UNIVERSIDAD TECNOLÓGICA METROPOLITANA

14/06/2024

PRACTICA 2

Consumo de APIS

Actividad 2

Aplicaciones WEB

Matinez Ruth

Basto Tumux Miguel

3A DSM

Introducción

El uso de APIs (Interfaces de Programación de Aplicaciones) se ha vuelto esencial en el mundo digital. Estas interfaces actúan como conectores que permiten una comunicación fluida y segura entre diversas plataformas y aplicaciones. Gracias a las APIs, se ha revolucionado la forma en que desarrolladores y empresas crean, implementan y optimizan sus productos y servicios.

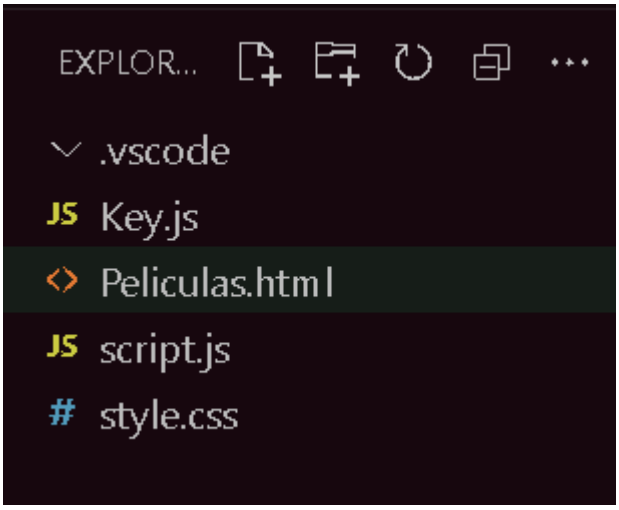
Las APIs son cruciales en el desarrollo de aplicaciones modernas, ya que brindan a los desarrolladores acceso a funcionalidades y datos de sistemas externos de manera estructurada y controlada. Esto no solo agiliza el desarrollo al aprovechar el trabajo existente, sino que también fomenta la reutilización de código y la modularidad en el diseño de software. Las APIs están presentes en prácticamente todos los ámbitos tecnológicos, desde redes sociales hasta sistemas de pago, desde el Internet de las Cosas hasta la inteligencia artificial.

La adopción generalizada de las APIs ha impulsado un ecosistema de innovación dinámico y colaborativo. Empresas de todos los tamaños pueden ahora incorporar fácilmente funciones avanzadas sin necesidad de desarrollarlas internamente, lo que reduce costos y plazos de desarrollo. Además, las APIs abren nuevas oportunidades de negocio al permitir la creación de productos y servicios que combinan diversas tecnologías de manera sinérgica.

