DOCUMENTACIÓN DE NAVEGACIÓN

Luis Alberto Padilla González

28 DE MAYO DE 2025

CONEXIÓN A APIS

Índice:

Explicación general:

La idea original de la API es que fuese escalable y adaptable para diferentes proyectos, ajenos al proyecto final. Puedes obtener la información tanto de luchadores como de canciones más populares que han sido parte de wwe, desde luchadores hasta eventos especiales. Como curiosidad el programa realizado en clase “MedievalAPI” es capaz de reconocer la API en cuestión y usar su información. (Aunque siendo sinceros es porque en etapas iniciales fue una base respecto al proyecto)


```
{
    "id": 38,
    "nombres": {
        "primer_nombre": "Cody",
        "segundo_nombre": "Rhodes"
    },
    "rango":"superestrella",
    "poder": 95,
    "descripcion": "La pesadilla americana, el hombre que acabara su historia sin importar que ",
    "precio": 20,
    "imagen": "https://raw.githubusercontent.com/LuisPadillaG/wwe-imagenes/refs/heads/main/superstars/cody.png"
}
```


Cada luchador cuenta con un precio, descripción, rango en la cartelera de wwe, poder, y un identificable

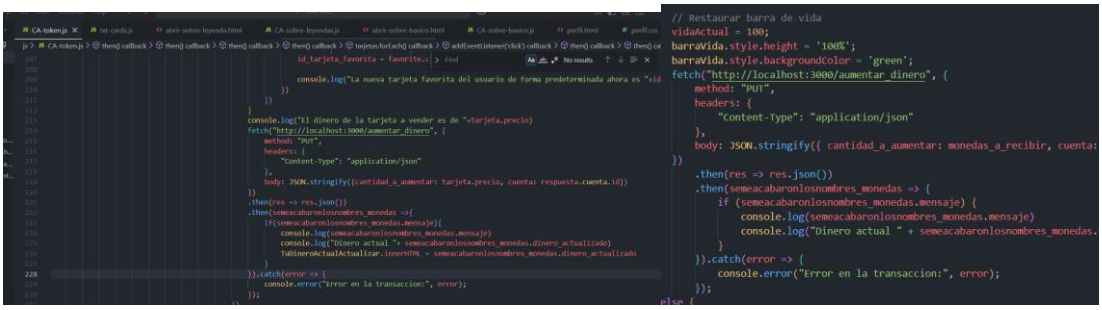

```
{
    "id": 5,
    "nombre": "Kingdom",
    "arista_principal":"DownStait",
    "enlace_cancion": "https://github.com/LuisPadillaG/wwe-imagenes/raw/refs/heads/main/canciones_wwe/Kingdom.mp3",
    "portada": "https://raw.githubusercontent.com/LuisPadillaG/wwe-imagenes/refs/heads/main/canciones_wwe/codycool.jpg"
}
```


Las canciones contienen la información más básica de la canción.

Las imágenes y audios se encuentran en un repositorio personal de github

https://github.com/LuisPadillaG/wwe-imagenes.git

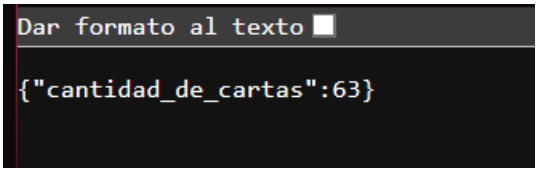
Como mencioné antes, cada acción esta divida para su reciclaje en la aplicación, por ejemplo, el aumento de las monedas se usa para el videojuego, así como para la venta de cartas



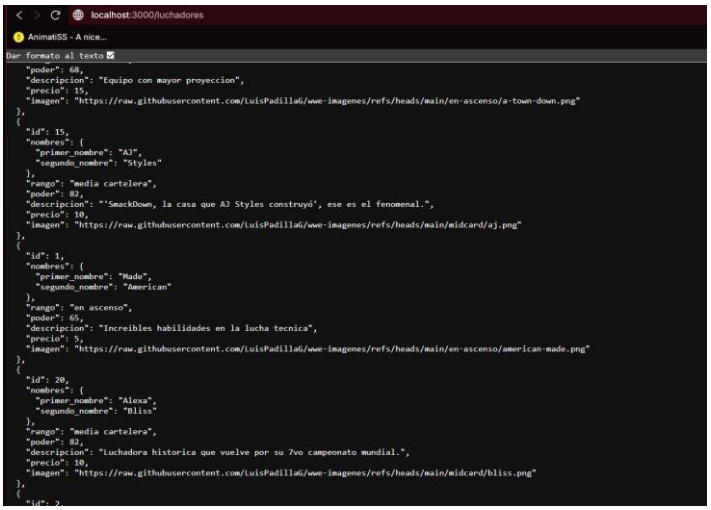
Explicaciones individuales del método GET:

“/”: Te da solamente cuantas cartas tiene el juego

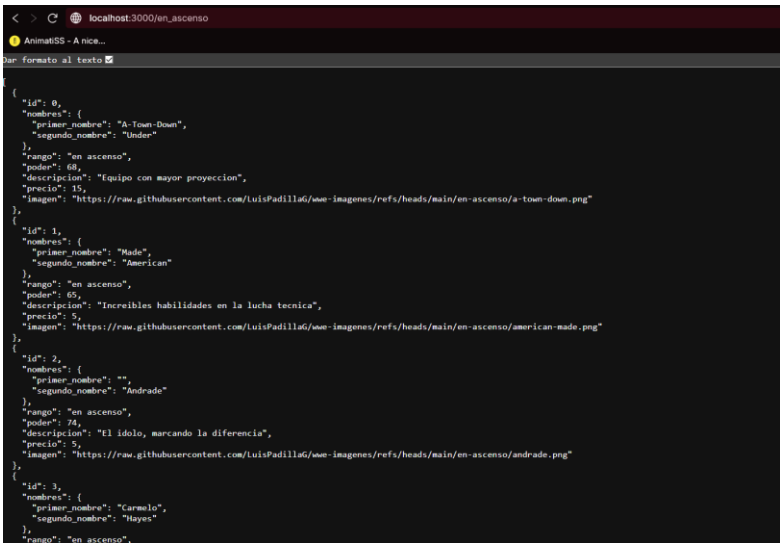
```
if (request.url == "/") { //esto nada mas le dara la cantidad de tarjetas que tiene la api
    var objeto_cantidad = {
        "cantidad_de_cartas": arreglo_tarjetas.length
    }
    response.statusCode = 200;
    response.setHeader("Content-Type", "application/json");
    response.end(JSON.stringify(objeto_cantidad));
} else if (request.url == "/en_ascenso") {
```



“/luchadores”: Da todos los luchadores de la api (enlace no usado en el juego)



“/en_ascenso”: Da todos los luchadores con rango en ascenso



De la misma manera funciona los get de "media_cartelera", "superestrella", "leyenda"

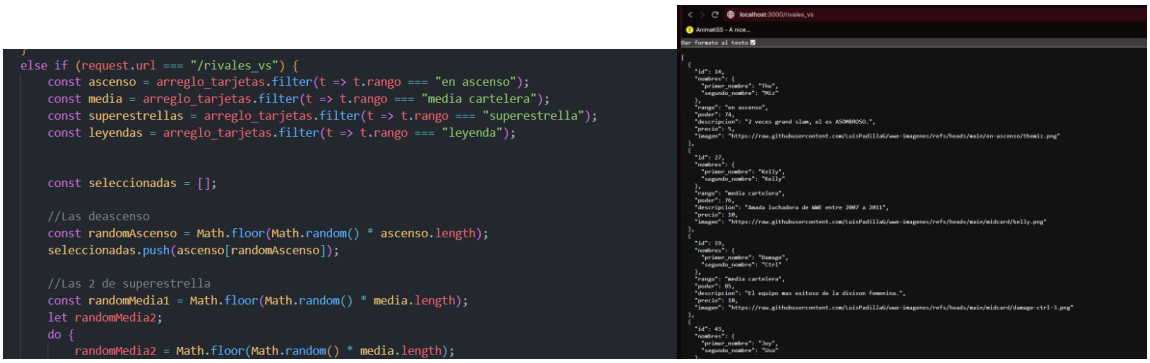
"/música": Da todos los json de las canciones

```
} else if (request.url == "/musica") {
    const musica_total = arreglo_musica
    response.statusCode = 200;
    response.setHeader("Content-Type", "application/json")
    response.end(JSON.stringify(musica_total))
}
```

