes de autenticación se basan en la `Data` clase para operaciones de autenticación básica:

Método	Objeto	Se utiliza por
<code>login({email, password})</code>	Autenticar credenciales de usuario	componente de la forma-login
<code>logout()</code>	Terminar sesión de usuario	componente de logout-btn
<code>getUser()</code>	Comprobar el estado actual de autenticación	componente de logout-btn

El `Data` clase sirve como la única fuente de la verdad para el estado de autenticación y las operaciones a lo largo de la aplicación.

**Fuentes:** [componentes/form-login/form-login.js48 - 51](#) [componentes/logout-btn/logout-btn.js10 a 14](#)

## 7.9.Integración de lanzamiento

El sistema de autenticación se integra con el sistema de enrutamiento de la aplicación a través de dos configuraciones de ruta:

- **authRoutes** : Rutas para usuarios no autenticados (página de blogs)
- **sitioRutas** : Rutas para usuarios autenticados (aplicación principal)

La navegación entre las zonas autenticadas y no autenticadas es manejada por la useRoutesutilidad, que proporciona la redirect()método de navegación programática.

**Fuentes:** [componentes/form-login/form-login.js54](#) [componentes/logout-btn/logout-btn.js14 a 19](#)

## 8. Perfil del profesor

### 8.1.Finalidad y alcance

El componente de Perfil Docente proporciona una visualización de información del profesorado autenticada dentro del sistema de gestión de asistencia. Este componente proporciona los detalles básicos del profesor, incluyendo nombre, correo electrónico e imagen de perfil en un diseño basado en tarjetas.

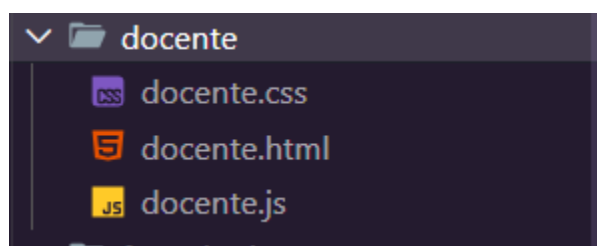
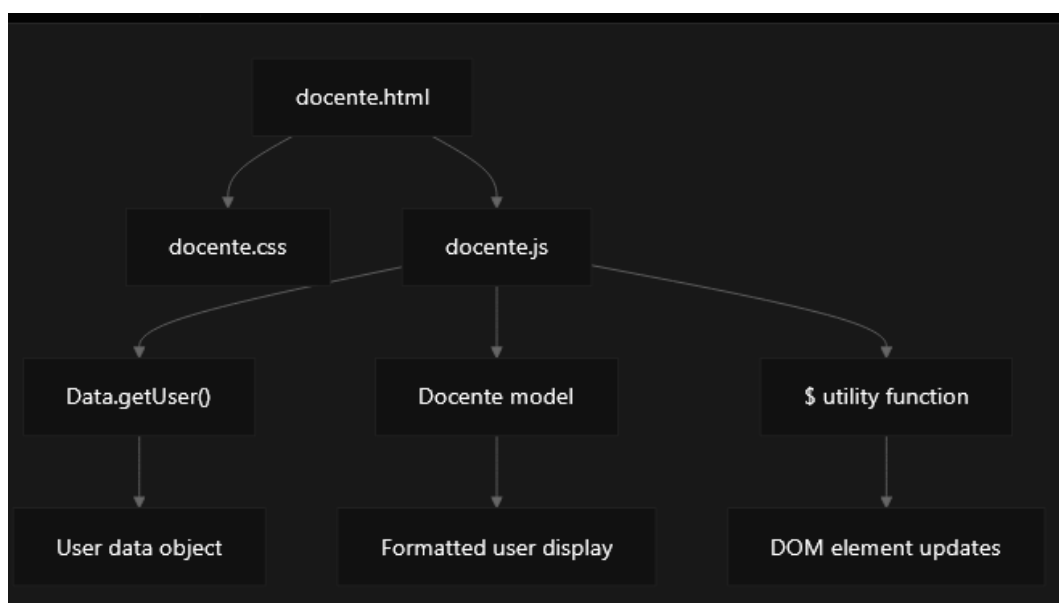
Para la autenticación y la gestión de las sesiones de los usuarios, consulte Sistema de autenticación. Para la navegación global del sistema, consulte Sistema de navegación.

## 8.2. Reseña del componente

El docente componente es un elemento de interfaz de usuario autónomo que muestra información de perfil del profesor. Sigue el patrón de arquitectura de componentes estándar con archivos separados HTML, CSS y JavaScript.

Archivo de componentes	Objeto
docente.html	Define la estructura y el diseño HTML
docente.css	Proporciona estilo y apariencia visual
docente.js	Maneja la recuperación de datos y la población de DOM

## 8.3. Estructura de componentes

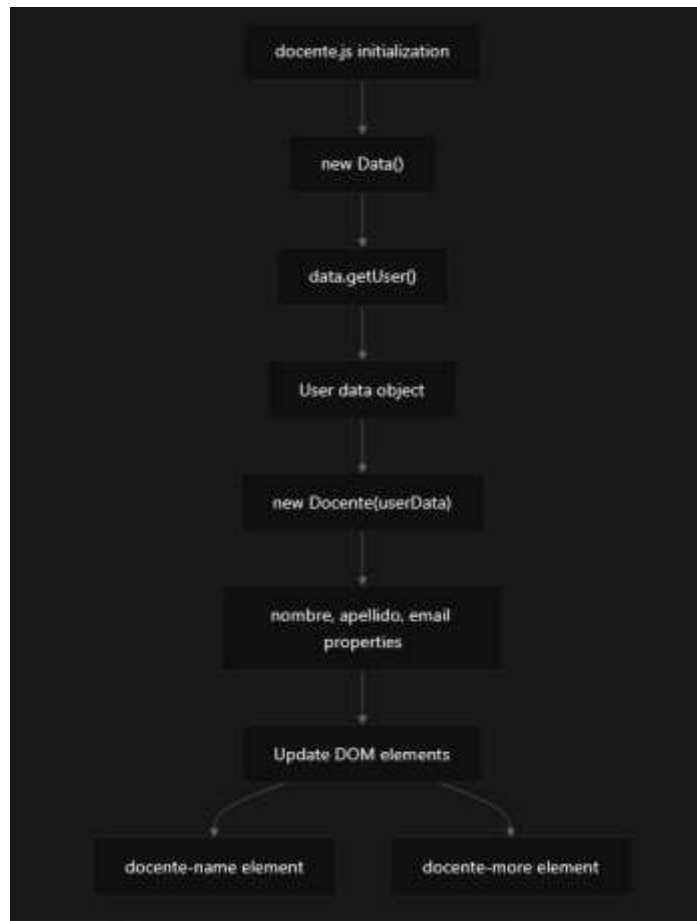


Fuentes: [componentes/docente/docente.html1-9](#) [componentes/docente/docente.css1-49](#)  
[componentes/docente/docente.js1 a 10](#)

#### 8.4. Flujo e integración de datos

El componente se integra con el sistema más amplio a través de la `Data` clase y `Docente` modelo para recuperar y en formato de la información del usuario.

