De 0 a Programador: Crea tu Primer Clon de Excel con HTML, CSS y JavaScript

Índice del Ebook

Introducción

En este tutorial, aprenderás cómo crear un clon básico de **Excel** utilizando **HTML**, **CSS** y **JavaScript**. Este proyecto está diseñado para ser accesible incluso si nunca has programado antes. Te guiaré paso a paso para que puedas crear una hoja de cálculo interactiva, aprenderás conceptos fundamentales de desarrollo web y serás capaz de aplicar tus nuevos conocimientos en proyectos reales.

Autor: Sebastián Piñeiro Madero
Fecha de creación: Enero 2025

Capítulo 1: Preparando el Proyecto

1.1 Requisitos Previos

Antes de comenzar, asegúrate de tener lo siguiente:

- **Editor de código**: Visual Studio Code (VSCode) es una excelente opción, ya que es gratuito y fácil de usar.
- **Navegador web**: Google Chrome, Firefox, o cualquier navegador moderno.
- **Conocimientos básicos de HTML, CSS y JavaScript**: Si nunca has programado, no te preocupes, te explicaré todo desde cero.

1.2 Estructura de Archivos

Este proyecto consta de tres archivos principales:

1. **index.html**: Contiene la estructura básica de la página web (HTML).
2. **styles.css**: Define los estilos y el diseño visual de la página (CSS).
3. **script.js**: Contiene la lógica de interacción de la hoja de cálculo (JavaScript).

La estructura de tu proyecto debe verse así:

```
/clon-excel
    /index.html
    /styles.css
    /script.js
```

1.3 Sintaxis de Comentarios

Los comentarios son útiles para explicar y documentar el código.
En **HTML**, **CSS** y **JavaScript**, los comentarios tienen sintaxis diferente, pero todos sirven para lo mismo: agregar notas sin afectar la ejecución del código.

- **JavaScript**:
  - Una sola línea: `// comentario`
  - Varias líneas: `/* comentario */`
- **HTML**:
  - Una sola o varias líneas: `<!-- comentario -->`
- **CSS**:
  - Una sola línea o varias líneas: `/* comentario */`

Recuerda, los comentarios son solo para ti y otros desarrolladores; no afectan el funcionamiento del código.

Capítulo 2: Creando la Estructura Básica de la Hoja de Cálculo

2.1 Crear el archivo HTML

En este capítulo, comenzaremos con el archivo **index.html**, que establece la estructura básica de nuestra página web. Aquí es donde comenzamos a construir la interfaz de nuestra hoja de cálculo.

```
<!DOCTYPE html>
<html lang="es">
<head>
    <meta charset="UTF-8"> <!-- Definimos la codificación de caracteres, permitiendo usar tildes y caracteres especiales -->
    <meta name="viewport" content="width=device-width, initial-scale=1.0"> <!-- Hace que la página sea responsive -->
    <title>Clon de Excel en JavaScript</title> <!-- Título de la página -->
    <link rel="stylesheet" href="styles.css"> <!-- Vincula el archivo CSS para los estilos -->
</head>
<body>
    <h1>Clon de Excel</h1> <!-- Título principal de la página -->

    <!-- Contenedor de los botones para la interacción con la hoja de cálculo -->
    <div class="controls">
        <button id="saveButton">Guardar</button> <!-- Botón para guardar datos -->
        <button id="loadButton">Cargar</button> <!-- Botón para cargar datos guardados -->
        <button id="exportButton">Exportar CSV</button> <!-- Botón para exportar datos a CSV -->
    </div>

    <!-- Tabla donde se generará la hoja de cálculo -->
    <table id="spreadsheet"></table> <!-- Aquí se generará la tabla dinámicamente -->

    <script src="script.js"></script> <!-- Vincula el archivo JavaScript para la lógica -->
</body>
</html>
```

Explicación del Código HTML:

- **`<!DOCTYPE html>`**: Especifica que estamos utilizando HTML5 para crear nuestra página web.
- **`<html lang="es">`**: Abre el documento HTML y establece que el idioma de la página es español.
- **`<meta charset="UTF-8">`**: Define que la codificación de caracteres es UTF-8, para poder mostrar correctamente caracteres especiales como tildes y la ñ.
- **`<meta name="viewport" content="width=device-width, initial-scale=1.0">`**: Asegura que la página se ajuste correctamente en dispositivos móviles, siendo completamente responsiva.
- **`<title>`**: Establece el título de la página, el cual aparecerá en la pestaña del navegador.
- **`<link rel="stylesheet" href="styles.css">`**: Vincula el archivo de estilos CSS, el cual determinará la apariencia de nuestra hoja de cálculo.
- **`<h1>`**: Este es el título visible en la página, indicándonos que estamos viendo un clon de Excel.
- **`<div class="controls">`**: Contenedor para los botones que permiten al usuario interactuar con la hoja de cálculo (guardar, cargar, exportar).
- **`<table id="spreadsheet"></table>`**: Aquí es donde se generará dinámicamente la tabla de la hoja de cálculo. Aún no contiene filas ni celdas.
- **`<script src="script.js"></script>`**: Vincula el archivo JavaScript que añadirá la lógica a nuestra página.

2.2 Estilos CSS para la Hoja de Cálculo

Ahora vamos a definir los estilos en el archivo **styles.css** para hacer que nuestra hoja de cálculo sea visualmente atractiva y funcional.

```
/* Estilos generales para la página */
body {
    font-family: Arial, sans-serif; /* Establece una fuente moderna y
legible */
    display: flex; /* Utiliza flexbox para organizar los elementos */
    flex-direction: column; /* Coloca los elementos de manera vertical */
    align-items: center; /* Centra los elementos horizontalmente */
    margin: 0; /* Elimina márgenes por defecto */
    padding: 20px; /* Añade un poco de espacio alrededor de los elementos
*/
}


/* Estilo para el título */
h1 {
    font-size: 24px; /* Tamaño de fuente grande para el título */
    margin-bottom: 20px; /* Espacio debajo del título */
}
```

```css
/* Estilo para los botones */
.controls {
    margin-bottom: 20px; /* Espacio entre los botones y la tabla */
}

button {
    padding: 10px 20px; /* Espaciado dentro del botón */
    margin: 5px; /* Espaciado entre los botones */
    font-size: 16px; /* Tamaño de texto en los botones */
    cursor: pointer; /* Cambia el cursor cuando el mouse pasa sobre el
botón */
    border: none; /* Elimina el borde del botón */
    background-color: #4CAF50; /* Color de fondo verde */
    color: white; /* Color de texto blanco */
    border-radius: 5px; /* Bordes redondeados */
}

button:hover {
    background-color: #45a049; /* Cambia el color del botón al pasar el
mouse */
}

/* Estilo para la tabla */
table {
    border-collapse: collapse; /* Elimina los bordes dobles entre celdas
*/
    width: 80%; /* Establece el ancho de la tabla al 80% de la pantalla
*/
}

/* Estilo para las celdas */
td {
    border: 1px solid #ccc; /* Borde gris claro */
    padding: 10px; /* Espaciado dentro de las celdas */
    text-align: center; /* Centra el texto dentro de las celdas */
    width: 100px; /* Ancho fijo para cada celda */
    height: 40px; /* Altura fija para cada celda */
}

td:empty {
    background-color: #f9f9f9; /* Fondo gris claro para celdas vacías */
}
```

Explicación del Código CSS:

- `body`: Establece la fuente, usa **Flexbox** para organizar los elementos, centra todo y les da algo de espacio con los márgenes y padding.
- `h1`: Se configura el tamaño y margen para el título principal de la página.
- `.controls`: Proporciona un margen entre los botones y la tabla para darle un poco de espacio visual.
- `button`: Define los estilos de los botones, como el padding, el color de fondo, el color del texto, el borde redondeado y un cambio de color al pasar el ratón sobre ellos.
- **`table
- `**: Configura la tabla para que sus bordes no se dupliquen y establece el ancho de la tabla.

- `td`: Define los estilos de las celdas de la tabla, como el borde, el padding, la alineación del texto, el tamaño fijo de cada celda y el color de fondo de las celdas vacías.