"/música/[id]": Da la información de una sola canción (peticion no usada en el juego)

```
    response.end(JSON.stringify(musica_total))
}
else if (request.url.startsWith("/musica/")) {
    const id = parseInt(request.url.split("/")[2]); //Extrae

    const cancion = arreglo_musica.find(c => c.id === id);

    if (cancion) {
        response.statusCode = 200;
        response.setHeader("Content-Type", "application/json
        response.end(JSON.stringify(cancion));
    } else {
        response.statusCode = 404;
        response.setHeader("Content-Type", "application/json
        response.end(JSON.stringify({ error: "Todavia no con
    }
}
```

"/rivales_vs": Toma un arreglo de 6 luchadores para el modo de juego de modo vs. Función específica para el videojuego. Da 1 carta en ascenso, 2 media cartelera, 2 superestrellas y 1 leyenda



```
else if (request.url === "/rivales_vs") {
    const ascenso = arreglo_tarjetas.filter(t => t.rango === "en ascenso");
    const media = arreglo_tarjetas.filter(t => t.rango === "media cartelera");
    const superestrellas = arreglo_tarjetas.filter(t => t.rango === "superestrella");
    const leyendas = arreglo_tarjetas.filter(t => t.rango === "leyenda");


    const seleccionadas = [];

    //Las deascenso
    const randomAscenso = Math.floor(Math.random() * ascenso.length);
    seleccionadas.push(ascenso[randomAscenso]);

    //Las 2 de superestrella
    const randomMedia1 = Math.floor(Math.random() * media.length);
    let randomMedia2;
    do {
        randomMedia2 = Math.floor(Math.random() * media.length);
```

“datos_protegidos”: Sistema de otorgamiento de información a la página sobre cuentas de algún usuario

Explicaciones puntuales del método POST:

Vamos a ignorar funcionamiento de cuentas por obviedad para concentrarnos en funciones del juego

“/cartas_ordenadas” toma un arreglo de números (ids) otorgado por el usuario, para investigar en los archivos del api (arreglo_tarjetas es donde guardamos la información de los luchadores) y en base al poder que va de 50 a 99 entre las tarjetas, regresar la información de las tarjetas manera ordenada

```
    })
} else if (request.url === "/cartas_ordenadas") {
    let body = "";
    request.on("data", chunk => body += chunk);
    request.on("end", () => {
        const { ids } = JSON.parse(body);

        const cartasFiltradas = arreglo_tarjetas.filter(c => ids.includes(c.id));

        cartasFiltradas.sort((a, b) => b.poder - a.poder);

        response.statusCode = 200;
        response.setHeader("Content-Type", "application/json");
        response.end(JSON.stringify(cartasFiltradas));
    });
} else if (request.url === "/crear_cuenta") {
```

“**/meter_carta_a_cuenta**” en base al token (por ser una acción delicada) introduce nuevos números a las id de la colección de carta que tiene el usuario, verificando que no existe ya en su colección. Esto está pensado en el momento de abrir sobres

```
} else if (request.url === "/meter_carta_a_cuenta") {
    let body = "";
    request.on("data", chunk => {
        body += chunk;
    });
    request.on("end", () => {
        if (!authHeader) {
            response.statusCode = 401;
            response.setHeader('Content-Type', 'application/json');
            response.end(JSON.stringify({
                "mensaje": "Token no proporcionado"
            }));
        } else {
            jwt.verify(authHeader, llaveSecretaFirmarJWT, (err, decoded) => {
                if (err) {
                    response.statusCode = 401;
                    response.setHeader('Content-Type', 'application/json');
                    response.end(JSON.stringify({
                        "mensaje": "Token inválido o expirado"
                    }));
                } else {
                    console.log("Token decodificado:", decoded);
```