#### 8.5. Proceso de recuperación de datos



El componente utiliza el siguiente patrón de flujo de datos:

1. Crea un `Data` instance para acceder a la información del usuario  
[componentes/docente/docente.js5](#)
2. Recupera los datos actuales de los usuarios a través de `data.getUser()`  
[componentes/docente/docente.js6](#)

3. Envuelve los datos del usuario en un Docente de modelo de instancia

[componentes/docente/docente.js6](#)

4. Populate elementos DOM usando el \$ función de utilidad

[componentes/docente/docente.js9 a 10](#)

Fuentes: [componentes/docente/docente.js5 a 10](#)

## 9. Sistema de navegación

### 9.1.Finalidad y alcance

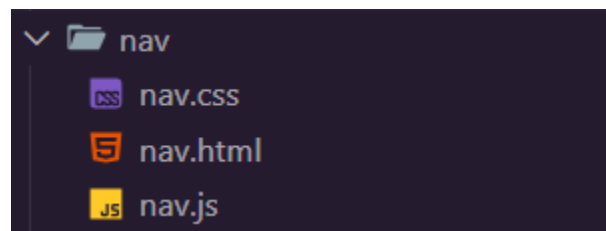
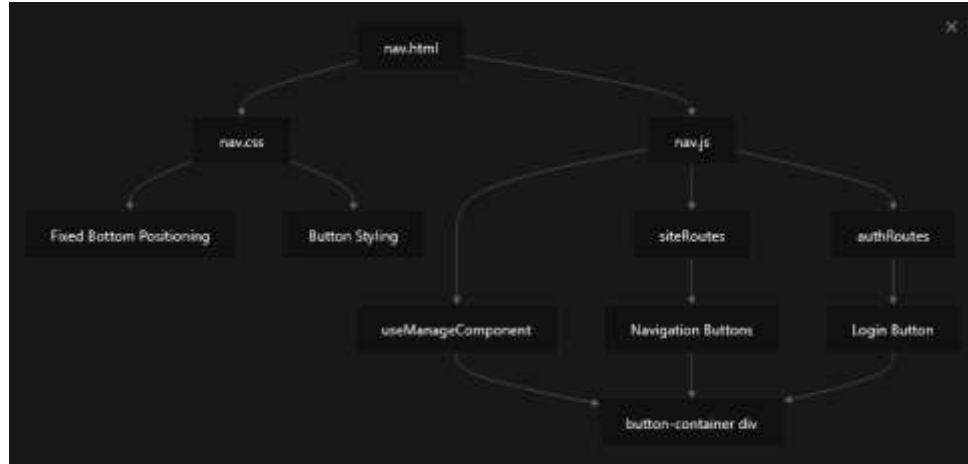
El Sistema de Navegación proporciona la interfaz de navegación primaria para la aplicación de gestión de la asistencia al profesorado. Crea una barra de navegación de fondo fija que permite a los usuarios moverse entre diferentes secciones de la aplicación. El sistema genera dinámicamente botones de navegación basados en rutas configuradas e se integra con el mecanismo de enrutamiento basado en hash de la aplicación.

Para obtener información sobre cómo se definen y gestionan las rutas, consulte [Arquitectura del Sistema](#). Para más detalles sobre las áreas funcionales individuales accesibles a través de la navegación, consulte [Componentes Core](#).

### 9.2.Estructura de componentes de navegación

El sistema de navegación se implementa como un componente modular siguiendo el patrón de componentes estándar utilizado en toda la aplicación. El componente consta de tres archivos principales que trabajan juntos para crear la interfaz de navegación.

### 9.1.2 Arquitectura de componentes



Fuentes: [componentes/nav/nav.html1 a 8](#) [componentes/nav/nav.js1-67](#)  
[componentes/nav/nav.css1-39](#)

El componente de navegación utiliza el `useManageComponent` utilidad para crear elementos DOM programáticamente. El contenedor de navegación principal se crea con el nombre de clase `button-container` y tipo de elemento `nav`, colocado en la parte inferior de la pantalla.

### 9.3.Integración de rutas y generación de botones

El sistema de navegación genera botones mediante el procesamiento de dos colecciones de rutas: `siteRoutes` y `authRoutes`. Cada objeto de ruta contiene metadatos utilizados para crear el botón de navegación correspondiente.

#### 9.4. Flujo de procesamiento de rutas



```

const nav = new useManageComponent({
  className: "button-container",
  element: "nav",
  inner: "holaaa",
  attributes: {
    id: "nav",
  },
});

const list = [];
siteRoutes.forEach((route) => {
  const a = new useManageComponent({
    className: "button",
    element: "a",
    inner: route.label,
    attributes: {
      id: route.label + "_link",
      href: "#" + route.path,
      "data-link": true,
    },
  });
});
  
```

Fuentes: [componentes/nav/nav.js14 - 37](#)

[componentes/nav/nav.js41 a 66](#)

Por cada ruta en siteRoutes, el sistema crea un elemento de botón con la siguiente estructura:

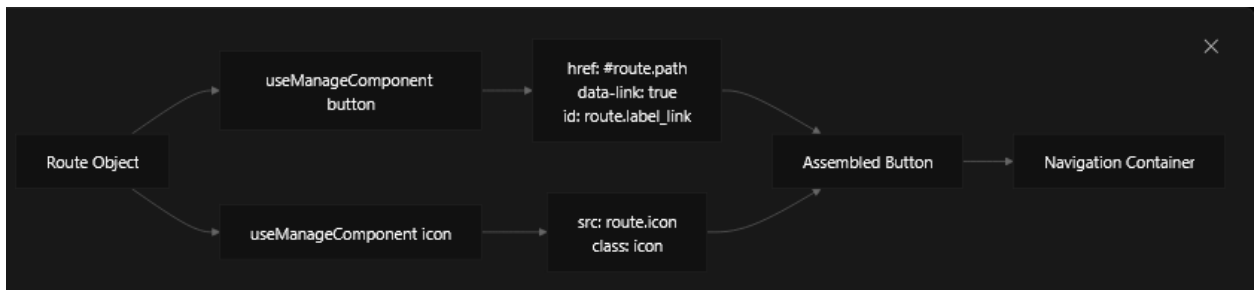


- Un elemento ancla con clase button
- Un ID único basado en la etiqueta de la ruta
- An hrefatributo fijado a #- ruta
- A data-linkatributo para la integración del router
- Un elemento de imagen icono con el icono especificado de la ruta

### 9.5.Aplicación del botón de navegación

Cada botón de navegación está construido con el useManageComponentpatrón, creando elementos DOM consistentes con atributos adecuados para la integración de enrutamiento.

### 9.6.Proceso de creación de botones



Fuentes:

[componentes/nav/nav.js16 a 36](#)

El proceso de creación de botones implica:

1. Creación de un elemento de anclaje con clase button [componentes/nav/nav.js16 a 25](#)
2. Añadir atributos incluyendo href y enlace de datos para enrutamiento  
[componentes/nav/nav.js20 a 24](#)
3. Creación de un elemento de imagen icono [componentes/nav/nav.js27 a 34](#)

4. Adjuntar el icono al botón usando addInner() [componentes/nav/nav.js35](#)
5. Añadir el botón completado a la lista de botones [componentes/nav/nav.js36](#)

## 9.7. Diseño visual y posicionamiento

El sistema de navegación utiliza un diseño de barra inferior fija con estilo específico para la accesibilidad y el atractivo visual.

