AtmosInsight – Análisis Técnico del Código Fuente (Versión Detallada)

Prototipo de repositorio: visor web de NASA GIBS con narración tipo podcast mediante Google Cloud Text-to-Speech (TTS). Este informe analiza cada archivo fuente, explica procedimientos clave, flujos de ejecución, contrato de API y sugiere pruebas y despliegue.

# 1. Visión General de Arquitectura

* Frontend (estático): index.html, styles.css, app.js (visor Leaflet, controles de capa/fecha, disparador TTS).
* Backend (Node/Express): server.js expone POST /narrate para Google TTS y sirve activos estáticos.
* Paquete: package.json define dependencias y script de inicio.
* Docs: README.md con ejecución local y configuración de credenciales.

Flujo de datos: Usuario >> selecciones UI >> construir URL WMTS >> Leaflet pide mosaicos GIBS >> (opcional) POST /narrate >> Google TTS >> MP3 Base64 >> HTMLAudioElement.

# 2. Análisis por Archivo

## 2.1 index.html (estructura y dependencias)

Función: Declara la estructura de la interfaz, carga Leaflet desde CDN, define controles (capa, fecha, voz) y arranca app.js. Incluye panel lateral para metadatos y reproducción de audio.

<link rel="stylesheet" href="https://unpkg.com/leaflet@1.9.4/dist/leaflet.css"/>  
<script src="https://unpkg.com/leaflet@1.9.4/dist/leaflet.js"></script>  
<select id="layer">MODIS/VIIRS/SMAP/TROPOMI/OPERA...</select>  
<input type="date" id="date"/>  
<button id="gen">Generate Narration</button>  
<select id="voice">en-US / es-ES / es-CO ...</select>  
<div id="map"></div>  
<aside id="panel">Info de capa • URL de tiles • Narración • <audio/></aside>  
<script src="./app.js"></script>

## 2.2 styles.css (maquetación y responsividad)

Función: Maquetación de dos columnas (mapa + panel), encabezado fijo con grupo de controles, paleta sistema y fallback móvil a una columna.

main{display:grid;grid-template-columns:1fr 380px}  
#map{height:100%}  
#panel{overflow:auto;background:#fff;border-left:1px solid #e5e7eb;padding:12px}  
@media (max-width:980px){ main{grid-template-columns:1fr} }

## 2.3 app.js (lógica de UI, integración GIBS, narración)

Función: Inicializa el mapa Leaflet, construye URLs WMTS para GIBS (EPSG:3857), actualiza mosaicos según selecciones, arma el texto de narración y solicita audio al backend.

function wmtsUrl(layer,date){  
 return `https://gibs.earthdata.nasa.gov/wmts/epsg3857/best/${layer}/default/${date}/GoogleMapsCompatible\_Level9/{z}/{y}/{x}.png`;  
}  
function defaultUTCDate(){ /\* ayer UTC, YYYY-MM-DD \*/ }  
const map=L.map("map",{center:[4.711,-74.072],zoom:4,minZoom:1,maxZoom:9});  
let tileLayer=L.tileLayer(wmtsUrl(layer,date),{ attribution:"© NASA EOSDIS GIBS" }).addTo(map);

Procedimientos clave:

* refreshLayer(): recalcula la URL WMTS y actualiza el TileLayer mediante setUrl().
* buildNarrative(): compone un texto explicativo breve con etiqueta de capa + fecha.
* Manejadores: cambios en <select id='layer'> y <input id='date'> disparan refreshLayer().
* Botón de narración: POST /narrate { text, voice } >> espera { audioContent: base64 } >> asigna a <audio> y reproduce.

## 2.4 server.js (Express + Google Cloud TTS)

Función: Sirve el frontend estático y expone POST /narrate que sintetiza audio MP3 con Google Cloud TTS. Mapea el código de voz al languageCode y devuelve Base64 al cliente.

const client=new tts.TextToSpeechClient();  
app.post("/narrate", async (req,res)=>{  
 const { text, voice="en-US-Neural2-C" }=req.body||{};  
 const [response]=await client.synthesizeSpeech({  
 input:{text},  
 voice:{ languageCode: voice.startsWith("es-")?"es-ES":"en-US", name:voice },  
 audioConfig:{ audioEncoding:"MP3", speakingRate:1.02 },  
 });  
 res.json({ audioContent: response.audioContent.toString("base64") });  
});

## 2.5 package.json (ejecución y dependencias)

{  
 "scripts": { "start": "node server.js" },  
 "dependencies": { "express":"^4.19.2","cors":"^2.8.5","@google-cloud/text-to-speech":"^5.3.0" }  
}

## 2.6 README.md (uso)

Incluye pasos para instalación, export de credenciales y arranque en http://localhost:8787.

# 3. Flujo de Ejecución

Usuario >> UI (capa/fecha/voz)  
 >> app.js (construye URL WMTS y actualiza Leaflet)  
 >> (opcional) POST /narrate  
 >> Google TTS >> MP3 Base64 >> <audio>.play()

# 4. Contrato de API – POST /narrate

Solicitud (JSON): { text: string, voice?: string }

Respuesta (200): { audioContent: string(base64 MP3) }

Errores: 400 (texto faltante), 500 (fallo de síntesis o credenciales).

# 5. Pruebas Sugeridas

* wmtsUrl(): valida que la URL contenga capa, fecha y plantilla GoogleMapsCompatible\_Level9.
* defaultUTCDate(): asegura formato YYYY-MM-DD y que sea ≤ hoy-1 (UTC).
* /narrate (mock): con cliente TTS fingido retorna base64; con credenciales reales produce MP3 reproducible.
* UI: cambio de capa/fecha actualiza el mapa y el panel; manejo de error en narración muestra alerta.

# 6. Rendimiento y Confiabilidad

* Respetar caché de GIBS; aplicar debounce al cambiar fecha/capa.
* Prefetch opcional de fechas adyacentes para animación fluida.
* Logging básico y limitación de tasa en /narrate.

# 7. Despliegue y Configuración

* Monolítico: servir frontend y /narrate en el mismo Express (puerto 8787).
* Separado: hosting estático + backend TTS independiente; actualizar endpoint en app.js y habilitar CORS.
* Entorno: GOOGLE\_APPLICATION\_CREDENTIALS apuntando a la cuenta de servicio con TTS habilitado.

# 8. Hoja de Ruta

* I18N (EN/ES UI).
* Animación con export.
* Permalinks con querystring.
* Métricas de uso.