ONSOLE OUTPUT TERMINAL PORTS

Explicaciones puntuales para el método PUT

“/aumentar_dinero” y “/restar_dinero” Como su nombre lo indica, sirve para transacciones de dinero en la cuenta modificando el valor de las monedas de la cuenta

```
case "PUT":
    if (request.url === "/aumentar_dinero") {
        let body = "";
        request.on("data", chunk => {
            body += chunk;
        });
        console.log("si se ejecuto el put")
        request.on("end", () => {
            const data = JSON.parse(body);
            const { cantidad_a_aumentar, cuenta } = data;
            console.log("Se han pedido " + cantidad_a_aumentar + " monedas")
            console.log("Se va a transferir a la cuenta con id " + cuenta)
            //Nota para mi: si quiero transferir un valor se puede hacer con el signo de interrogacion,
            conexion_bd.query("UPDATE cuentas SET monedas = monedas + ? WHERE id = ?", [cantidad_a_aume
                if (err) {
                    console.error("Error al aumentar las monedas:", err);
                    response.statusCode = 500;
                    response.setHeader("Content-Type", "application/json");
                    response.end(JSON.stringify({ mensaje: "Error al aumentar las monedas" }));
                } else {
                    conexion_bd.query("SELECT monedas FROM cuentas WHERE id = ?", [cuenta], (err2, resu
                        if (err2) {
                            console.error("Error al obtener el las monedas actualizado:", err2)
```

```
} else if (request.url == "/restar_dinero") {
    let body = "";
    request.on("data", chunk => {
        body += chunk;
    });
    request.on("end", () => {
        const data = JSON.parse(body);
        const { cantidad_a_restar, cuenta } = data;
        console.log("Se restaran " + cantidad_a_restar + " monedas de la cuenta")
        console.log("Se restaran esas monedas a la cuenta con id " + cuenta)
        //Nota para mi: si quiero transferir un valor se puede hacer con el signo de interrogacion, de igualforma se le pu
        conexion_bd.query("UPDATE cuentas SET monedas = monedas - ? WHERE id = ?", [cantidad_a_restar, cuenta], (err, resu
            if (err) {
                console.error("Error al restar las monedas:", err);
                response.statusCode = 500;
                response.setHeader("Content-Type", "application/json");
                response.end(JSON.stringify({ mensaje: "Error al restar las monedas" }));
            } else {
                conexion_bd.query("SELECT monedas FROM cuentas WHERE id = ?", [cuenta], (err2, resultado2) => {
                    if (err2) {
                        console.error("Error al obtener el las monedas actualizado:", err2)
                        response.statusCode = 500;
                        response.setHeader("Content-Type", "application/json");
```

“/cambiar_puntaje_maximo” En dado caso de haber realizado un nuevo record en el juego, se realiza una petición actualizando el valor de la propiedad en la cuenta

```
    })
} else if (request.url == "/cambiar_puntaje_maximo") {
    let body = "";
    request.on("data", chunk => {
        body += chunk;
    });
    request.on("end", () => {
        const data = JSON.parse(body);
        const { puntaje_maximo, idCuenta } = data;
        console.log("El puntaje  " + puntaje_maximo + " sera el nuevo puntaje")
        console.log("La cuenta con el id " + idCuenta + " sera la del nuevo puntaje")
        /** */
        conexion_bd.query("UPDATE cuentas SET puntaje_maximo = ? WHERE id = ?", [puntaje_maximo, idCuenta], (err, resultado) => {
            if (err) {
                console.error("Error con la base de datos", err);
                response.statusCode = 500;
                response.setHeader("Content-Type", "application/json");
                response.end(JSON.stringify({ mensaje: "Error al actualizar" }));
                console.log("Se ejecuto esta")
            } else {
                response.statusCode = 200;
                response.setHeader("Content-Type", "application/json");
                response.end(JSON.stringify({mensaje: "Se ha cambiado correctamente"}));
                console.log("Se ejecuto el bueno")
            }
CONSOLE  OUTPUT  TERMINAL  PORTS                                                    node
```

“/acceso_compra” Las cuentas tienen un valor en ellas que verifica si se está realizando algún tipo de transacción, que se cambia en cuanto se realiza la compra. Este es un método de seguridad improvisado para evitar que pueda meterse a los enlaces de abridores de sobres o compras sin haber comprado el producto realmente. También puede servir para meter micro transacciones (ya eso es muy a futuro).

Si el valor es 1, es porque se encuentra realizando una compra, si el valor es 0, es porque no se encuentra realizando una compra

	realizando_compra
0	0
0	0
0	0
0	0
0	0

```
} else if (request.url == "/acceso_compra") {
    let body = "";
    request.on("data", chunk => {
        body += chunk;
    });
    request.on("end", () => {
        if (!authHeader) {
            response.statusCode = 401;
            response.setHeader('Content-Type', 'application/json');
            response.end(JSON.stringify({
                "mensaje": "Token no proporcionado"
            }));
        } else {
            jwt.verify(authHeader, llaveSecretaFirmarJWT, (err, decoded) => {
                if (err) {
                    response.statusCode = 401;
                    response.setHeader('Content-Type', 'application/json');
                    response.end(JSON.stringify({
                        "mensaje": "Token inválido o expirado"
                    }));
                } else {
                    console.log("Token decodificado:", decoded);
```

“/cambiar_carta_favorita” se le cambia el número a la carta favorita del usuario, a través de las cartas disponibles que tiene el jugador.