### 9.1.7 Especificaciones de diseño

Bienes	Valor	Objeto
Posición	fixed	Estancias visibles durante el desplazamiento
Abajo	10px	Situado por encima de la pantalla inferior
Mímbre	250px	Ancho de contenedor fijo
Altura	40px	Altura de botón consistente
Radio fronterizo	10px	Esquinas de contenedores redondeadas
Antecedentes	var(--color-primary)	Utiliza el color principal tema

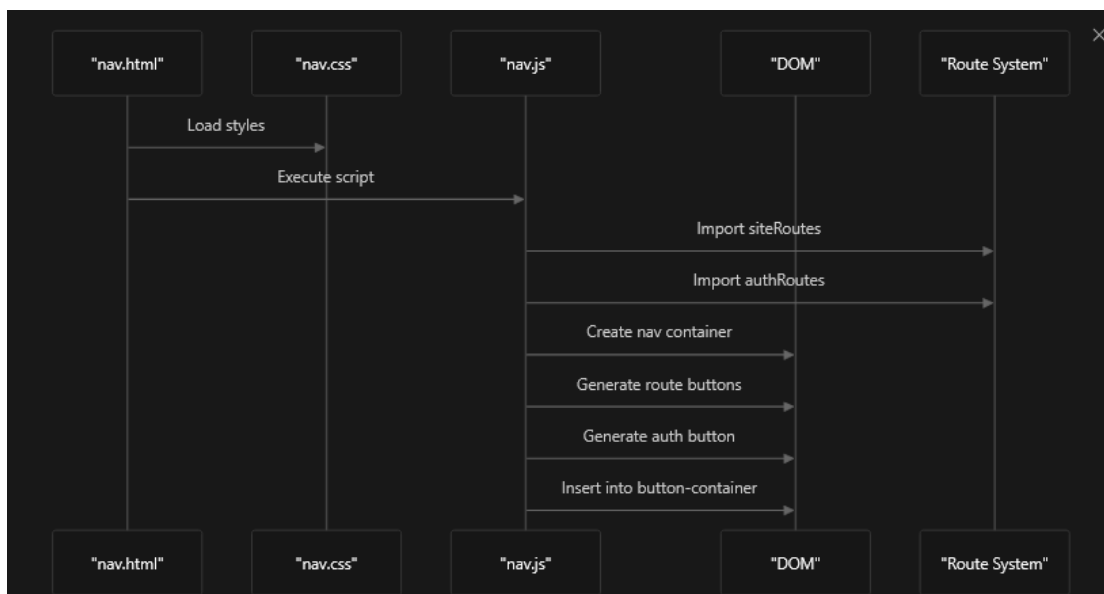
Fuentes: [componentes/nav/nav.css1 a 16](#)

El contenedor de botón se centra horizontalmente usando left: 50% y transform: translateX(-50%) [componentes/nav/nav.css4 a 5](#) Los botones individuales están diseñados como elementos circulares con efectos flotantes que los traducen hacia arriba por 3 píxeles [componentes/nav/nav.css32 a 34](#)

## 9.8. Integración e Inicialización de Componentes

El componente de navegación se integra con el sistema de enrutamiento de la aplicación a través de enlaces de navegación basados en hash y data-links sistema de atributos.

### 9.1.8 Secuencia de la inicialización



Fuentes: [componentes/nav/nav.html1](#) [componentes/nav/nav.html8](#) [componentes/nav/nav.js2 a 3](#)

El componente de navegación se inicializa cuando el archivo HTML carga el módulo JavaScript [componentes/nav/nav.html8](#) El script importa las configuraciones de enrutamiento necesarias [componentes/nav/nav.js2 a 3](#) y crea la estructura de navegación programáticamente, finalmente insertándola en el contenedor designado [componentes/nav/nav.js66](#)

El data-linkEl atributo en cada botón de navegación permite al sistema de enrutamiento de la aplicación interceptar clics y manejar la navegación sin recargas de página completas, soportando la arquitectura de la aplicación de una sola página.

## 10. Componentes de la interfaz de usuario

Este documento cubre los componentes de IU reutilizables y los elementos visuales utilizados en todo el sistema de gestión de la asistencia al profesorado. Los componentes de

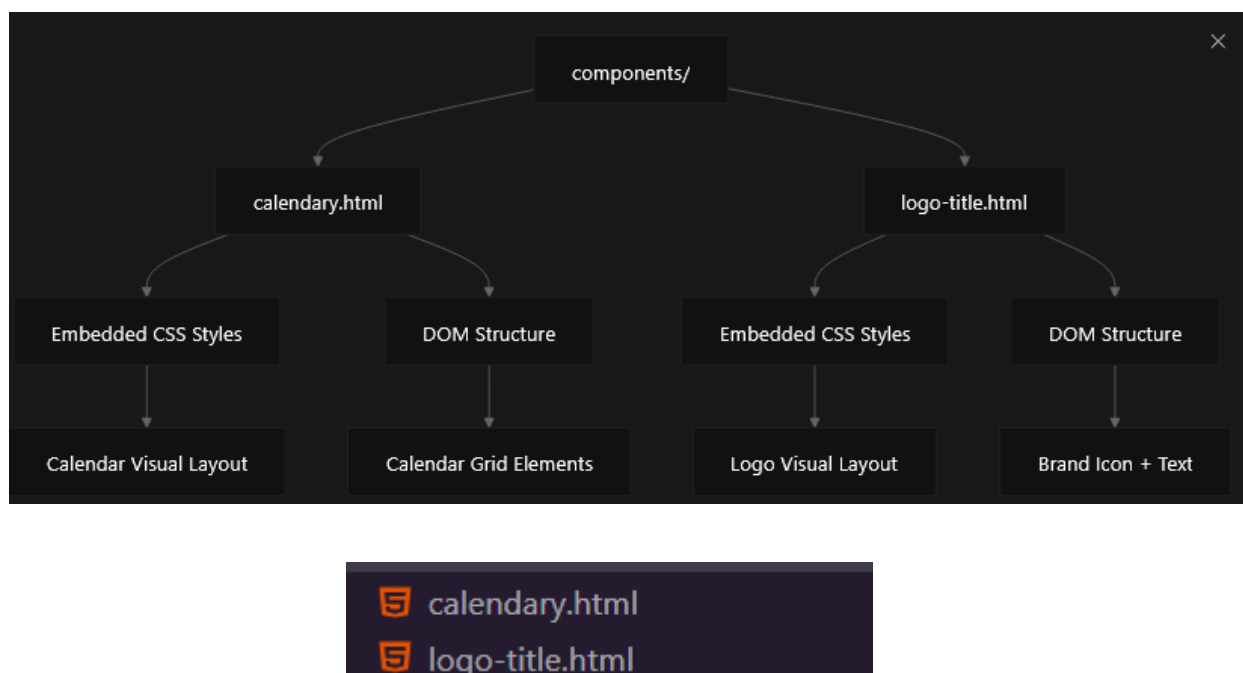
la interfaz de usuario proporcionan elementos visuales estandarizados que se pueden renderizar en diferentes partes de la aplicación para mantener la consistencia y reducir la duplicación de códigos.

Para obtener información sobre los componentes funcionales que manejan la lógica del negocio, consulte [Componentes Core](#). Para más detalles sobre activos estáticos como imágenes e iconos, vea [Activos y Recursos](#).

### 10.1. Arquitectura de componentes

El sistema utiliza una arquitectura basada en componentes donde cada componente de interfaz de usuario se implementa como un archivo HTML autónomo con estilo CSS integrado. Los componentes están diseñados para ser elementos visuales independientes que se pueden incluir en diferentes partes de la aplicación.

