### `app.py`

Este archivo configura y ejecuta una aplicación web usando Flask. Aquí están los elementos clave:

1. \*\*Configuración de la Aplicación Flask:\*\*

- `app = Flask(\_\_name\_\_)`: Crea una instancia de la aplicación Flask.

- `app.config.from\_object(Config)`: Carga configuraciones desde `Config`.

2. \*\*Clase `AsistenteVirtual`:\*\*

- Maneja un asistente virtual que responde preguntas y guarda un historial de preguntas y respuestas en una base de datos SQLite.

- La base de datos se crea y se gestiona en el constructor de la clase.

- El método `responder()` genera una respuesta y la guarda en la base de datos.

3. \*\*Rutas de Flask:\*\*

- `/`: Renderiza `index.html`.

- `/api/modulos`: Devuelve una lista de módulos en formato JSON.

- `/api/tareas`: Devuelve una lista de tareas en formato JSON.

- `/api/notificaciones`: Devuelve una lista de notificaciones en formato JSON.

- `/api/asistente`: Maneja solicitudes POST para interactuar con el asistente virtual, generando respuestas y devolviéndolas en formato JSON.

### `config.py`

Define la configuración para la aplicación Flask:

- `SECRET\_KEY`: Usado para la seguridad de la sesión.

- `CLAUDE\_API\_KEY`: Una clave de API para acceder a un servicio externo (probablemente para el asistente virtual).

### `main.css`

Contiene estilos para la interfaz web. Define la apariencia de los botones, las listas de tareas y notificaciones, y el área del asistente virtual. Incluye:

- Estilos para botones rectangulares y de submódulos.

- Estilos para listas de tareas y notificaciones.

- Estilos para el área del asistente virtual y la ventana de chat.

### `main.js`

Este archivo JavaScript maneja la lógica del lado del cliente para la interfaz web. Incluye:

- \*\*Funciones para Crear Botones:\*\*

- `createRectangularButton()`: Crea botones rectangulares con un color de fondo.

- `createSubmoduleButton()`: Crea botones para submódulos.

- \*\*Carga Dinámica de Datos:\*\*

- `loadModules()`, `loadSubmodules()`, `loadSubmoduleContent()`: Cargan módulos, submódulos y su contenido en la página web.

- `loadTasks()`, `loadNotifications()`: Obtiene y muestra tareas y notificaciones desde las rutas de la API.

- \*\*Interacción con el Asistente Virtual:\*\*

- `enviarPreguntaAsistente()`: Envía una pregunta al asistente virtual y muestra la respuesta en la ventana de chat.

### `base.html`

Proporciona una plantilla base para otras páginas HTML. Incluye:

- Un encabezado (`<header>`) con el título "CalculAI".

- Un menú lateral (`<nav>`) y contenido principal (`<main>`).

- Un área de asistente virtual (`<aside>`) con un área de chat y un campo de entrada para preguntas.

- Incluye enlaces a los archivos CSS y JS.

### `erp\_logic.py`

Contiene la lógica del backend para el sistema ERP. Incluye:

- \*\*Clase `AsistenteVirtual`:\*\*

- Similar a la clase en `app.py`, pero con una implementación diferente que usa una API externa para generar respuestas.

- \*\*Clase `ERP`:\*\*

- Maneja los módulos del ERP y proporciona métodos para obtener datos históricos y de análisis.

- \*\*Clase `ModuloBanco`:\*\*

- Maneja transacciones y cuentas bancarias, con métodos para obtener datos y realizar transacciones.

### `index.html`

Extiende `base.html` y añade contenido específico para la página principal:

- Botones para módulos y submódulos.

- Listas de tareas y notificaciones.

- Gráficos y un calendario.

- Área de asistente virtual con un área de chat.

\*\*Resumen:\*\* La aplicación web en conjunto parece ser un sistema ERP con un asistente virtual integrado. Los archivos proporcionan una estructura para la interfaz web, la lógica del servidor, y la integración con una API externa para la generación de respuestas del asistente. El estilo y la funcionalidad se gestionan a través de CSS y JavaScript, mientras que el backend maneja la lógica de datos y respuestas.