1. Ir a la página de DIVIO y de allí coger 2-3 apartados para explicar sobre nuestro reto
2. Crear un informe final con los apartados que hay en la segunda página de este documento donde pone “Informe final del proyecto” documento

Informe Final del Proyecto

Dentro de la aplicación desarrollada con Electron, deberá incluirse una sección o ventana accesible desde el menú principal con el título “*Informe Final del Proyecto*”. Este informe debe presentarse de forma clara y ordenada, y debe incluir como mínimo los siguientes apartados:

- Introducción:
    - Explica brevemente el contexto del proyecto, el problema o necesidad detectada en la empresa, y los objetivos principales del desarrollo.
- Detalles del proyecto:
    - Descripción técnica de la aplicación (tecnologías utilizadas, estructura general del código, principales funcionalidades).
    - Capturas o ejemplos de las pantallas principales de la aplicación o del código de algunas cosas de interés, dónde habéis ido más allá de lo que ha pedido el profesorado.
    - Donde y como habéis usado la IA.
    - Dificultades técnicas encontradas y cómo se resolvieron.
- Reflexión Personal:
    - Reflexiona sobre los resultados obtenidos, el trabajo en equipo, lo que se aprendió durante el proceso y posibles mejoras futuras.
- Conclusión

Esto lo implementaremos durante la semana en 2 pestañas diferentes de nuestra aplicación.
Yo crearía un par de documentos con todo lo que queráis meter en cada ventana (DIVIO e Informe Final).

cualquier duda mandarme un whatsapp

Ejemplo:
Informe Final del Proyecto

Introducción
Este proyecto responde a la necesidad detectada en la empresa de disponer de una pequeña aplicación de escritorio multiplataforma para (describir la necesidad concreta: p. ej. gestión, visualización o automatización de X). El objetivo principal ha sido desarrollar una aplicación con Electron que ofrezca una interfaz simple y funcional, con persistencia local y capacidad de despliegue modular.

Detalles del proyecto

Tecnologías utilizadas
- Electron (aplicación de escritorio multiplataforma).
- HTML/CSS/JavaScript para la interfaz (index.html, main.js).
- Node.js para la capa nativa y módulos del side-backend si procede.
- Dependencias gestionadas con package.json / package-lock.json.
- Herramientas de IA (ChatGPT / Copilot) para generación de código, documentación y solución de problemas puntuales durante el desarrollo.

Estructura general del código
- Raíz del repositorio:
  - index.html — ventana principal de la app.
  - main.js — proceso principal de Electron (inicio de ventanas, menús).
  - package.json / package-lock.json — dependencias y scripts.
  - image.png — ejemplo/captura utilizada en la documentación.
  - Carpetas vacías o pendientes de completar: `backend/`, `Reto2Niger2.0/`.
- Sugerencia de organización dentro de la app:
  - /src/renderer — código de la interfaz.
  - /src/main — código de proceso principal (main.js).
  - /assets/docs — documentación y el Informe Final (este archivo).

Funcionalidades principales implementadas
- Interfaz principal con navegación básica (index.html).
- Menú principal configurable desde main.js.
- Pantalla/ventana destinada al "Informe Final del Proyecto" accesible desde el menú.
- Persistencia simple (según alcance: localStorage o fichero JSON).

Capturas / ejemplos de pantallas y código
- Captura incluida en el repo: image.png (usar como ejemplo de pantalla principal).
  - Ruta en el repositorio: image.png
- Archivos relevantes:
  - index.html — ventana principal — archivo: https://github.com/Enriquecebanc/Reto2Niger2.0/blob/main/index.html
  - main.js — configuración del proceso principal — archivo: https://github.com/Enriquecebanc/Reto2Niger2.0/blob/main/main.js

> Nota: las rutas anteriores apuntan a los archivos del repositorio donde se puede ver la implementación actual.