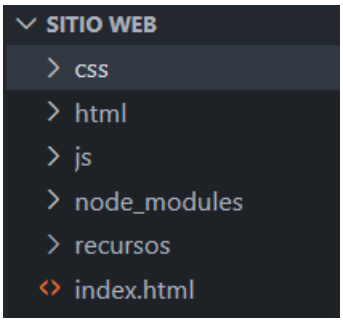
```
}
else if (request.url == "/cambiar_carta_favorita") {
    console.log("se pidio cambiar el id de una tarjeta")
    let body = "";
    request.on("data", chunk => {
        body += chunk;
    });

    request.on("end", () => {
        const data = JSON.parse(body);
        const { nueva_card_favorita, cuenta } = data;

        console.log("Cuenta:", cuenta);
        console.log("ID de la NUEVA tarjeta favorita:", nueva_card_favorita);

        conexion_bd.query(
            "UPDATE cuentas SET carta_favorita = ? WHERE id = ?",
            [nueva_card_favorita, cuenta],
            (err, resultado) => {
                if (err) {
```

El caso delete solo tiene un enlace que es la eliminación de una carta de tu lista de cartas.

```
case "DELETE":
    if (request.url === "/borrar_carta_de_la_cuenta") {
        let body = "";
        request.on("data", chunk => {
            body += chunk;
        });
        console.log("si se ejecuto el delete")
        request.on("end", () => {
            const data = JSON.parse(body);
            const { idTarjeta, cuenta } = data;

            console.log("Cuenta:", cuenta);
            console.log("ID de tarjeta a eliminar:", idTarjeta);

            conexion_bd.query("SELECT cartas_cuenta FROM cuentas WHERE id = ?", [cuenta], (err, resu
                if (err) {
                    console.error("No esta la cuenta: ", err);

                    response.writeHead(500, { "Content-Type": "application/json" });
                    response.end(JSON.stringify({ mensaje: "Error de base de datos" }));
                } else if (resultado.length === 0) {
                    response.writeHead(404, { "Content-Type": "application/json" });
                    response.end(JSON.stringify({ mensaje: "No encontramos la cuenta" }));
                }
```

La explicación que no existe otra acción es que realmente la mayoría la veía relacionada con put, incluyendo el restar dinero, porque más que eliminar, solo quitas un porcentaje de las monedas.

Organización de archivos del sitio

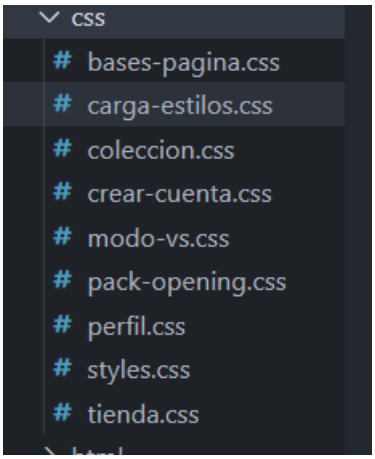


El sitio web solamente tiene en index fuera de cualquier carpeta, a continuación veamos que tiene cada carpeta

CSS: css tiene cada hoja de estilos de la web. Como información importante:

- bases-página se encuentra conectada a todas las páginas y tiene los elementos más comunes, claves y repetidos de la web.
- Carga-estilos solo son los estilos para el elemento de carga que existe y se puede reutilizar en la web.

- Crear-cuenta.css realmente es usado para crear-cuenta e iniciar-sesion
- Pack-opening.css se usa para los 4 enlaces de abridores de sobres
- Styles es solo para index



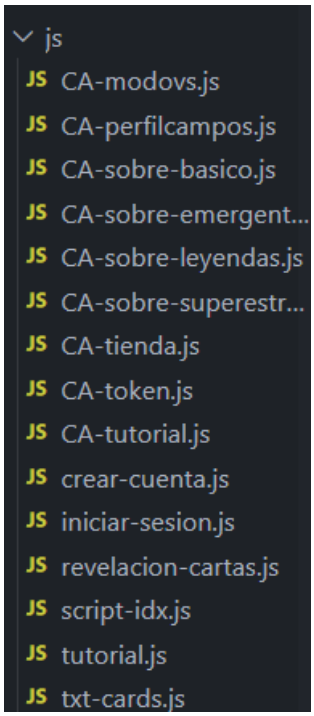
JS:

Posiblemente si se quiere expandir y mejorar el sitio, lo primero que se debe cambiar. Los archivos se encuentran con lógica, pero relativamente mal estructurados.

