¡Ese es un reto interesante! Para hacerlo, vas a necesitar herramientas que permitan trabajar con archivos comprimidos y manipular datos, pero, **del lado del cliente (en el navegador)**, esto es un poco más complicado. Afortunadamente, podemos usar algunas bibliotecas de JavaScript que hacen posible resolverlo.

Voy a desglosar el proceso en los pasos que mencionaste:

**1. Descargar el archivo .gz**

Esto lo podemos hacer como antes, utilizando fetch para obtener el archivo comprimido desde el enlace.

**2. Descomprimir el archivo .gz**

Para descomprimir un archivo .gz en el navegador, necesitamos una biblioteca como [pako](https://github.com/nodeca/pako), que permite manejar archivos comprimidos en formato gzip.

**3. Transformar el CSV a SQL**

Usaremos JavaScript para:

* Leer el contenido del archivo .csv descomprimido.
* Dividir el contenido en filas y columnas utilizando el separador ;.
* Construir consultas SQL de tipo INSERT INTO.

**Código Completo**

Aquí tienes cómo hacer todo esto:

html

CopiarEditar

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>Procesar Archivo</title>

<script src="https://cdnjs.cloudflare.com/ajax/libs/pako/2.1.0/pako.min.js"></script>

</head>

<body>

<h1>Descargar, Descomprimir y Transformar Archivo</h1>

<button id="processFile">Procesar Archivo</button>

<pre id="output"></pre>

<script>

document.getElementById("processFile").addEventListener("click", async () => {

const outputElement = document.getElementById("output");

outputElement.textContent = "Procesando archivo...";

try {

// Paso 1: Descargar el archivo .gz

const response = await fetch("https://www.gpro.net/gb/GetMarketFile.asp?market=drivers&type=csv");

const compressedData = new Uint8Array(await response.arrayBuffer());

// Paso 2: Descomprimir el archivo .gz

const decompressedData = pako.inflate(compressedData, { to: 'string' });

// Paso 3: Transformar el CSV en consultas SQL

const sqlStatements = csvToSQL(decompressedData);

// Mostrar las consultas SQL generadas

outputElement.textContent = sqlStatements.join("\n\n");

} catch (error) {

outputElement.textContent = `Error: ${error.message}`;

}

});

// Función para transformar CSV a SQL

function csvToSQL(csvData) {

const lines = csvData.split("\n");

const headers = lines[0].split(";"); // Obtener las cabeceras de la primera fila

const tableName = "drivers\_market"; // Nombre de la tabla en la base de datos

const sqlStatements = [];

for (let i = 1; i < lines.length; i++) {

const row = lines[i].split(";");

if (row.length !== headers.length || !row[0]) continue; // Saltar filas vacías

const values = row.map(value => `'${value.replace(/'/g, "''")}'`).join(", ");

const sql = `INSERT INTO ${tableName} (${headers.join(", ")}) VALUES (${values});`;

sqlStatements.push(sql);

}

return sqlStatements;

}

</script>

</body>

</html>

**¿Cómo funciona este código?**

1. **Descarga el archivo comprimido (.gz)**:
   * Usamos fetch para descargar el archivo desde la URL.
   * Convertimos la respuesta en un ArrayBuffer, que es necesario para descomprimirlo.
2. **Descomprime el archivo**:
   * Usamos la biblioteca pako para descomprimir el archivo GZIP y obtener el contenido como texto plano.
3. **Transforma el CSV en SQL**:
   * Divide el contenido descomprimido por líneas (\n) para procesar cada fila.
   * Usa la primera fila para obtener los nombres de las columnas (cabeceras).
   * Genera consultas SQL INSERT INTO para cada fila con datos.
4. **Muestra el resultado**:
   * Genera y muestra las consultas SQL directamente en la página web.

**¿Qué obtendrás?**

Cuando hagas clic en el botón:

1. Descargarás el archivo.
2. Lo descomprimirás directamente en el navegador.
3. Verás las consultas SQL generadas en un <pre> (como texto formateado).

**Requisitos y Consideraciones**

1. **Biblioteca pako**: Se incluye desde un CDN, por lo que no necesitas instalar nada.
2. **Tamaño del archivo**: Esto funciona bien para archivos de tamaño moderado. Para archivos muy grandes, podrías enfrentar problemas de memoria en el navegador.
3. **SQL Personalizado**: Si necesitas ajustar el formato de las consultas SQL (nombres de columnas, tipos de datos, etc.), avísame y lo adaptamos.

¿Quieres que ajuste algo más o lo probamos juntos? 🚀