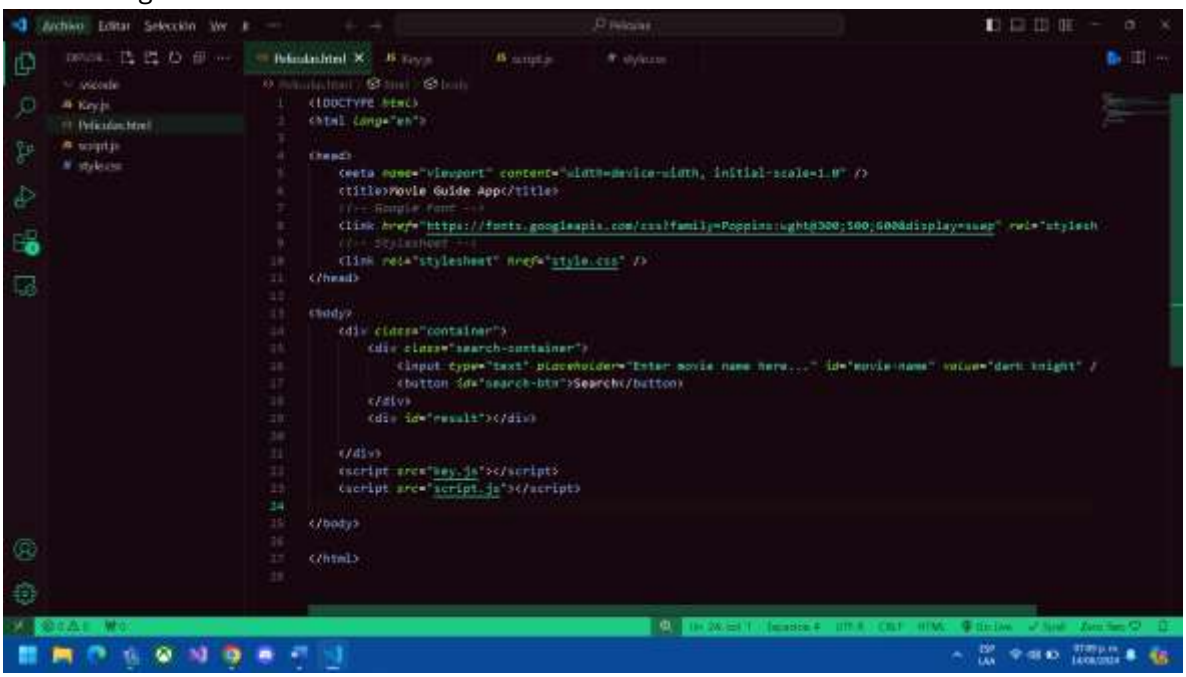
Sin embargo, el uso de APIs presenta desafíos. La seguridad y la gestión de versiones son preocupaciones importantes, ya que las APIs pueden ser puntos vulnerables para ciberataques si no se manejan adecuadamente. Además, la dependencia de APIs externas puede generar riesgos operativos si no se supervisan y gestionan de forma eficaz.

INSTRUCCIONES.

Crear estructura: En Visual Studio Code (VSCode), crea una carpeta llamada "películas" y dentro de ella, cuatro archivos: uno HTML, dos JavaScript y uno CSS.



Diseño HTML básico: En el archivo `películas.html`, agrega un botón, un campo de entrada para búsqueda y un contenedor (`div`) con el ID "result" donde se mostrará el contenido generado dinámicamente.