- CA significa conectar-api, por lo tanto si un archivo comienza con estas iniciales, su principal característica es referente a la conexión entre el sitio y la api
- Script-idx es solo para el icono del sonido en el video inicial de la pagina
- Txt-cards es en dado caso que una tarjeta tenga un nombre muy largo que pueda llegar a no entrar en la tarjeta de forma normal
- Crear cuenta e iniciar sesión por obviedad también se conecta al api, a pesar de no tener estas iniciales
- CA token es para la conexión en colección, es el archivo con más peticiones diferentes
- CA sobre leyenda sin embargo es el que más peticiones tiene, aunque todas las aperturas de sobre son muchas
- Tutorial.js tiene el sistema de música
- El sistema de peticiones para abrir un sobre es el siguiente

1. Pide todas las tarjetas de x rango

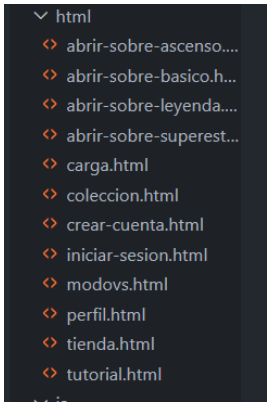
2. Dentro de la primera petición hace un ciclo for
3. En el ciclo for cada vez que se realiza hace un rango entre el número más bajo de los id del rango de turno y el más alto
4. Cada repetición hace una petición de la carta individual con un número aleatorio entre el rango obtenido en el punto número 3
5. Rellena con la información de esa petición
6. Y así funciona para cada repetición, el resto son porcentajes básicos



HTML

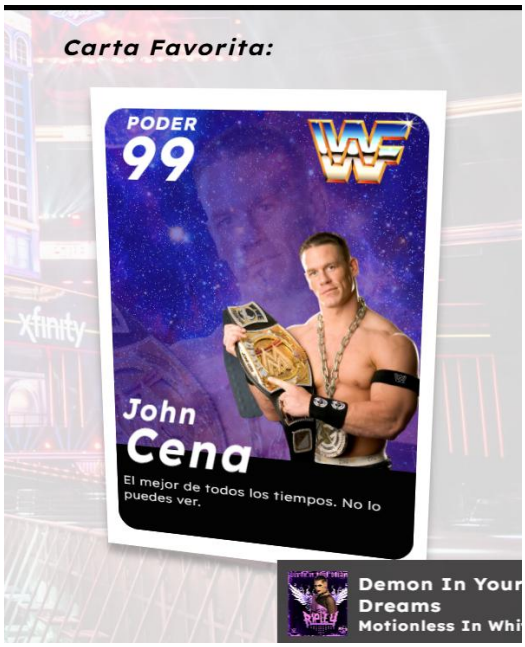
Aquí vienen el resto de los html

- Cada sobre tiene diferente html
- Carga.html es solo una pantalla de prueba con la animación de carga, no es relevante



Librerías usadas:

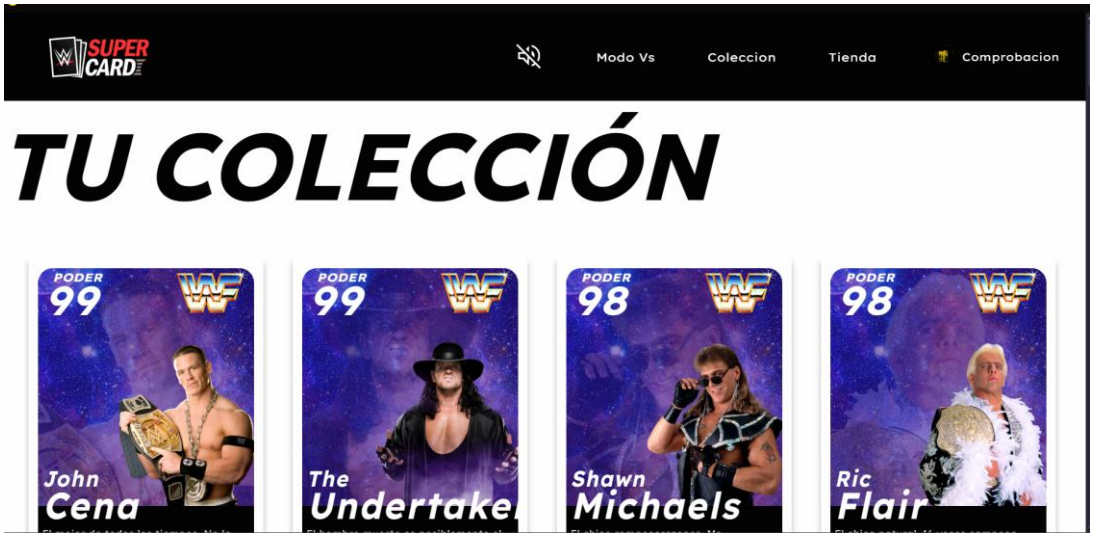
Vanilla Tilt: La animación en 3D de las cartas en sección de perfil es una librería para el uso del mouse.