Resumen del Capítulo 2

2.1 Crear el archivo HTML

Este capítulo cubre cómo estructurar el archivo HTML de la página, incluyendo el título, los botones para interactuar con la hoja de cálculo y una tabla vacía donde se generará la cuadrícula de celdas.

2.2 Estilos CSS para la Hoja de Cálculo

En esta sección, se definen los estilos básicos para la página, el título, los botones de interacción y la tabla, haciendo que la hoja de cálculo tenga una apariencia limpia y funcional.

Capítulo 3: Agregando Funcionalidad Básica con JavaScript

3.1 Crear la Cuadrícula Dinámica

Ahora vamos a agregar la funcionalidad utilizando **JavaScript** para crear la cuadrícula interactiva. El código siguiente creará una tabla de 10x10 de celdas editables.

```
document.addEventListener('DOMContentLoaded', () => {
    const table = document.getElementById('spreadsheet'); // 1. Obtenemos
el elemento de la tabla
    const rows = 10; // 2. Número de filas
    const cols = 10; // 3. Número de columnas

    // 4. Creamos la tabla de manera dinámica
    for (let i = 0; i < rows; i++) {
        const row = document.createElement('tr'); // 5. Creamos una nueva
fila
        for (let j = 0; j < cols; j++) {
            const cell = document.createElement('td'); // 6. Creamos una
nueva celda
            cell.contentEditable = true; // 7. Hacemos la celda editable
            row.appendChild(cell); // 8. Añadimos la celda a la fila
        }
        table.appendChild(row); // 9. Añadimos la fila a la tabla
    }
});
```

Explicación del Código JavaScript:

1. **`document.addEventListener('DOMContentLoaded', () => {...})`**: Asegura que el código se ejecute solo después de que todo el contenido del DOM haya sido cargado completamente.
2. **`const table = document.getElementById('spreadsheet')`**: Accede al elemento de la tabla con el ID **spreadsheet** que hemos definido en el archivo HTML.
3. **`const rows = 10; const cols = 10;`**: Establecemos el número de filas y columnas que tendrá nuestra hoja de cálculo (en este caso, una cuadrícula de 10x10).
4. **`for (let i = 0; i < rows; i++)`**: Creamos un bucle para generar las filas de la tabla.
5. **`const row = document.createElement('tr')`**: Crea una nueva fila para cada iteración del bucle.
6. **`const cell = document.createElement('td')`**: Dentro de cada fila, creamos una celda (elemento).
7. **`cell.contentEditable = true`**: Hacemos que la celda sea editable, permitiendo que el usuario ingrese datos en ella.
8. **`row.appendChild(cell)`**: Añade la celda creada a la fila.
9. **`table.appendChild(row)`**: Finalmente, añade la fila completa a la tabla.

3.2 Guardar y Cargar los Datos

Agregamos la funcionalidad de guardar y cargar los datos de la hoja de cálculo usando **localStorage** para almacenar la información en el navegador.

```
document.getElementById('saveButton').addEventListener('click', () => {
    const table = document.getElementById('spreadsheet');
    let data = [];

    for (let i = 0; i < table.rows.length; i++) {
        let row = [];
        for (let j = 0; j < table.rows[i].cells.length; j++) {
            row.push(table.rows[i].cells[j].innerText); // 1. Guardamos
el contenido de cada celda
        }
        data.push(row); // 2. Añadimos la fila al array de datos
    }

    localStorage.setItem('spreadsheetData', JSON.stringify(data)); // 3.
Guardamos los datos en localStorage
});

document.getElementById('loadButton').addEventListener('click', () => {
    const data = JSON.parse(localStorage.getItem('spreadsheetData')); //
1.Cargamos los datos guardados

    if (data) {
        const table = document.getElementById('spreadsheet');
        for (let i = 0; i < table.rows.length; i++) {
            for (let j = 0; j < table.rows[i].cells.length; j++) {
                table.rows[i].cells[j].innerText = data[i][j]; // 2.
Restauramos el contenido de las celdas
            }
        }
    }
});
```

