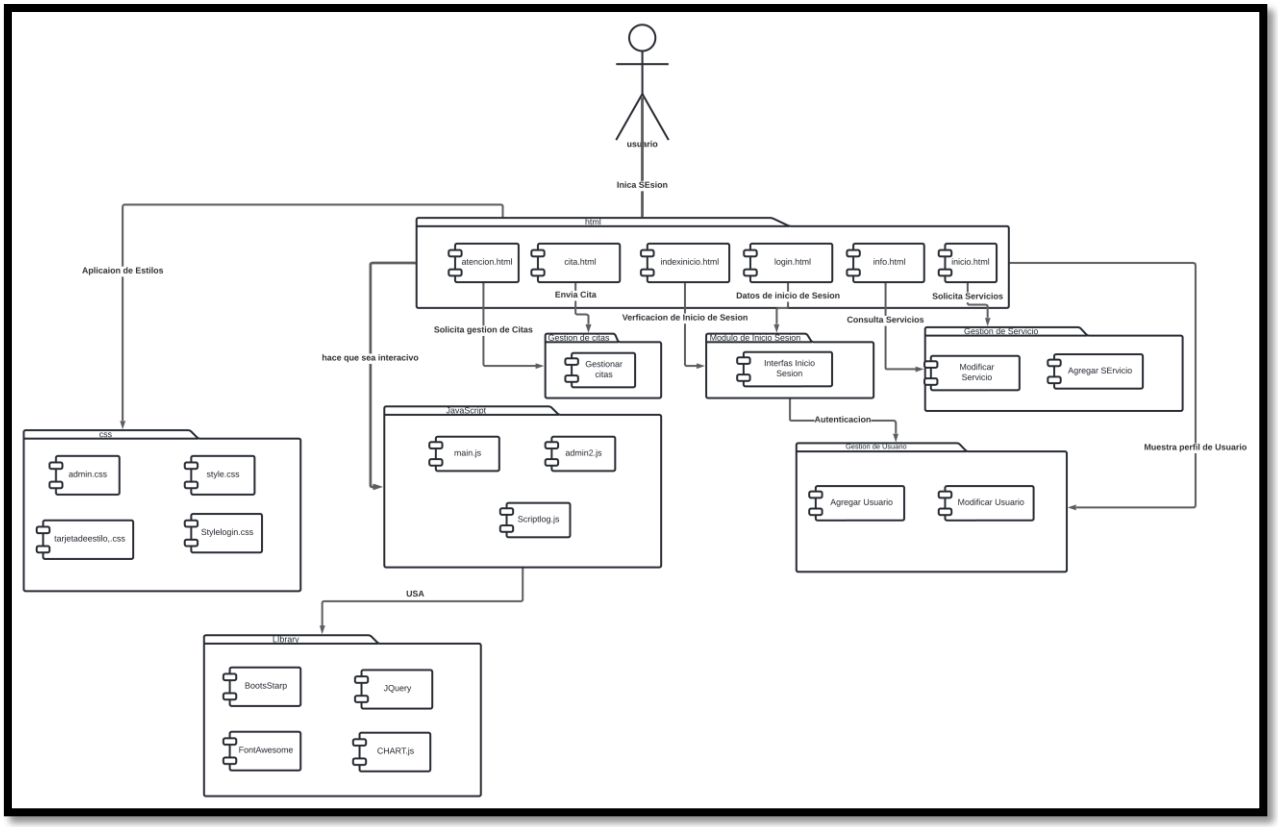

Nuestro proyecto sigue una **arquitectura de 3 capas**. Esta arquitectura organiza la aplicación en tres capas que se comunican entre sí, donde cada capa tiene una función específica y está diseñada para trabajar de manera independiente. Aquí te explico cómo funciona cada capa en tu arquitectura:

1. Capa de Presentación (Front-End)

- **Descripción**: Esta es la capa visible para el usuario y es donde ocurre la interacción directa con el sistema. Se encarga de mostrar la interfaz gráfica y de recibir las acciones de los usuarios (como clics, formularios completados, etc.).
- **En tu proyecto**:
    - Tienes **páginas HTML** como Atencion.html, Cita.html, indexInicio.html, Info.html, inicio.html, y login.html, que se encargan de la visualización y presentación de la información.
    - Las **hojas de estilo CSS** (sb-admin-2.css, style.css, stylecard.css, stylelogin.css) se encargan de la apariencia visual, como los colores, el formato, y la disposición del contenido.
    - Los **archivos JavaScript** (main.js, sb-admin-2.js, scriptlog.js) añaden interactividad a las páginas, permitiendo, por ejemplo, validaciones de formularios, animaciones, o respuestas dinámicas a las acciones del usuario.
- **Función**:

- El usuario interactúa con las páginas HTML, que dependen de las hojas de estilo y JavaScript para presentar la interfaz de manera adecuada y funcional.
- Esta capa se comunica con la capa de lógica de negocio (Back-End) cuando el usuario envía formularios (ej. login, citas), disparando acciones que el sistema debe procesar.

2. Capa de Lógica de Negocio (Back-End)

- **Descripción**: Esta es la capa que procesa toda la lógica del negocio y las reglas que definen el comportamiento del sistema. Se encarga de procesar las peticiones que provienen de la capa de presentación, interactuar con la base de datos y devolver la información o resultado al usuario.
- **El proyecto**:
  - Incluye módulos como **Gestión de Usuarios**, **Gestión de Servicios**, y **Gestión de Citas**.
  - Por ejemplo, cuando un usuario completa el formulario de inicio de sesión (login.html), los datos se envían al módulo de **Interfaz de Inicio de Sesión**, que valida los datos y autentica al usuario.
  - Para las páginas de citas, el módulo de **Gestión de Citas** recibe las solicitudes y realiza las operaciones necesarias (agregar, modificar citas).
  - **Operaciones como agregar/modificar usuarios o servicios** también se gestionan aquí, aplicando la lógica de negocio.
- **Función**:
  - Esta capa procesa las reglas del negocio, es decir, toma las acciones del usuario desde la capa de presentación, realiza las verificaciones o cálculos necesarios, y devuelve una respuesta adecuada. Por ejemplo, si un usuario envía sus credenciales para iniciar sesión, esta capa valida las credenciales y decide si otorgar o no el acceso.
  - Esta capa también interactúa con la base de datos para obtener o actualizar información según las acciones del usuario.

3. Capa de Acceso a Datos (Base de Datos)

- **Descripción**: Esta capa es responsable de gestionar el acceso y la persistencia de los datos. Aquí es donde se almacenan y recuperan los datos de la aplicación, como la información de usuarios, servicios, citas, entre otros.
- **El proyecto**:

    - Aunque no está reflejada explícitamente en el diagrama, es parte esencial del sistema. Tu base de datos en **XAMPP** con **MariaDB** se utiliza para almacenar la información necesaria, como los datos de los usuarios, servicios, citas, etc.
    - La capa de lógica de negocio interactúa con esta capa para realizar operaciones CRUD (crear, leer, actualizar, eliminar) sobre los datos de la base de datos.
    - Las tablas que mencionaste en tu proyecto, como Cliente, Categorias, Productos, Movilidad, Pedidos, y Citas, forman parte de esta capa.
- **Función**:
    - La capa de datos responde a las solicitudes de la capa de lógica de negocio, ya sea para recuperar datos (por ejemplo, mostrar la lista de citas de un usuario) o para almacenar nueva información (como registrar un nuevo usuario o actualizar un servicio).
    - Cuando el módulo de **Gestión de Usuarios** agrega un nuevo usuario, esta información se guarda en la base de datos a través de esta capa.

Otros tipos de arquitecturas que se podrían relacionar parcialmente:

1. **Arquitectura MVC (Model-View-Controller)**: Aunque nuestro proyecto no sigue este patrón al pie de la letra, algunos aspectos del MVC podrían estar presentes. El **Front-End** actuaría como la "Vista", los módulos de negocio (gestión de usuarios, servicios, etc.) como el "Controlador", y la base de datos sería el "Modelo".
2. **Arquitectura basada en servicios (SOA)**: Si en el futuro decidimos desacoplar los módulos de gestión (usuarios, citas, servicios) en servicios independientes, podríamos evolucionar hacia una arquitectura basada en servicios o microservicios.