AnimatiCSS:

Se uso para las animaciones de las cartas en colección. Esto realmente es puro código css y no se tuvo que descargar nada, pero es necesario dar los créditos correspondientes

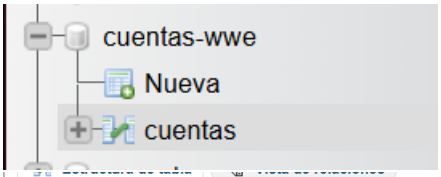
https://xsgames.co/animatiss/



Pasos para descargar el proyecto:

1. Tener dos VS, uno abriendo solo sitio web y el otro abriendo API

2. Crear una base de datos que se llame “cuentas-wwe” y allí se pondrá el sql de la única tabla que tiene esta base de datos. Debería tener estos campos



#	Nombre	Tipo	Cotejamiento	Atributos	Nulo	Predeterminado	Comentarios	Extra	Acción
1	**ID**	int			No	*Ninguna*		AUTO_INCREMENT	Cambiar Eliminar Más
2	**foto_perfil**	longtext	utf8mb3_spanish_ci		Sí	*NULL*			Cambiar Eliminar Más
3	**usuario**	varchar(20)	utf8mb3_spanish_ci		No	*Ninguna*			Cambiar Eliminar Más
4	**contrasena**	varchar(50)	utf8mb3_spanish_ci		No	*Ninguna*			Cambiar Eliminar Más
5	**monedas**	int			No	50			Cambiar Eliminar Más
6	**cartas_cuenta**	json			No	*Ninguna*			Cambiar Eliminar Más
7	**carta_favorita**	int			No	1			Cambiar Eliminar Más
8	**puntaje_maximo**	int			No	*Ninguna*			Cambiar Eliminar Más
9	**realizando_compra**	tinyint(1)			No	0			Cambiar Eliminar Más

Seleccionar todo *Para los elementos que están marcados:* Examinar Cambiar Eliminar Primaria Único Índice E
Texto completo

3. En el archivo api.js que será donde se hará el node, por la línea 49 donde se crea la constante para la conexión, tengo la contraseña que puse en mi MySQL 9.0, teóricamente con dejar la contraseña en blanco si su computador no tiene basta, pero lo señalo porque es algo exclusivo de mí máquina.

```
47  const basedatos = require("mysql2")
48  //linea del codigo para que funcione mi base de datos
49  const conexion_bd = basedatos.createConnection({
50      host: "localhost",
51      user: "root",
52      password: "patata12",
53      database: "cuentas-wwe"
54  })
```

4. Lo hice con Live Server, en el hipotético caso que no se vea ciertas cosas es por eso. (Si el video de al inicio no sale es común, es porque como todavía no ha hecho una interacción el usuario, el navegador no da autoplay)

Recomendaciones de uso para no perderse:

Crear una cuenta

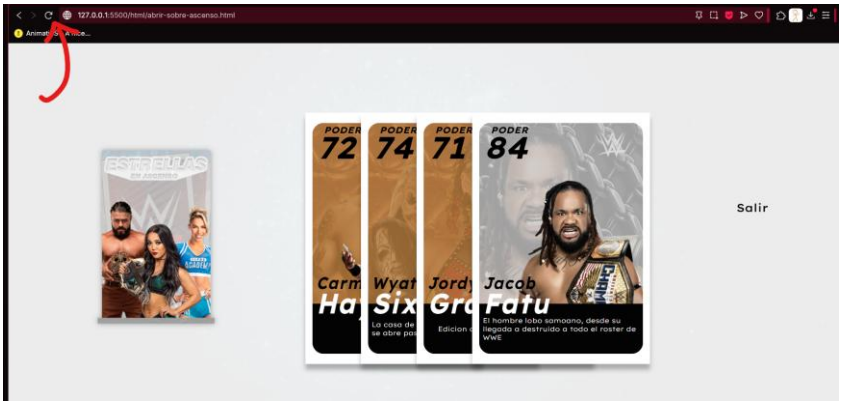
Leer lo que dice John Cena en el tutorial para información rápida sobre lo disponible (si alguna vez quiere volver se encuentra el link en su perfil)