Explicación del Código JavaScript (Guardar y Cargar):

Guardar Datos:

1. **`const table = document.getElementById('spreadsheet')`**: Obtenemos la referencia de la tabla.
2. **`let data = []`**: Inicializamos un array vacío donde almacenaremos los datos de cada celda.
3. **`row.push(table.rows[i].cells[j].innerText)`**: Recorremos cada celda y extraemos su contenido de texto.
4. **`data.push(row)`**: Añadimos cada fila de celdas al array `data`.
5. **`localStorage.setItem('spreadsheetData', JSON.stringify(data))`**: Guardamos los datos del array en el almacenamiento local (localStorage) en formato JSON.

Cargar Datos:

1. **`const data = JSON.parse(localStorage.getItem('spreadsheetData'))`**: Obtenemos los datos guardados en `localStorage` y los parseamos de JSON a un array.
2. **`table.rows[i].cells[j].innerText = data[i][j]`**: Restauramos el contenido de las celdas con los datos previamente guardados.

3.3 Exportar Datos a CSV

Ahora vamos a implementar la funcionalidad de exportar los datos a un archivo **CSV** que el usuario podrá descargar.

```
document.getElementById('exportButton').addEventListener('click', () => {
    const table = document.getElementById('spreadsheet');
    let csv = '';

    for (let i = 0; i < table.rows.length; i++) {
        let row = [];
        for (let j = 0; j < table.rows[i].cells.length; j++) {
            row.push('"' + table.rows[i].cells[j].innerText + '"'); // 1.
Añadimos cada celda con comillas para el formato CSV
        }
        csv += row.join(',') + '\n'; // 2. Unimos las celdas de la fila
con comas y añadimos un salto de línea
    }

    const hiddenElement = document.createElement('a'); // 3. Creamos un
enlace de descarga
    hiddenElement.href = 'data:text/csv;charset=utf-8,' + encodeURI(csv);
// 4. Codificamos el CSV como URI
    hiddenElement.target = '_blank'; // 5. Abrimos el enlace en una nueva
pestaña
    hiddenElement.download = 'hoja_de_calculo.csv'; // 6. Establecemos el
nombre del archivo
    hiddenElement.click(); // 7. Simulamos el clic para descargar el
archivo CSV
});
```

Explicación del Código JavaScript (Exportar a CSV):

1. **`let csv = '';`**: Inicializamos una cadena vacía donde almacenaremos los datos en formato CSV.
2. **`row.push('"' + table.rows[i].cells[j].innerText + '"');`**: Añadimos cada celda con comillas dobles para garantizar el formato CSV correcto.
3. **`csv += row.join(',') + '\n';`**: Unimos todas las celdas de una fila con una coma y añadimos un salto de línea.
4. **`const hiddenElement = document.createElement('a');`**: Creamos un enlace de descarga invisible.

5. `hiddenElement.href = 'data:text/csv;charset=utf-8,' + encodeURI(csv);`: Establecemos el enlace con los datos CSV como una URI codificada.
6. `hiddenElement.download = 'hoja_de_calculo.csv';`: Definimos el nombre del archivo descargado.
7. `hiddenElement.click();`: Simulamos el clic en el enlace para activar la descarga del archivo CSV.

Resumen del Capítulo 3

3.1 Crear la Cuadrícula Dinámica

Se enseña cómo usar JavaScript para generar dinámicamente una tabla de 10x10 con celdas editables. Cada celda es editable para que el usuario pueda ingresar datos.

3.2 Guardar y Cargar los Datos

Se explica cómo usar `localStorage` para guardar y cargar los datos de la hoja de cálculo. El contenido de las celdas se guarda y puede ser restaurado cuando el usuario regresa a la página.

3.3 Exportar Datos a CSV

Se muestra cómo implementar la funcionalidad para exportar los datos de la hoja de cálculo a un archivo CSV que el usuario puede descargar.