#### 10.1.1 Estructura de archivos de componentes



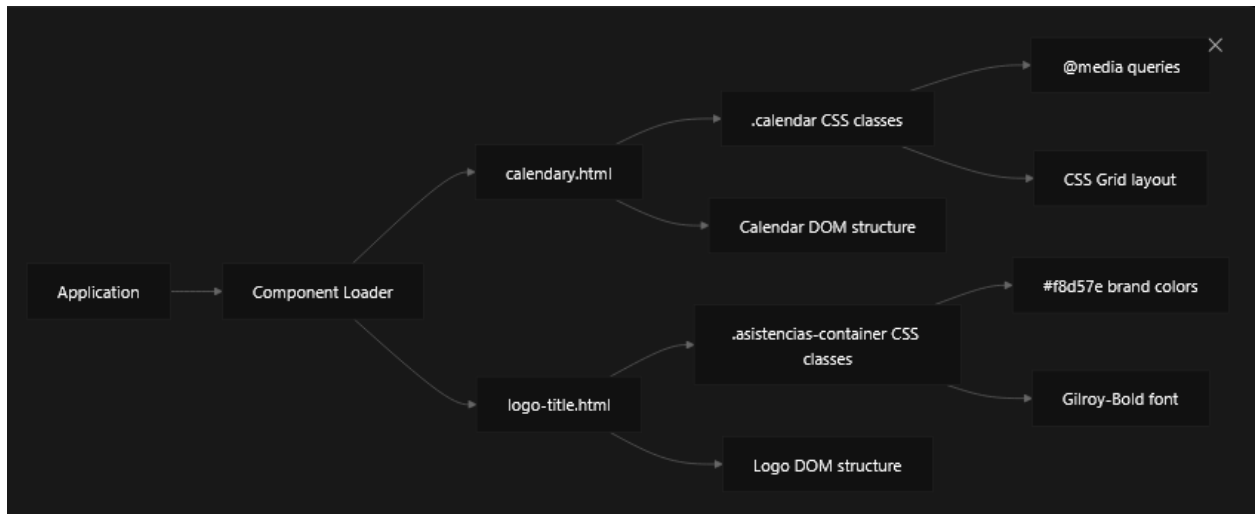


```
logo-title.html X
components > logo-title.html > style
1 <style>
27
28
29 .asistencias-text {
30   position: absolute;
31   left: 43.33px;
32   top: 2.74px;
33   color: black;
34   font-size: 20.12px;
35   font-family: 'Gilroy-Bold', sans-serif;
36   font-weight: 400;
37   line-height: 20.12px;
38   word-wrap: break-word;
39 }
40 </style>
41
42 <div class="asistencias-container">
43   <div class="asistencias-icon-bg"></div>
44   <div class="asistencias-icon-line"></div>
45   <div class="asistencias-text">Asistencias</div>
46 </div>
```

```
calendary.html X
components > calendary.html > style > .calendar
1 <style>
96
97 .calendar-day .dot.upper {
98   top: 4px;
99 }
100 </style>
101
102 <div class="calendar">
103   <div class="calendar-header">
104     <span class="calendar-month">March, 2025</span>
105     <span class="calendar-range">Two weeks</span>
106   </div>
107   <div class="calendar-body">
108     <div class="calendar-weekday">Mon</div>
109     <div class="calendar-weekday">Tue</div>
110     <div class="calendar-weekday">Wed</div>
111     <div class="calendar-weekday">Thur</div>
112     <div class="calendar-weekday">Fri</div>
113     <div class="calendar-weekday">Sat</div>
114     <div class="calendar-weekday">Sun</div>
115   </div>
116 </div>
```

Fuentes: [componentes/calendary.html11-144](#) [componentes/logo-title.html11-46](#)

### 10.1.1 Patrón de Integración de Componentes



Fuentes: [componentes/calendarly.html1-99](#) [componentes/logo-title.html1-40](#)

## 10.2. Componentes de IU disponibles

El sistema proporciona actualmente dos componentes principales de la interfaz de usuario:

Componente	Ruta de archivo	Objeto	Características clave
Calendario	components/calendarly.html	Exhibición y selección de fechas	Diseño de cuadrícula, diseño sensible, indicadores de actividad
Título del logo	components/logo-title.html	Marca de aplicaciones	Icono personalizado, tipografía, colores de marca

#### ▪ Estructura de componentes de calendario

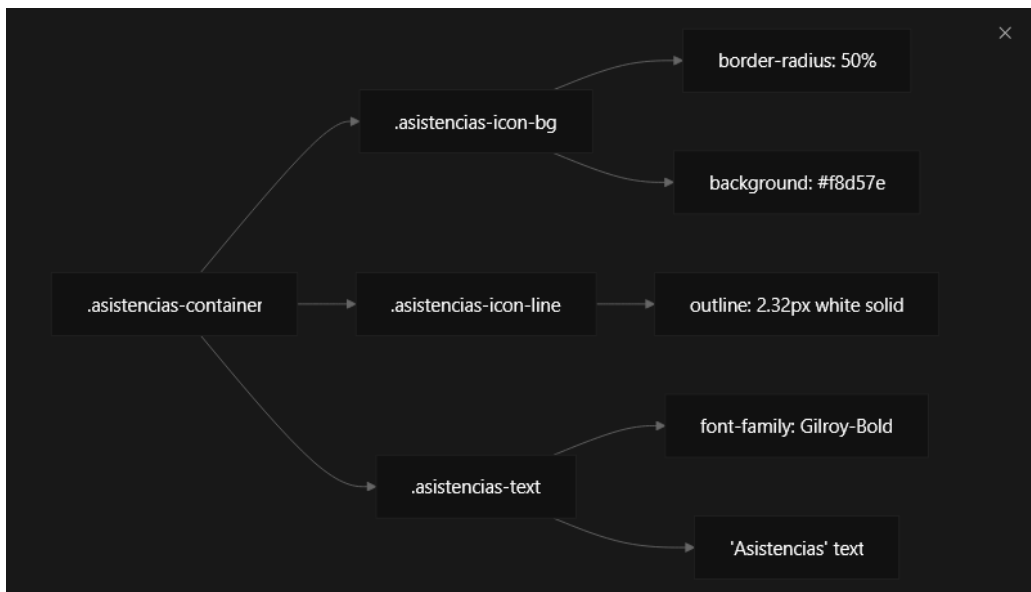
El componente del calendario aplica una visión mensual con los siguientes elementos clave:



Fuentes: [componentes/calendarary.html101 a 144](#)

### 10.1.2 Estructura de componentes de logotipo

El componente del logo combina un icono y un texto para la marca de aplicaciones:



Fuentes:

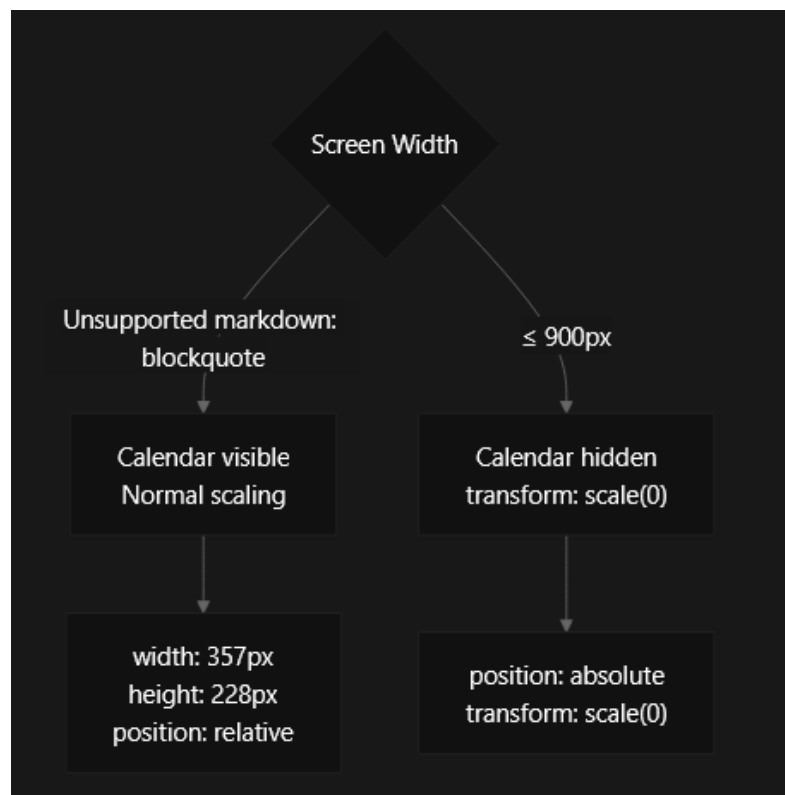
[componentes/logo-title.html42 - 46](#)

### 10.3. Características de diseño sensibles

#### 10.1.3 Calendar la respuesta

El componente de calendario incluye un comportamiento sensible para dispositivos móviles:

- **Escritorio** : Visual normalidad a 357px 228px
- **Móvil** (900px): Oculto con transform: scale(0) y el posicionamiento absoluto



Fuentes: [componentes/calendar.html12-17](#)



## 10.4. Arquitectura de estilo

### 10.1.4 Patrón de Organización del CSS

Ambos componentes siguen un patrón de organización CSS consistente:

1. **Estilos de contenedores:** Dimensiones y posicionamiento de elementos de raíz
2. **Estilos de diseño:** Flexbox, rejilla y propiedades de posicionamiento
3. **Estilo de tipografía:** Familias de fuentes, tamaños y colores
4. **Estilos de Estado:** Activo, descolorido e estados interactivos
5. **Estilos sensibles:** Consultas de medios para diferentes tamaños de pantalla

### 10.2.4 Esquema de color

Los componentes utilizan una paleta de color consistente:

Color	Código de hex	Uso
Marca amarillo	#F8D57E	Activos estados, indicadores, elementos de marca
Púrpura oscura	#2C2543	Texto primario, días naturales
Gray Mutado	#667180	Texto descolorido, elementos inactivos
Blanco	#FFFFFF	Antecedentes, esbozos

Fuentes:

[componentes/calendary.html](#)35 a 85 [componentes/logo-title.html](#)15

## 10.5. Integración de componentes

Estos componentes de la interfaz de usuario están diseñados para ser incrustados directamente en la estructura HTML de la aplicación. Proporcionan consistencia visual a diferentes áreas funcionales del sistema de gestión de asistencia, manteniendo su propio estilo y comportamiento encapsulado.

Para la implementación detallada de cada componente, vea el Componente y el Logotipo y la marca.

Fuentes: [componentes/calendar.html1-144](#) [componentes/logo-title.html1-46](#)

## 11. Activos y recursos

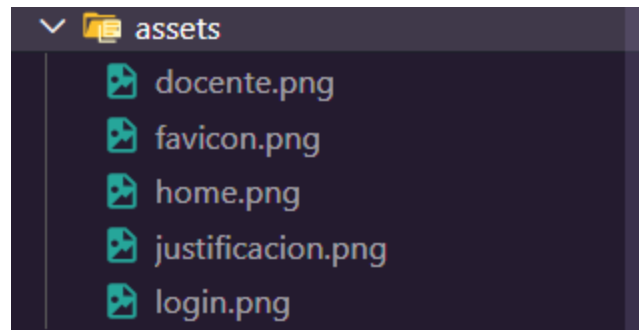
Esta página documenta los activos estáticos y los recursos utilizados en el sistema de gestión de asistencia al profesorado. Estos recursos incluyen imágenes, iconos y otros archivos estáticos que admiten la interfaz de usuario y la marca de la aplicación.

Los activos cubiertos aquí son archivos estáticos que no contienen código ejecutable. Para obtener información sobre los componentes de IU que utilizan estos activos, consulte Componentes de IU.

### ○ Estructura de Directorio de Activos

La aplicación mantiene todos los activos estáticos en un assets/directorio ubicado en la raíz del proyecto. Este directorio contiene archivos de imagen utilizados en toda la aplicación para elementos de interfaz de usuario, marca y componentes visuales.





**Fuentes:** [activos/docente.png1](#) [activos/favicon.png1](#) [activos/home.png1](#)

### 11.1. Activos de imagen

El sistema utiliza imágenes en formato PNG para elementos visuales e iconos. Todos los activos de imagen están optimizados para la visualización web y siguen convenciones de nombres consistentes utilizando terminología en español para que coincidan con el idioma objetivo de la aplicación.

Archivo de activos	Dimensiones	Objeto	Se utiliza en
docente.png	Variable	Maestra avatar/imagen de perfil	Componente de perfil del profesor
favicon.png	16x16, 32x32	Icono de la pestaña del navegador	Jefe de documento HTML
home.png	Variable	Icono de navegación de inicio	Sistema de navegación

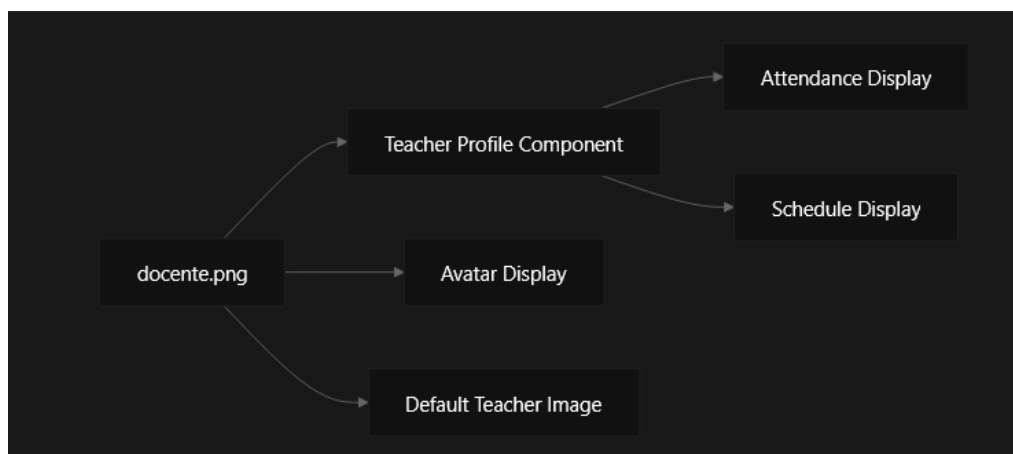
**Fuentes:**

[activos/docente.png1](#) [activos/favicon.png1](#)

[activos/home.png1](#)

## 11.2. Activos de Perfil del Maestro

El `docente.png` archivo sirve como la representación visual primaria para los perfiles de los profesores dentro del sistema. Esta imagen es probablemente referenciada por componentes que muestran información y perfiles del profesorado.