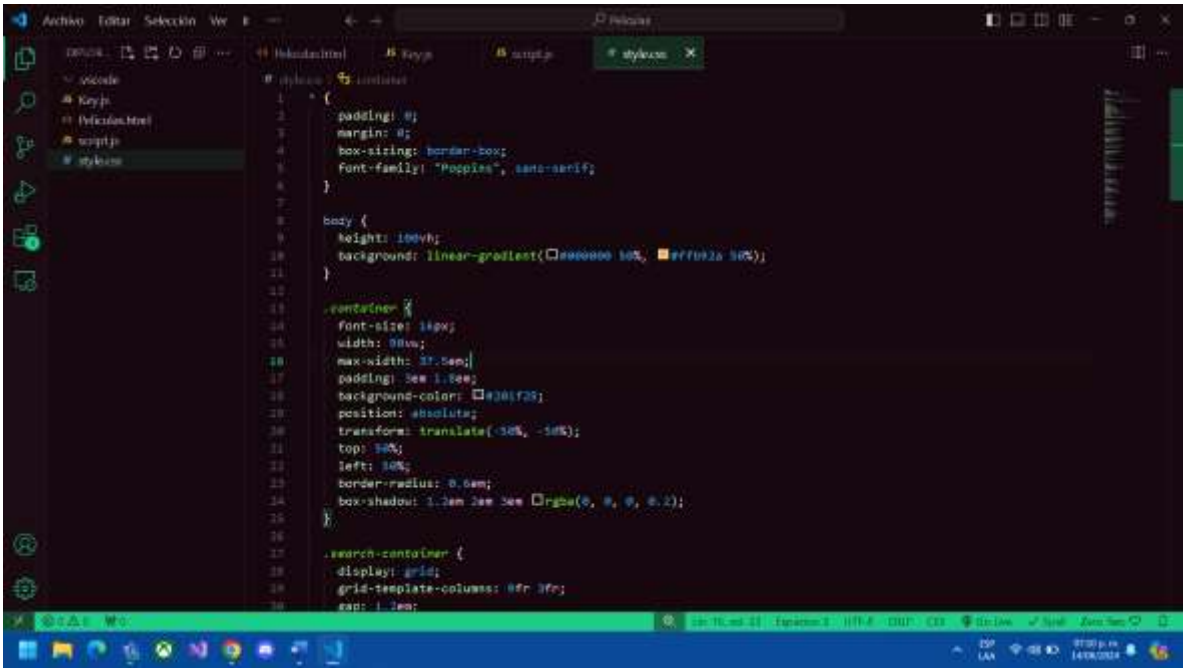
Estilos CSS: En el archivo CSS, añade el código proporcionado para dar estilo visual a la aplicación.

```
* {
    padding: 0;
    margin: 0;
```

```
  box-sizing: border-box;
  font-family: "Poppins", sans-serif;
}

body {
  height: 100vh;
  background: linear-gradient(#000000 50%, #ffb92a 50%);
}

.container {
  font-size: 16px;
  width: 90vw;
  max-width: 37.5em;
  padding: 3em 1.8em;
  background-color: #201f28;
  position: absolute;
  transform: translate(-50%, -50%);
  top: 50%;
  left: 50%;
  border-radius: 0.6em;
  box-shadow: 1.2em 2em 3em rgba(0, 0, 0, 0.2);
}

.search-container {
  display: grid;
  grid-template-columns: 9fr 3fr;
  gap: 1.2em;
}

.search-container input,
.search-container button {
  font-size: 0.9em;
  outline: none;
  border-radius: 0.3em;
}

.search-container input {
  background-color: transparent;
  border: 1px solid #a0a0a0;
  padding: 0.7em;
  color: #ffffff;
}

.search-container input:focus {
  border-color: #ffffff;
}

.search-container button {
  background-color: #ffb92a;
  border: none;
```

```
  cursor: pointer;
}

#result {
  color: #ffffff;
}

.info {
  position: relative;
  display: grid;
  grid-template-columns: 4fr 8fr;
  align-items: center;
  margin-top: 1.2em;
}

.poster {
  width: 100%;
}

h2 {
  text-align: center;
  font-size: 1.5em;
  font-weight: 600;
  letter-spacing: 0.06em;
}

.rating {
  display: flex;
  align-items: center;
  justify-content: center;
  gap: 0.6em;
  margin: 0.6em 0 0.9em 0;
}

.rating img {
  width: 1.2em;
}

.rating h4 {
  display: inline-block;
  font-size: 1.1em;
  font-weight: 500;
}

.details {
  display: flex;
  font-size: 0.95em;
  gap: 1em;
  justify-content: center;
  color: #ada0a0;
```

```
    margin: 0.6em 0;
    font-weight: 300;
  }

  .genre {
    display: flex}
```