Resultado Final del codigo Script.js:

```
document.addEventListener('DOMContentLoaded', () => {

    // Crear la tabla 10x10

    const table = document.getElementById('spreadsheet');

    const rows = 10;

    const cols = 10;
```

```
// Generar celdas
for (let i = 0; i < rows; i++) {
    const row = document.createElement('tr');
    for (let j = 0; j < cols; j++) {
        const cell = document.createElement('td');
        cell.contentEditable = true;
        row.appendChild(cell);
    }
    table.appendChild(row);
}

// Guardar los datos
document.getElementById('saveButton').addEventListener('click', () => {
    const data = [];
    for (let i = 0; i < table.rows.length; i++) {
        const row = [];
        for (let j = 0; j < table.rows[i].cells.length; j++) {
            row.push(table.rows[i].cells[j].innerText);
        }
        data.push(row);
    }

    // Guardamos los datos en localStorage
```

```
    localStorage.setItem('spreadsheetData', JSON.stringify(data));
});

// Cargar los datos guardados
document.getElementById('loadButton').addEventListener('click', () => {
    const data = JSON.parse(localStorage.getItem('spreadsheetData'));

    if (data) {
        for (let i = 0; i < table.rows.length; i++) {
            for (let j = 0; j < table.rows[i].cells.length; j++) {
                table.rows[i].cells[j].innerText = data[i][j];
            }
        }
    }
});

// Exportar a CSV
document.getElementById('exportButton').addEventListener('click', () => {
    let csv = '';
    for (let i = 0; i < table.rows.length; i++) {
        const row = [];
        for (let j = 0; j < table.rows[i].cells.length; j++) {
            row.push('"' + table.rows[i].cells[j].innerText + '"');
```

```
            }
            csv += row.join(',') + '\n';
        }
        // Crear un enlace para la descarga del archivo CSV
        const hiddenElement = document.createElement('a');
        hiddenElement.href = 'data:text/csv;charset=utf-8,' + encodeURI(csv);
        hiddenElement.target = '_blank';
        hiddenElement.download = 'spreadsheet.csv';
        hiddenElement.click();
    });
});
```

Capítulo 4: Alojar y Desplegar el Proyecto con Git y GitHub Pages

Ahora que has creado tu clon de Excel básico utilizando **HTML**, **CSS** y **JavaScript**, es hora de llevar tu proyecto al siguiente nivel: **alojarlo en GitHub** y **desplegarlo en GitHub Pages**. GitHub Pages es una excelente herramienta que te permite **publicar sitios web estáticos** (como el tuyo) directamente desde un repositorio en GitHub, sin necesidad de un servidor propio.

En este capítulo, aprenderás cómo:

- **Inicializar tu proyecto con Git** para realizar un seguimiento de los cambios.
- **Subir tu proyecto a GitHub**.
- **Configurar GitHub Pages** para que tu clon de Excel sea accesible desde cualquier navegador.

4.1. Inicializando Git en tu Proyecto

Antes de poder subir tu proyecto a GitHub, necesitas **inicializar Git en tu carpeta de proyecto** para que pueda realizar un seguimiento de todos los cambios que realices.

1. **Abre la terminal (o línea de comandos)** en tu computadora.
2. Navega a la carpeta que contiene tu proyecto `clon-excel`. Si estás usando **VSCode**, puedes abrir la terminal directamente desde el editor con `Ctrl + ñ` (en Windows) o `Cmd + ñ` (en macOS).

   ```
   bash
   Copiar
   cd /ruta/a/tu/proyecto/clon-excel
   ```

   Este comando te llevará a la carpeta del proyecto.

3. **Inicializa el repositorio Git**:

   ```
   bash
   Copiar
   git init
   ```

   Este comando convierte tu carpeta en un repositorio Git. A partir de ahora, Git podrá realizar un seguimiento de todos los archivos y cambios dentro de esa carpeta.

4.2. Subiendo tu Proyecto a GitHub

Ahora que tienes Git configurado en tu máquina local, el siguiente paso es **subir tu proyecto a GitHub** para que sea accesible desde cualquier lugar. A continuación, te explicaré cómo crear un repositorio en GitHub y vincularlo con tu proyecto local.

