Realice los siguientes problemas haciendo primero el análisis respectivo. Los problemas deben ser presentados en un repositorio de GitHub:

1. **Crear una Promesa Simple**
   Escribir una promesa que resuelva después de 2 segundos con el mensaje "Promesa cumplida". Luego, consumirla usando `.then()` para imprimir el mensaje en consola.
2. **Manejo de Errores en Promesas**
   Crear una promesa que se rechace si un número es menor a 0.5. Manejar el error con `.catch()` e imprimir el mensaje "Promesa rechazada" en ese caso.
3. **Encadenamiento de Promesas**
   Simular un flujo de pasos donde cada paso tarda 1 segundo en completarse. Usar tres promesas y encadenarlas con `.then()` para que se ejecuten en orden.
4. **Usar `Promise.all()`**
   Crear tres promesas que se resuelvan en diferentes tiempos (por ejemplo, 1s, 2s, 3s). Usar `Promise.all()` para esperar a que todas se resuelvan e imprimir los resultados en un arreglo.
5. **Usar `Promise.race()`**
   Crear dos promesas: una se resuelve en 2 segundos y otra en 5 segundos. Usar `Promise.race()` para determinar cuál se resuelve primero y mostrar el resultado.
6. **Crear una Función Asíncrona Simple**
   Escribir una función asíncrona que devuelva un mensaje "Hola, Mundo" y consumirla con `.then()` o `await`.
7. **Uso de `await` con una Promesa**
   Crea una función asíncrona que utilice `await` para esperar una promesa que resuelve en 3 segundos y luego imprime "Proceso terminado".
8. **Procesar Datos en Serie con `await`**
   Crear una función que procese una lista de nombres (en un arreglo) uno por uno. Usa `await` para simular un retraso de 1 segundo entre cada nombre procesado. Implementar la función delay y el arreglo de nombres.

   ```
   async function procesarNombres(nombres) {
     for (const nombre of nombres) {
           await delay(1000);
           console.log(`Procesado: ${nombre}`);
     }
   }
   ```
9. **`Promise.all` en una Función Asíncrona**
   Escribir una función asíncrona que use `Promise.all` para esperar varias promesas. Por ejemplo, realizar tres búsquedas simuladas (con un retraso aleatorio) y mostrar los resultados juntos.

Proyectos a realizar para aplicar asincronicidad

Puede realizar las siguientes aplicaciones usando consola y prompt, o puede implementar también Html Css

1. Realizar una aplicación que genere miles de archivos de texto de la forma más rápida posible, de tal manera que se aproveche la asincronicidad.
2. Crear una aplicación que genere resúmenes de contenido de texto mediante un LLM (IA), de tal forma que estos resúmenes se obtengan de la IA lo más rápido posible haciendo de esta manera que la app sea eficiente.
3. Crear una app que muestre el clima de Medellín lo más rápido posible. Para esto use dos o tres apis externas para consultar por el clima de Medellín y use Promise.race
4. Crear una app que envíe correo masivo de tal manera que permita enviar miles de correos al mismo tiempo. Como sugerencia puede usar la librería de Nodejs llamada Nodemailer