Apartados de DIVIO consultados y cómo se relacionan con el reto
He consultado la página de DIVIO y he seleccionado 3 apartados útiles para contextualizar buenas prácticas de despliegue y operación:

1. Managed Cloud (hosting gestionado)
   - Idea principal tomada: usar un entorno gestionado facilita despliegues reproducibles, escalado y operaciones automatizadas.
   - Relación con el reto: aunque nuestra aplicación es de escritorio (Electron), el concepto de entornos gestionados inspira la separación de responsabilidades (UI local vs. servicios alojados) y la preparación para desplegar componentes backend si fuese necesario.

2. Deployments & CI/CD
   - Idea principal tomada: pipelines automatizados y despliegues reproducibles para garantizar consistencia entre entornos.
   - Relación con el reto: recomendamos añadir un pipeline simple (GitHub Actions o similar) para comprobar linting, tests y empaquetado de la app Electron antes de generar releases.

3. Backups & Security
   - Idea principal tomada: copias periódicas, control de accesos y medidas básicas de seguridad (actualizaciones, configuración segura).
   - Relación con el reto: en caso de usar almacenamiento remoto o sincronización, implementar backups y cifrado; mantener dependencias actualizadas y minimizar permisos en el proceso principal de Electron.

(Se han usado estos apartados como referencia conceptual — para detalles de cada servicio ver: https://www.divio.com/ y la documentación asociada.)

Dónde y cómo se ha usado la IA
- Generación y revisión de fragmentos de código (p. ej. esqueleto de ventanas, menús).
- Redacción y mejora de la documentación y README.
- Soporte en resolución de errores y propuestas de arquitectura (sugerencias de buenas prácticas).
- Generación de textos de la interfaz y pruebas de casos de uso.

Dificultades técnicas y cómo se resolvieron
- Problema: Configuración inicial del proceso principal de Electron y comunicación con renderer.
  - Solución: Estructurar main.js para exponer API segura mediante contextBridge o IPC, limitar acceso directo a Node en el renderer.
- Problema: Integración de recursos estáticos en el paquete final.
  - Solución: Incluir assets en carpetas controladas y configurar el empaquetador (electron-packager/electron-builder) para incluirlos.
- Problema: Falta de experiencia previa con despliegue automatizado.
  - Solución: Propuesta de pipeline básico (lints/test/build) y documentación para desplegar releases.

Reflexión personal
El proyecto ha permitido consolidar conocimientos en desarrollo de aplicaciones de escritorio con Electron, modularización del proyecto y prácticas básicas de despliegue. El trabajo en equipo (si aplica) facilitó dividir tareas: interfaz, lógica y documentación.
Aprendizajes clave:
- Importancia de separar responsabilidades entre proceso principal y renderer.
- Necesidad de automatizar pruebas y empaquetado para mantener calidad.
- Valor de la documentación (README, Informe Final) para facilitar mantenimiento.

Posibles mejoras futuras:
- Añadir un pequeño backend alojado (p. ej. en un Managed Cloud) para sincronización y backups.
- Implementar CI/CD con pruebas automáticas y creación de installers.
- Añadir más funcionalidades solicitadas por el usuario final y mejorar la UX.

Conclusión
Se ha entregado una aplicación base funcional y la documentación asociada. El Informe Final (este documento) debe estar accesible desde el menú principal de la aplicación (elemento: "Informe Final del Proyecto"). Con los pasos recomendados (CI/CD, empaquetado y medidas de seguridad), la aplicación puede evolucionar a una solución más robusta y mantenible.

Archivos y referencias en este repositorio:
- index.html — https://github.com/Enriquecebanc/Reto2Niger2.0/blob/main/index.html
- main.js — https://github.com/Enriquecebanc/Reto2Niger2.0/blob/main/main.js
- image.png — https://github.com/Enriquecebanc/Reto2Niger2.0/blob/main/image.png

Instrucciones rápidas para integrar en Electron:
1. Guardar este archivo como `docs/Informe_Final_del_Proyecto.md` o en `assets/docs/`.
2. Añadir un elemento de menú en main.js que abra una ventana nueva y muestre el Markdown (renderizarlo con una vista web que cargue un convertidor MD->HTML o usar un componente que muestre directamente el Markdown).
3. Opcional: empaquetar la carpeta docs/assets dentro del build para que el informe esté disponible en distribuciones.