1. **Crea un nuevo repositorio en GitHub**:
   - Ve a GitHub y crea una cuenta si aún no tienes una.
   - Una vez que inicies sesión, haz clic en el ícono de **"+"** en la esquina superior derecha y selecciona **"New repository"**.
   - Dale un nombre a tu repositorio, por ejemplo: `clon-excel`.
   - No es necesario inicializar el repositorio con un `README.md` ni con una licencia, ya que lo haremos desde tu máquina local.
2. **Conecta tu repositorio local con GitHub**: Después de crear el repositorio, GitHub te proporcionará una URL como esta:

   ```
   arduino
   Copiar
   https://github.com/tu_usuario/clon-excel.git
   ```

   Ahora, vuelve a tu terminal y vincula tu repositorio local con este repositorio remoto utilizando el siguiente comando:

   ```
   bash
   Copiar
   git remote add origin https://github.com/tu_usuario/clon-excel.git
   ```

Con este comando, Git sabe a qué repositorio remoto debe enviar tus archivos.

3. **Sube los archivos a GitHub**: Primero, agrega todos los archivos a Git para que los comience a rastrear:

   ```
   bash
   Copiar
   git add .
   ```

   Luego, haz un **commit** de los archivos con el siguiente mensaje:

   ```
   bash
   Copiar
   git commit -m "Primer commit: Proyecto Clon de Excel"
   ```

   Finalmente, sube los archivos a GitHub con este comando:

   ```
   bash
   Copiar
   git push -u origin master
   ```

   Si usas `main` como rama principal en lugar de `master`, usa `main` en lugar de `master` en el comando.

4.3. Desplegar en GitHub Pages

Una vez que tu código esté en GitHub, puedes hacer que tu clon de Excel sea accesible a través de **GitHub Pages**. Esto es muy fácil de configurar y te permitirá ver tu proyecto en vivo desde cualquier navegador.

1. **Accede a la configuración de tu repositorio**: Dirígete a la página de tu repositorio en GitHub (por ejemplo, `https://github.com/tu_usuario/clon-excel`) y haz clic en la pestaña **"Settings"** en la parte superior.
2. **Habilitar GitHub Pages**:
   - Desplázate hacia abajo hasta la sección de **GitHub Pages**.
   - En el campo **Source**, selecciona la rama `master` (o `main` si usas esa rama) y haz clic en **Save**.
3. **Accede a tu página web**: Después de unos minutos, GitHub generará una URL para tu sitio en vivo. La URL será algo similar a:

   ```
   arduino
   Copiar
   https://tu_usuario.github.io/clon-excel/
   ```

   Ahora podrás acceder a tu clon de Excel desde cualquier navegador.

4.4. Mantener tu Proyecto Actualizado

A medida que sigas desarrollando tu clon de Excel, es importante que puedas **mantener tu proyecto actualizado** en GitHub. Para hacer esto, sigue estos pasos cada vez que realices cambios en tu proyecto:

1. **Realiza cambios en tu código**. Esto puede incluir mejoras, corrección de errores o nuevas funcionalidades.
2. **Agrega los archivos modificados a Git**:

   ```
   bash
   Copiar
   git add .
   ```

3. **Haz un commit de los cambios**:

   ```
   bash
   Copiar
   git commit -m "Descripción de los cambios realizados"
   ```

4. **Sube los cambios a GitHub**:

   ```
   bash
   Copiar
   git push origin master
   ```

Git registrará estos cambios y GitHub los reflejará en tu repositorio remoto, así como en tu página de GitHub Pages.

Resumen del Capítulo 4

- **Inicializar Git**: Aprendiste a inicializar un repositorio Git en tu máquina local para realizar un seguimiento de los cambios en tu proyecto.
- **Subir tu Proyecto a GitHub**: Subiste tu clon de Excel a un repositorio en GitHub, haciendo que tu código esté disponible en línea.
- **Desplegar en GitHub Pages**: Configuraste **GitHub Pages** para que tu clon de Excel fuera accesible a través de una URL pública.
- **Mantener tu Proyecto Actualizado**: Aprendiste cómo mantener tu proyecto actualizado con Git, subiendo cambios a GitHub cada vez que realices mejoras.

