**# Prompts para Describir Clasificar los proyectos.**

"Se presenta a continuación un proyecto de desarrollo web compuesto por los siguientes archivos: HTML [CÓDIGO HTML] y JavaScript [CÓDIGO JS]. Esta aplicación tiene como objetivo principal [OBJETIVO PRINCIPAL]. A partir de este código, realiza una evaluación exhaustiva considerando los siguientes 5 puntos:"

1.- Evaluación del proyecto:

"Califica este proyecto de desarrollo web en una escala de 1 a 5, siendo 1 básico y 5 avanzado, y justifica tu respuesta considerando las tecnologías utilizadas, la complejidad del proyecto y la calidad del código."

2.- Descripción del proyecto:

"Redacta una descripción concisa y persuasiva de este proyecto para un reclutador, destacando los desafíos superados, las tecnologías empleadas y el impacto que tuvo."

3.- Completar el archivo README:

"Completa el archivo README.md siguiendo esta estructura [LA PLANTILLA], asegurándote de incluir una introducción clara, una descripción detallada de las funcionalidades, instrucciones de instalación y uso, y una sección de agradecimientos."

4.- Revisar el archivo .gitignore:

"Personaliza el archivo .gitignore para este proyecto específico, teniendo en cuenta las características del proyecto y las mejores prácticas de Git. Siguiendo esta estructura [LA PLANTILLA]

5.- "Resume tus hallazgos clave sobre este proyecto, incluyendo tanto los aspectos positivos como las áreas de mejora. ¿Qué aprendizajes obtuviste durante la evaluación? ¿Qué recomendaciones harías para futuros desarrollos?"

**1.- Evaluación del proyecto**

**Calificación:** 4/5  
**Justificación:**  
El proyecto utiliza tecnologías web esenciales (HTML5, CSS3 y JavaScript) junto con el uso de APIs externas, lo que demuestra una comprensión intermedia de conceptos de desarrollo web. La implementación de funcionalidades como el cambio de tema, un historial de conversiones y la integración con una API de tasas de cambio muestra un diseño bien pensado y orientado al usuario. Sin embargo, no incluye un backend propio ni pruebas automatizadas, lo cual podría mejorar su complejidad y calidad.

**2.- Descripción del proyecto**

Este proyecto, **Conversor de Moneda**, es una aplicación web interactiva diseñada para convertir monedas en tiempo real utilizando tasas oficiales actualizadas a través de una API externa. Desafíos superados incluyen la integración de datos externos con el manejo de errores para garantizar una experiencia de usuario confiable, y la implementación de un diseño adaptable con tema oscuro/claro. El proyecto demuestra un entendimiento sólido del desarrollo web frontend y está enfocado en resolver un problema cotidiano con una solución funcional y estéticamente atractiva.

**3.- Archivo README.md**

# Conversor de Moneda

[Ver el proyecto aquí](URL\_DEL\_PROYECTO)

![Vista del proyecto](URL\_DE\_LA\_IMAGEN)

## Descripción

El \*\*Conversor de Moneda\*\* permite a los usuarios calcular el valor de monedas internacionales en tiempo real. Utilizando tasas actualizadas desde una API externa, proporciona una experiencia fluida y confiable. Este proyecto nació con la intención de explorar la integración de APIs y ofrecer una herramienta útil para viajeros, comerciantes y estudiantes.

## Características

- \*\*Interfaz amigable y responsiva\*\*: Un diseño adaptativo con opción de tema oscuro/claro.

- \*\*Historial de conversiones\*\*: Los usuarios pueden visualizar las conversiones realizadas en la sesión actual.

- \*\*Tasas de cambio actualizadas\*\*: Los valores se obtienen directamente de una API confiable.

- \*\*Selección interactiva de monedas\*\*: Banderas y nombres para una experiencia más intuitiva.

## Tecnologías Utilizadas

- \*\*HTML5\*\*: Estructura semántica para un código limpio y accesible.

- \*\*CSS3\*\*: Diseño responsivo con Flexbox y uso de íconos de Font Awesome.

- \*\*JavaScript\*\*: Lógica de negocio, manejo de eventos y consumo de APIs externas.

- \*\*ExchangeRate API\*\*: Fuente de datos para tasas de cambio actualizadas.

## Instalación

1. Clona el repositorio:

```bash

git clone <https://github.com/usuario/nombre_del_proyecto.git>

1. Abre el archivo index.html en tu navegador preferido para interactuar con la aplicación.

**## Cómo Contribuir**

1. Haz un fork del repositorio.
2. Crea una nueva rama (git checkout -b nombre-de-la-rama).
3. Realiza tus cambios.
4. Envía un pull request describiendo los cambios realizados.  
   ¡Toda colaboración es bienvenida!

**## Licencia**

Este proyecto está distribuido bajo la licencia **MIT**.

**## Autor**

Este proyecto fue desarrollado por **Jairo Cárdenas**.  
Para preguntas o sugerencias:

* Correo electrónico: [jairo.cardenas.dev@gmail.com](mailto:jairo.cardenas.dev@gmail.com)
* GitHub: [@jairo51067](https://github.com/jairo51067)

**4.- Archivo .gitignore**

---

### 4.- Archivo `.gitignore`

```plaintext

# Archivos de configuración local

config.local

.env.local

settings.json

# Logs de npm o yarn

npm-debug.log\*

yarn-debug.log\*

yarn-error.log\*

# Dependencias de Node.js

node\_modules/

# Carpetas de distribución y compilación

dist/

build/

out/

# Archivos temporales

\*.tmp

\*.log

\*.swp

# Archivos generados por editores

.vscode/

.idea/

# Caché

cache/

# Certificados y claves privadas

\*.pem

\*.key

\*.crt

# Archivos de resultados de pruebas

\*.result

\*.coverage

# Archivos de sistema

Thumbs.db

.DS\_Store

# Configuración de Webpack

webpack.config.js

webpack.config.prod.js

**5.- Resumen y hallazgos clave**

**Aspectos positivos:**

* **Interfaz intuitiva y funcionalidad robusta:** La experiencia de usuario es clara, con detalles como las banderas de los países en los menús desplegables.
* **Uso efectivo de tecnologías básicas:** La aplicación combina elementos esenciales (HTML, CSS, JS) con una API para ofrecer un producto funcional.
* **Buena organización del código:** El uso de funciones para modularizar la lógica es una práctica destacable.

**Áreas de mejora:**

* **Pruebas y validaciones:** Incluir pruebas automatizadas y validaciones de formularios más avanzadas.
* **Gestión de errores más robusta:** Mejorar los mensajes de error para casos de fallos en la API.
* **Optimización de rendimiento:** Reducir el tiempo de carga inicial minimizando solicitudes externas y empleando técnicas como lazy loading para recursos adicionales.

**Aprendizajes:**

* La importancia de diseñar una interfaz adaptativa.
* La integración de datos en tiempo real con APIs.
* Implementación del tema oscuro/claro como funcionalidad adicional para mejorar la experiencia del usuario.

**Recomendaciones futuras:**

* Explorar frameworks como React o Vue para escalabilidad.
* Añadir soporte para múltiples idiomas.
* Implementar una base de datos backend para almacenar historiales de usuario de manera persistente.