Fuentes: [activos/docente.png1](#)

## 11.3. Navegador y Navegación activos

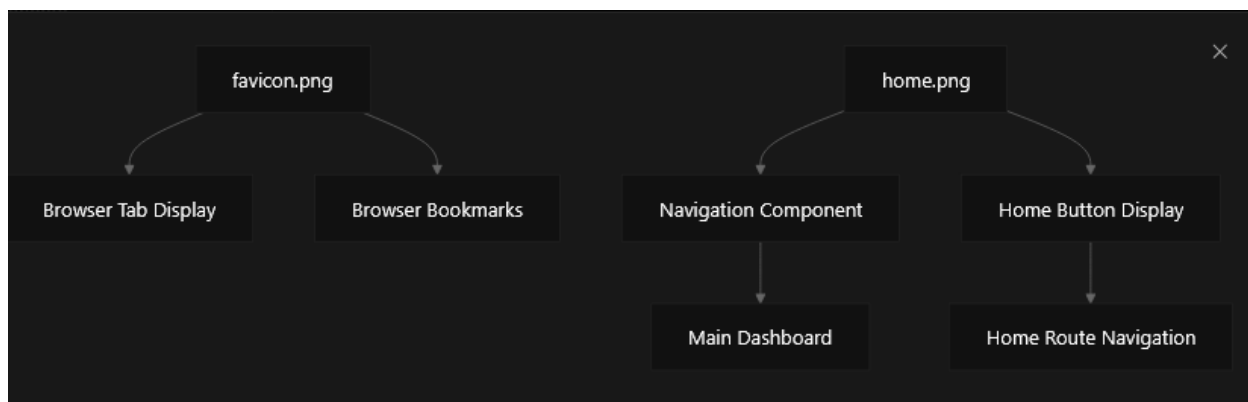
El sistema incluye activos dedicados para la integración del navegador y elementos de navegación:

### 11.1.3 Favicon

El `favicon.png` proporciona identidad de marca en pestañas del navegador y marcadores. Este pequeño icono ayuda a los usuarios a identificar la aplicación cuando se abren múltiples pestañas.

### 11.2.3 Icono de Home

El `home.png` sirve como un indicador visual para la navegación, proporcionando señales de navegación intuitivas para los usuarios.



Fuentes: [activos/favicon.png1](#)

[activos/home.png1](#)

#### 11.4. Carga de activos e integración

Los activos estáticos son cargados directamente por el navegador a través de elementos de imagen HTML estándar y referencias de fondo CSS. La estructura de directorio de activos permite una gestión y despliegue de activos sencillos.

##### *11.1.4 Estándas de formato de archivo*

Todos los activos visuales utilizan el formato PNG por las siguientes razones:

- Compresión sin pérdidas para la exhibición de iconos crujientes
- Apoyo a la transparencia para la integración flexible de IU
- Compatibilidad del navegador amplio
- Tamaño óptimo del archivo para la entrega en la web

##### *11.2.4 Convenios de nombramiento*

Los archivos de activos siguen las convenciones de nombres de español que se alinean con la base de usuarios objetivo de la aplicación:

- docente- Español para "maestro"
- Nombres descriptivos en inglés para elementos técnicos (favicon, home)

**Fuentes:** [activos/docente.png1](#) [activos/favicon.png1](#)

[activos/home.png1](#)

### 11.5. Mejores prácticas de la Organización de Activos

El centralizado assets/El directorio dispone:

1. **Fuente única de la verdad** - Todos los recursos estáticos en una sola ubicación
2. **Easy Maintenance** - Estructura de archivos simple para actualizaciones
3. **Finalidad clara** - Nombres descriptivos descriptivos indican uso
4. **Escalabilidad** - Habitación para categorías de activos adicionales

Esta organización es compatible con la arquitectura modular de componentes del sistema de gestión de asistencia, manteniendo al mismo tiempo una clara separación entre los recursos estáticos y el código ejecutable.

**Fuentes:**

[activos/docente.png1](#) [activos/favicon.png1](#) [activos/home.png1](#)

## 12. Configuración de desarrollo

Este documento cubre la configuración del entorno de desarrollo y el flujo de trabajo para el sistema de gestión de la asistencia al profesorado. Proporciona los requisitos de configuración técnica, la configuración del servidor de desarrollo y las directrices para trabajar con la arquitectura basada en componentes de la base de código.

Para obtener información sobre la arquitectura general del sistema y las interacciones con los componentes, consulte [Arquitectura del Sistema](#). Para más detalles sobre componentes funcionales específicos, consulte [Componentes Core](#).

### 12.1. Requisitos del entorno para el desarrollo

El proyecto se configura como una aplicación JavaScript del lado del cliente que se puede desarrollar utilizando herramientas de desarrollo web estándar. El entorno de desarrollo primario utiliza el código de estudio visual con servidor vivo para el desarrollo local.

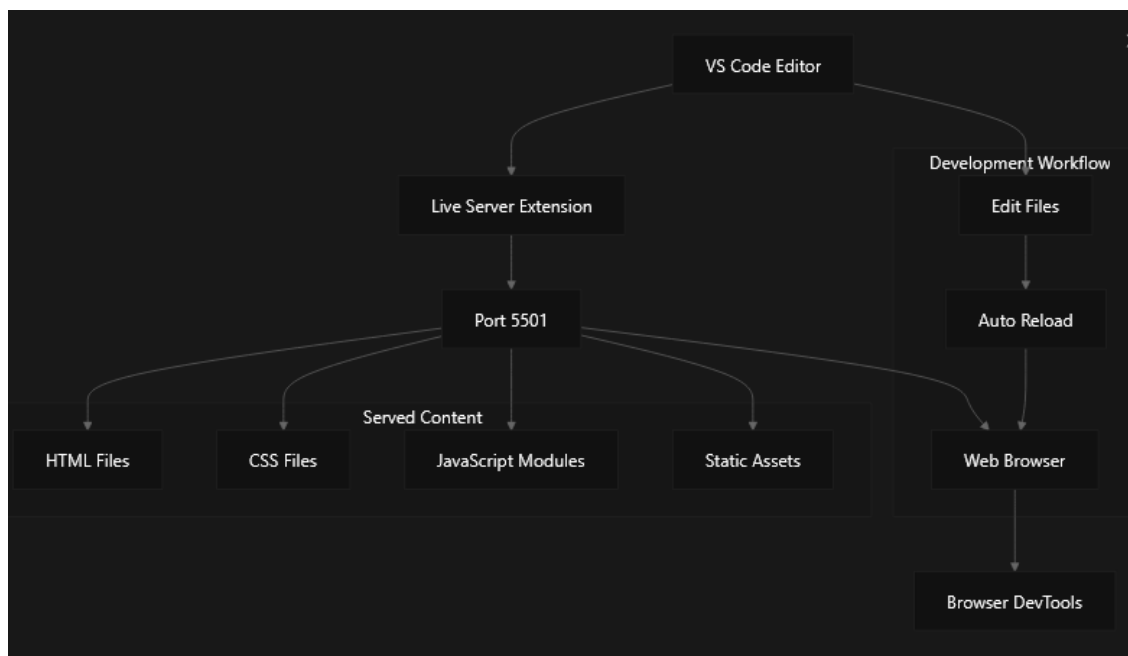
#### 12.1.1 Herramientas requeridas

Herramienta	Objeto	Configuración
Código de Estudio Visual	IDE primaria	Configurado con extensión de Servidor en Vivo
Extensión de servidor en vivo	Servidor de desarrollo	Puerto 5501 (configurado)
Navegador web moderno	Pruebas y depuración	Soporte de módulo ES6 necesario

La configuración de Live Server se define en [.vscode/settings.json1 a 3](#) que establece el servidor de desarrollo para funcionar en el puerto 5501.