Con estos pasos, ahora tienes tu **clon de Excel** disponible en la web, accesible desde cualquier lugar. Además, con Git y GitHub, tienes una forma poderosa de controlar y compartir tu código, facilitando la colaboración y el despliegue de nuevas características.

Espero que este capítulo te haya sido útil y te haya mostrado cómo **alojar y desplegar** tu proyecto con **Git** y **GitHub Pages**. ¡Ahora es el momento de compartir tu clon de Excel con el mundo!

Capítulo 5: Mejoras Futuras

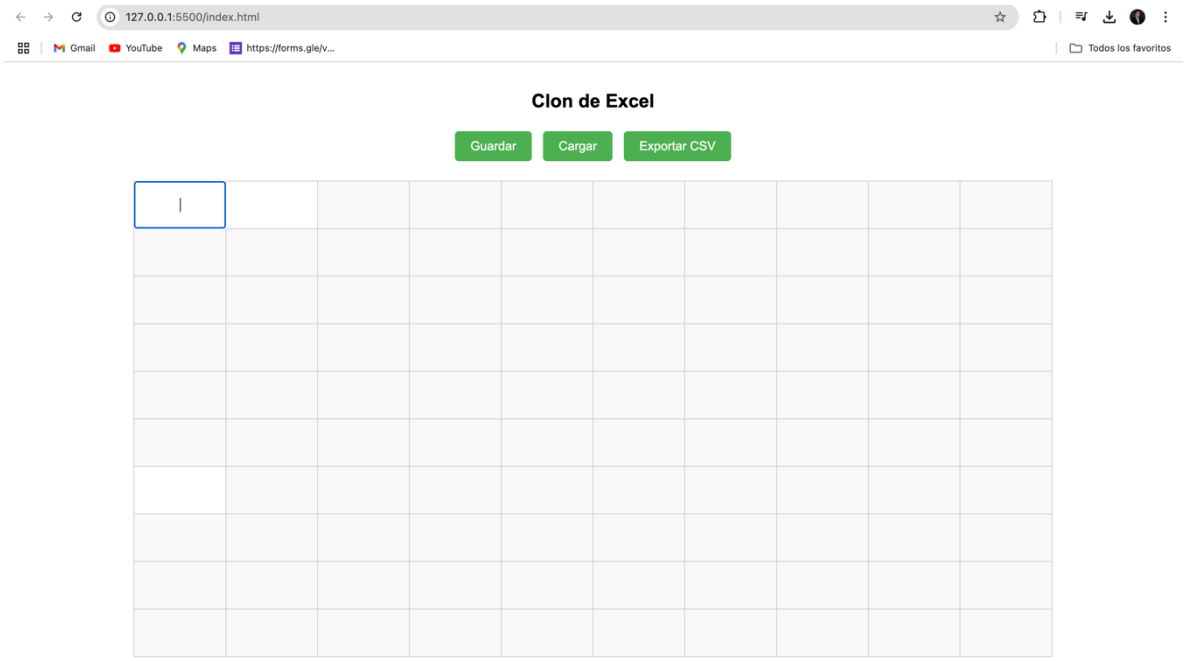
Aquí hay algunas ideas para mejorar tu clon de Excel:

- **Agregar Fórmulas**: Implementa cálculos automáticos cuando el usuario ingrese una fórmula.
- **Interfaz de Usuario Mejorada**: Agrega botones para cambiar el color de las celdas, el tamaño de la fuente, etc.
- **Autoguardado**: Guarda automáticamente los datos en **localStorage** después de ciertos intervalos de tiempo.
- **Exportar a otros formatos**: Además del CSV, permite exportar a Excel o PDF.

Capítulo 6: Protección Legal y Licencia

Conclusión

¡Felicidades! Has creado tu propio clon de Excel básico utilizando **HTML**, **CSS** y **JavaScript**. Ahora tienes las bases para seguir mejorando este proyecto y aplicar tus nuevos conocimientos a otros proyectos web.


127.0.0.1:5500/index.html
Gmail
YouTube
Maps
https://forms.gle/v...
Todos los favoritos
Clon de Excel
Guardar
Cargar
Exportar CSV