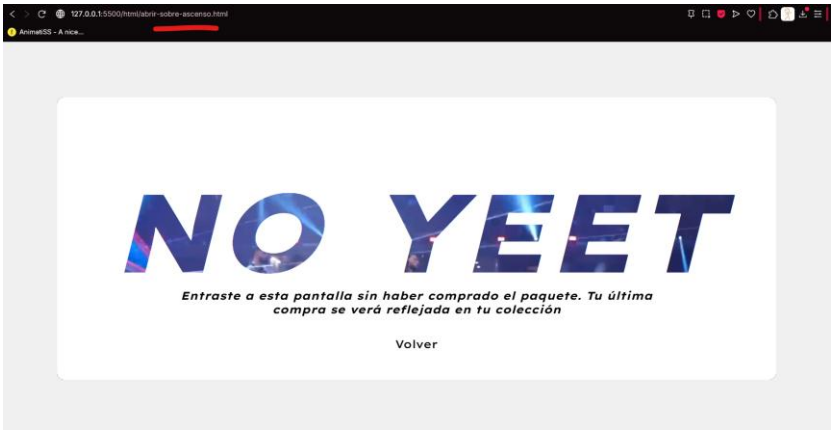
Compre algún sobre, si se queda sin dinero vaya a colección y venda sus cartas para obtener más dinero. El flujo de dinero está pensado para que por sí solo nunca te quedes sin dinero.

También existe el modo VS que es un excelente método para farmear dinero.

Notas extras:

- Ver perfil si quiere ver su carta favorita.
- Si borra su carta favorita en colección se le asigna automáticamente como nueva carta favorita la más poderosa en su baraja
- El sistema de comprar sobres te prohíbe reiniciar la página o entrar por medio de escribir directamente el link para evitar que obtenga cartas de forma injusta. (dicho sistema es el único que utiliza el token, sería practico que los demás también fuesen así).





Gracias por leer***. En el muy hipotético caso que no funcione el método post para crear una cuenta (cosa que no debería de ser así, a mí me funciona). Estos son datos genéricos recomendados para insertar directamente en phpMyAdmin.***

ID	int			35
foto_perfil	longtext		☐	 TMAAAgAElEQVR4nOzdeYxm6XXf93OXd6+l956VM5yhSIoUqYUWo42KEyiJJJPsPJZ JjJwEcJAisAI5sIHECBEkgKXA2WED+SuAAhmWJiJxYC01Z4iYxFMkRl+Eiipx97Z 7p6bX2qne7712Cc57nufe+1T0bu2d6OPX9CK3qruVd7tusrt+c85wjAAAAAAAAAA AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
usuario	varchar(20)			SuNombre
contrasena	varchar(50)			SuContrasena
monedas	int			50
cartas_cuenta	json			1
carta_favorita	int			1
puntaje_maximo	int			0
realizando_compra	tinyint(1)			0

https://www.base64-image.de

El anterior enlace es una página para que pueda obtener su propio url de base64 de una imagen.

Nota: También se puede verificar esto, que es el archivo config.inc.php

Se encuentra en la carpeta de phpMyAdmin

po > OS (C:) > xampp > phpMyAdmin >

```
$i++;

/* Authentication type and info */
$cfg['Servers'][$i]['auth_type'] = 'config';
$cfg['Servers'][$i]['user'] = 'root';
$cfg['Servers'][$i]['password'] = 'patata12';
$cfg['Servers'][$i]['extension'] = 'mysqli';
$cfg['Servers'][$i]['AllowNoPassword'] = true;
$cfg['Lang'] = '';

/* Bind to the localhost ipv4 address and tcp */
```

Puede ser eso también. De cualquier forma, me encuentro disponible si llegase a no funcionar. En el muy hipotético caso.

Con esto ya mencionado, ahora si es todo. Gracias por leer y espero disfrute el proyecto tanto como yo disfruté haciéndolo.