Fuentes: [.vscode/settings.json1 a 3](#)

## 12.2. Configuración de servidor de desarrollo



El servidor de desarrollo sirve automáticamente a los archivos estáticos y proporciona funcionalidad de recarga en vivo cuando los archivos se modifican. La configuración del servidor garantiza que los módulos ES6 se sirvan correctamente con los tipos MIME correctos.

Fuentes:

[.vscode/settings.json2](https://code.visualstudio.com/docs/editor/remote-explorer#_live-server)

## 12.3. Proyecto de Arquitectura para el Desarrollo

La base de código sigue una arquitectura modular basada en componentes que facilita el desarrollo y el mantenimiento. Cada área funcional se organiza en componentes discretos con una clara separación de preocupaciones.





#### 12.4. Flujo de trabajo de desarrollo de componentes

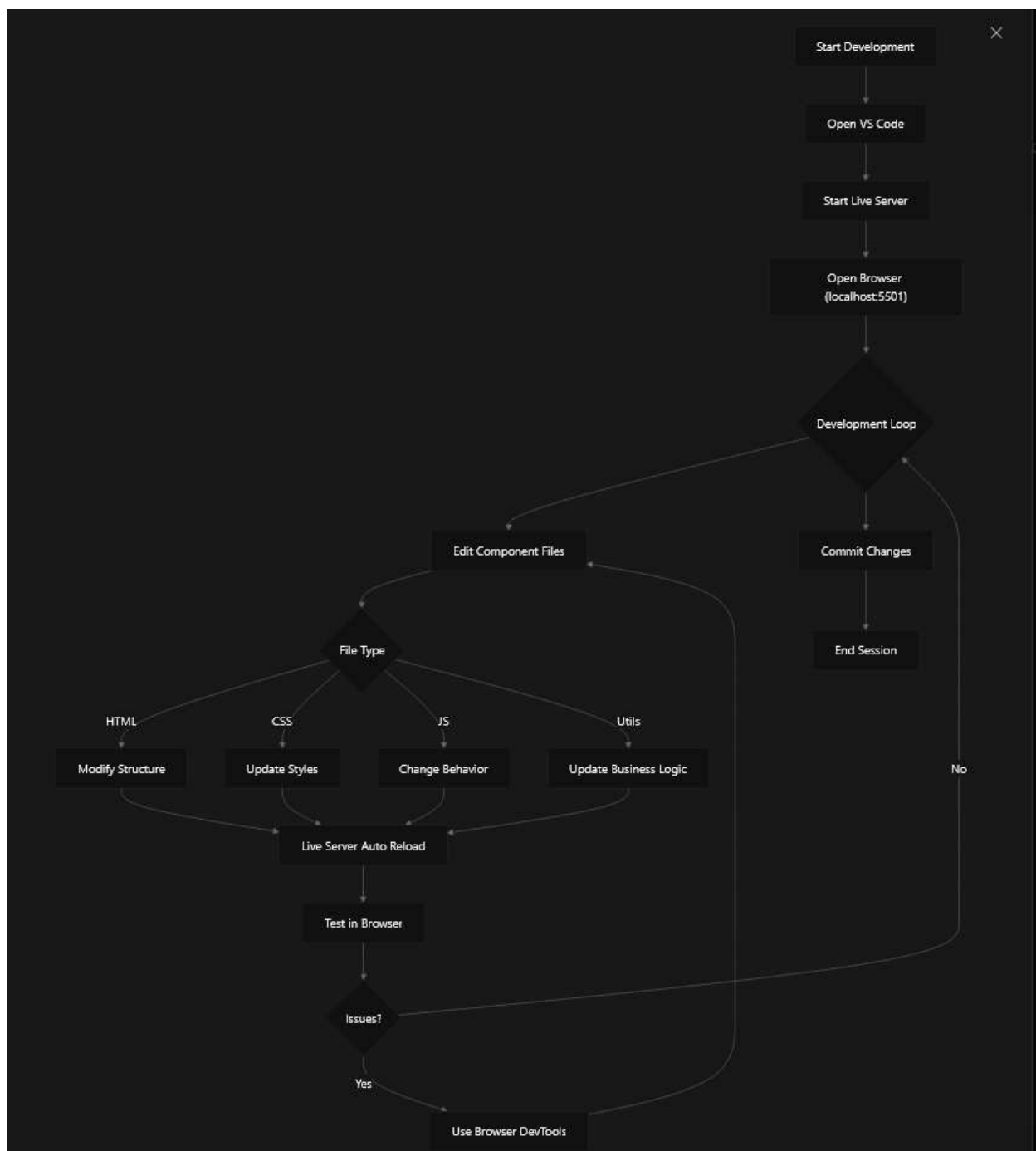
Cada componente sigue una estructura coherente que simplifica el desarrollo:

1. **Plantilla HTML** (component.html- Define la estructura de los componentes
2. **Estilo** (component.css- Estilos específicos de los componentes
3. **Comportamiento (Comportamiento)** component.js- La inicialización de los componentes y la manipulación de eventos
4. **Lógica de Negocios** (useComponent.utils.js- Manipulación de datos y funcionalidad de núcleos

Fuentes: Análisis de arquitectura del sistema de diagramas proporcionados

#### 12.5. Flujo de trabajo para el desarrollo

El proceso de desarrollo aprovecha la arquitectura modular para permitir un desarrollo eficiente basado en componentes.



## 12.6. Consideraciones clave para el desarrollo

Aspecto	Aplicación	Notas
Cargando del módulo	Módulos ES6	Requiere un navegador moderno o servidor de desarrollo
Aislamiento del componente	Separar archivos HTML/CSS/JS	Mantiene una clara separación de las preocupaciones

Gestión de datos	Centralizado Dataclase	Fuente única de la verdad para el estado de aplicación
Lanzamiento	Acoso useRoutessistema	Navegación del lado del cliente sin recargas de la página

Fuentes: [.vscode/settings.json1 a 3](#) Análisis de la arquitectura del sistema

## 12.7. Medidas de configuración del entorno para el desarrollo

1. **Instale Visual Studio Code** - Descargar e instalar la última versión
2. **Instalar extensión de servidor en vivo** - Agregar la extensión del servidor en vivo a código VS
3. **Proyecto Clone/Download** - Obtener los archivos del proyecto
4. **Abrir Proyecto en Código VS** - Abra el directorio raíz del proyecto
5. **Inicia Live Server** - Haga clic con el botón derecho en un archivo HTML y seleccione "Abrir con servidor en vivo"
6. **Verificar Configuración** - Asegúrese de que el servidor se inicia en el puerto 5501 configurado

El servidor en vivo detectará automáticamente los cambios de archivo y recargará el navegador, proporcionando una experiencia de desarrollo eficiente para la arquitectura basada en componentes.

Fuentes: [.vscode/settings.json1 a 3](#)

## 12.8. Requisitos del navegador

La aplicación requiere un navegador web moderno con soporte para:

- Módulos ES6 y sintaxis de importación/exportación

- Características modernas de JavaScript utilizadas en utilidades de componentes
- Características CSS3 para el peinado de componentes
- API de manipulación DOM para la representación de contenidos dinámicos

Las pruebas deben realizarse a través de varios navegadores para garantizar la compatibilidad con la base de usuario de destino.

