Frontend (Web App SmartEnviro ISPC)
1. Descripción general

El archivo cod 4 frontend.txt forma parte del módulo Web App del ecosistema SmartEnviro ISPC, una aplicación web que permite visualizar, analizar y gestionar los datos ambientales recolectados por los sensores del sistema IoT.

Esta capa representa la interfaz gráfica del usuario (UI) y funciona como el punto de interacción entre el backend (API REST) y los usuarios finales (productores agrícolas, técnicos y administradores).

Está diseñada con un enfoque reactivo, modular y responsive, lo que permite su correcta visualización tanto en computadoras como en dispositivos móviles.

2. Función principal del Frontend

El frontend:

Recibe datos del backend mediante peticiones HTTP o WebSocket.

Procesa la información en el navegador.

Muestra gráficos, alertas y estados de dispositivos en tiempo real.

Permite la configuración de parámetros y la gestión de usuarios.

Ofrece reportes e indicadores para la toma de decisiones.

3. Arquitectura y tecnologías utilizadas

El código implementa una arquitectura SPA (Single Page Application), lo que significa que la aplicación se carga una sola vez y se actualiza dinámicamente sin recargar toda la página.

🛠️ Tecnologías base:

Componente	Descripción
React.js / Vue.js / JavaScript moderno (ES6)	Motor principal del frontend.
HTML5 + CSS3 (Tailwind o Bootstrap)	Estructura y diseño responsivo.
Axios / Fetch API	Comunicación asíncrona con el backend.
Chart.js / Recharts	Visualización gráfica de los datos ambientales.
JWT (JSON Web Token)	Manejo de autenticación segura.
NGINX / Docker	Despliegue y contenedorización.

4. Estructura del código

El archivo muestra la estructura típica de un frontend modular, dividido en componentes y vistas.

🔹 Componentes principales:

Header / Navbar:

Contiene el logo, el nombre del sistema y accesos rápidos a secciones principales (Dashboard, Sensores, Endpoints, Configuración, etc.).

Sidebar:
Menú lateral con navegación rápida entre módulos.

Dashboard:
Componente central que muestra los datos en tiempo real con gráficos dinámicos.

Endpoints y Sensores:
Páginas que listan dispositivos activos, con información sobre su estado, batería, ubicación y lecturas más recientes.

Alertas y Reportes:
Módulos que muestran alertas generadas por el sistema y permiten la descarga de reportes históricos.

Login / Auth:
Página de inicio de sesión que valida credenciales y genera el token JWT para mantener la sesión activa.

5. Flujo de funcionamiento

🔄 1. Inicio de sesión

El usuario ingresa sus credenciales en la pantalla de acceso:

```
POST /api/auth/login
{
  "username": "admin",
  "password": "********"
}
```

El backend responde con un token JWT, que se guarda temporalmente en el navegador (localStorage o cookies seguras).

🔄 2. Carga del Dashboard

Una vez autenticado, la aplicación solicita los datos al backend:

```
GET /api/sensors/current
GET /api/endpoints/status
```

Los resultados se renderizan en gráficos e indicadores visuales:

Temperatura

Humedad

Presión atmosférica

Calidad del aire

Estado de endpoints (Online/Offline)

🔄 3. Actualización en tiempo real

Cada cierto intervalo (por ejemplo, cada 30 o 60 segundos), el frontend vuelve a consultar los datos actualizados.
Esto se realiza mediante Axios o WebSockets, dependiendo de la configuración.

Ejemplo:

```
setInterval(() => {
  axios.get("/api/sensors/current").then(updateDashboard);
}, 60000);
```

🔄 4. Interacción del usuario

El usuario puede:

Seleccionar un sensor o endpoint específico.

Consultar su historial (últimas 24 h o 7 días).

Cambiar límites de alerta o configuraciones de notificación.

Generar reportes descargables en formato PDF o CSV.

6. Comunicación con el Backend

El frontend se comunica con el backend usando peticiones HTTPS y autenticación con token JWT.

Ejemplo de consulta:

```
axios.get("https://api.gonaiot.com/plata/datos_dispositivos/", {
  headers: {
    Authorization: `Bearer ${token}`,
  },
});
```

Ejemplo de respuesta:

```
{
  "sensor_id": "01",
  "temperature": 25.4,
  "humidity": 58.9,
  "location": "Silo_Norte",
  "timestamp": "2025-10-28T14:00:00Z"
}
```

El frontend toma estos datos y los representa visualmente en el dashboard con gráficos, colores y alertas.

7. Seguridad y autenticación

Autenticación basada en JWT (JSON Web Token).

Encriptación de las credenciales mediante HTTPS.

Restricción de acceso por roles:

Administrador: Acceso total.

Técnico: Configuración y mantenimiento.

Productor: Visualización y reportes.

8. Diseño visual y experiencia de usuario

El frontend aplica un diseño minimalista e intuitivo, enfocado en la claridad de la información:

🎨 Colores principales: verde (ambiental) y azul (tecnología).
📱 Diseño responsive: adaptable a PC, tablet y smartphone.
📊 Gráficos dinámicos: muestran evolución de variables.
🔔 Alertas visuales: cambios de color y notificaciones instantáneas.

9. Ejemplo de flujo visual

Login: Usuario ingresa → backend valida credenciales.

Dashboard: Se cargan los datos del backend → se muestran gráficos.

Sensores: Usuario selecciona sensor → se muestra historial.

Configuración: Usuario ajusta límites → se guardan cambios.

Reportes: Se genera archivo descargable → PDF/CSV.

Todo este flujo ocurre sin recargar la página (SPA).

10. Ventajas del Frontend SmartEnviro ISPC

✅ Acceso remoto y multiplataforma.
✅ Interfaz clara y moderna.
✅ Actualización de datos en tiempo real.
✅ Sistema de alertas y reportes.
✅ Compatibilidad con API REST.
✅ Fácil mantenimiento e integración con nuevas funciones.

11. Despliegue

La aplicación puede ejecutarse localmente o en contenedores Docker:

```
frontend:
  build: ./frontend
```

```
  container_name: smartenviro_frontend
  ports:
    - "8080:80"
  depends_on:
    - backend
```

Servida mediante NGINX en modo producción y conectada al backend Flask/Node.js.

12. Conclusión

El frontend SmartEnviro ISPC constituye la capa de presentación inteligente del sistema IoT, ofreciendo una experiencia de usuario moderna y eficaz.
Permite transformar grandes volúmenes de datos IoT en información visual y accionable, accesible desde cualquier dispositivo conectado.

Su integración con el backend mediante API REST garantiza confiabilidad, seguridad y escalabilidad para la gestión remota de entornos agrícolas y ambientales.