Fuentes: Análisis de arquitectura del sistema de diagramas proporcionados

## 13. Vistas de la pagina

### 13.1. Login



The image shows a login form for Uleam, Universidad Laica Eloy Alfaro de Manabí. The form is centered on a light blue background. At the top, the Uleam logo and name are displayed. Below the logo, there is a text prompt: "Ingresa tus credenciales para acceder a tu sesión.s". The login form itself is a white card with a blue header "Login" and a blue underline. It contains two input fields: one for the email address "juanperez@gmail.com" and another for the password, represented by six dots. A dark grey "LOGIN" button is positioned at the bottom of the card.

**Uleam**  
UNIVERSIDAD LAICA ELOY ALFARO DE MANABÍ

Ingresa tus credenciales para acceder a tu sesión.s

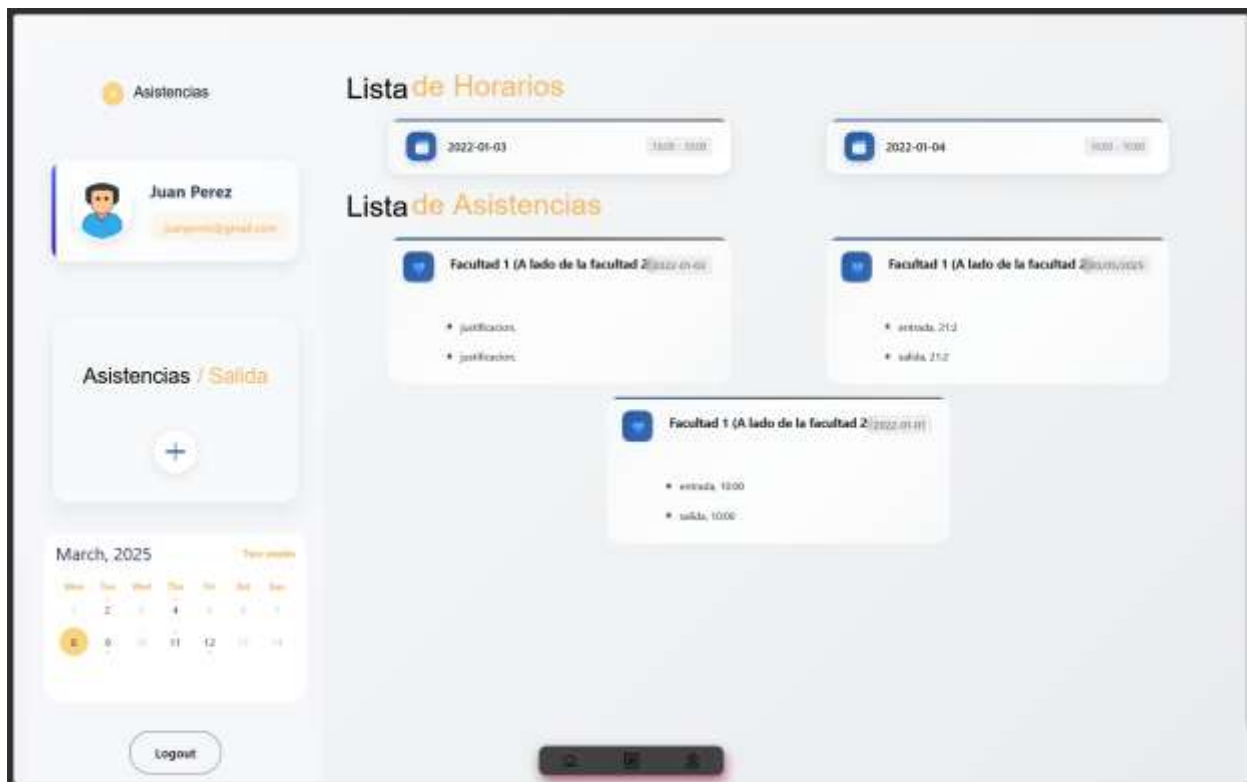
**Login**

juanperez@gmail.com

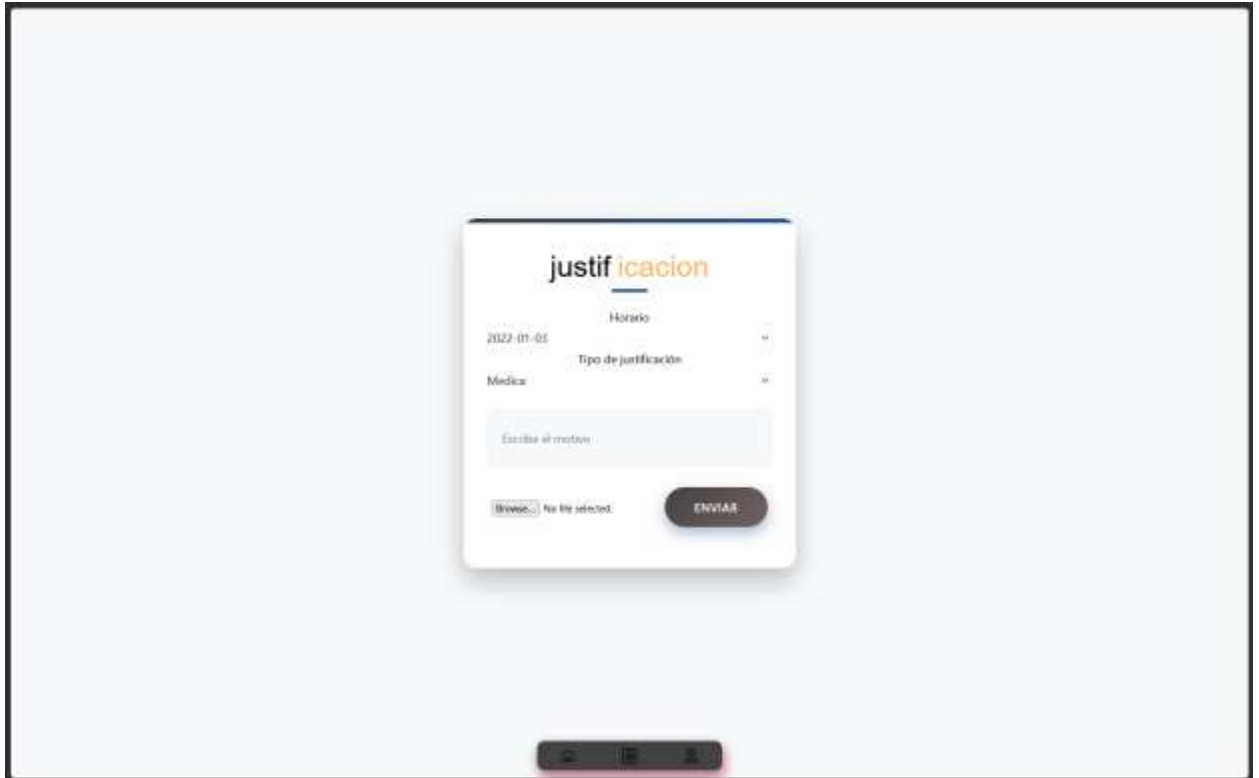
.....

LOGIN

## 13.2. Home



### 13.3. Justificación



The screenshot shows a mobile application interface for submitting a justification. The app is titled "justificación" in a stylized font. The form includes the following fields and elements:

- Horario:** A date field showing "2022-01-05".
- Tipo de justificación:** A dropdown menu.
- Medica:** A dropdown menu.
- Escritura al motivo:** A text input field.
- Upload button:** A button labeled "Browse" with the text "No file selected" below it.
- Submit button:** A dark button labeled "ENVIAR".

The interface is displayed on a light blue background, and the app is shown within a mobile device frame.