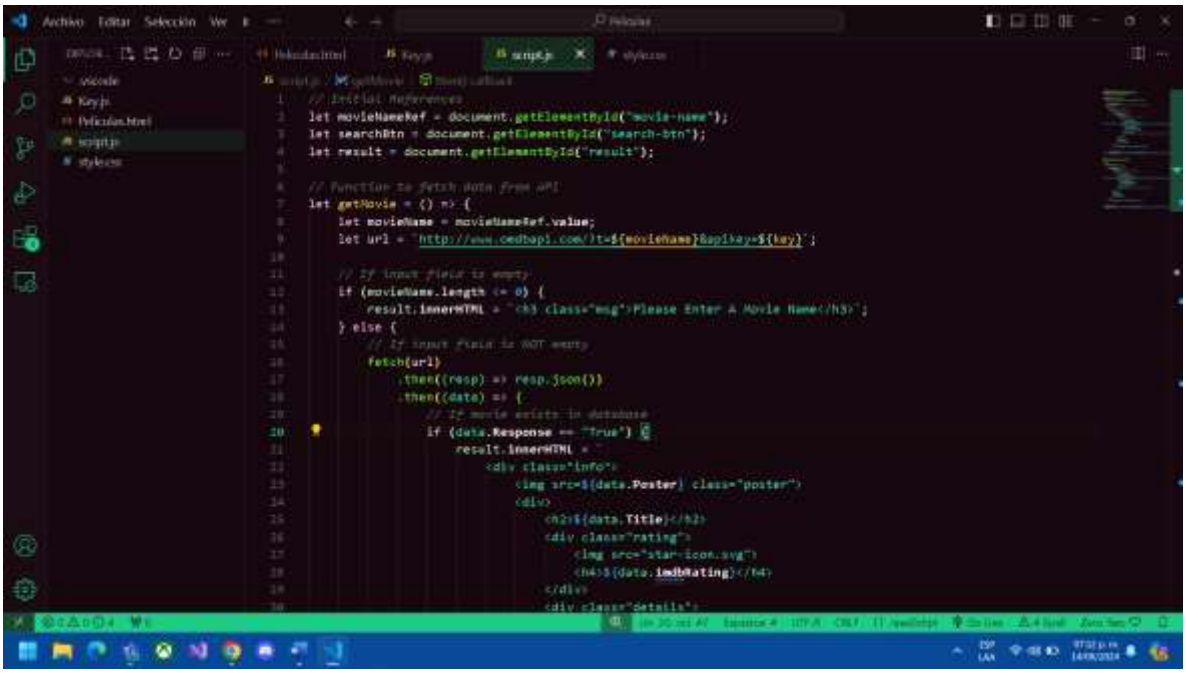


Lógica JavaScript: En el archivo script.js, agrega el código JavaScript para manejar eventos, consumir la API y mostrar la información de las películas dinámicamente.

```
// Initial References
let movieNameRef = document.getElementById("movie-name");
let searchBtn = document.getElementById("search-btn");
let result = document.getElementById("result");

// Function to fetch data from API
let getMovie = () => {
    let movieName = movieNameRef.value;
    let url = `http://www.omdbapi.com/?t=${movieName}&apikey=${key}`;

    // If input field is empty
    if (movieName.length <= 0) {
        result.innerHTML = `<h3 class="msg">Please Enter A Movie Name</h3>`;
    } else {
        // If input field is NOT empty
        fetch(url)
            .then((resp) => resp.json())
            .then((data) => {
                // If movie exists in database
                if (data.Response == "True") {
                    result.innerHTML = `
                        <div class="info">
                            <img src=${data.Poster} class="poster">
                            <div>
                                <h2>${data.Title}</h2>
                                <div class="rating">
                                    <img src="star-icon.svg">
                                    <h4>${data.imdbRating}</h4>
                                </div>
                                <div class="details">
                                    <span>${data.Rated}</span>
                                    <span>${data.Year}</span>
                                    <span>${data.Runtime}</span>
                                </div>
                                <div class="genre">
                                    ${data.Genre.split(",").map(genre =>
`<div>${genre.trim()}</div>`).join("")}
                                </div>
                            </div>
                        </div>
                        <h3>Plot:</h3>
                        <p>${data.Plot}</p>
                        <h3>Cast:</h3>
                        <p>${data.Actors}</p>
                    `;
                } else {
                    // If movie does NOT exist in database
```

```
                    result.innerHTML = `<h3 class='msg'>${data.Error}</h3>`;
                }
            })
            // If error occurs
            .catch(() => {
                result.innerHTML = `<h3 class="msg">Error Occured</h3>`;
            });
    }
};

searchBtn.addEventListener("click", getMovie);
window.addEventListener("load", getMovie);

fetch(url)
    .then((resp) => resp.json())
    .then((data) => {
        // If movie exists in database
        if (data.Response == "True") {
            result.innerHTML = `
                <div class="info">
                    <img src=${data.Poster} class="poster">
                    <div>
                        <h2>${data.Title}</h2>
                        <div class="rating">
                            <img src="star-icon.svg">
                            <h4>${data.imdbRating}</h4>
                        </div>
                        <div class="details">
                            <span>${data.Rated}</span>
                            <span>${data.Year}</span>
                            <span>${data.Runtime}</span>
                        </div>
                        <div class="genre">
                            <div>${data.Genre.split(",").join("</div><div>")
}</div>
                        </div>
                    </div>
                </div>
                <h3>Plot:</h3>
                <p>${data.Plot}</p>
                <h3>Cast:</h3>
                <p>${data.Actors}</p>
            `;
        } else {
            // If movie does NOT exist in database
            result.innerHTML = `<h3 class='msg'>${data.Error}</h3>`;
        }
    })
    // If error occurs
    .catch(() => {
```

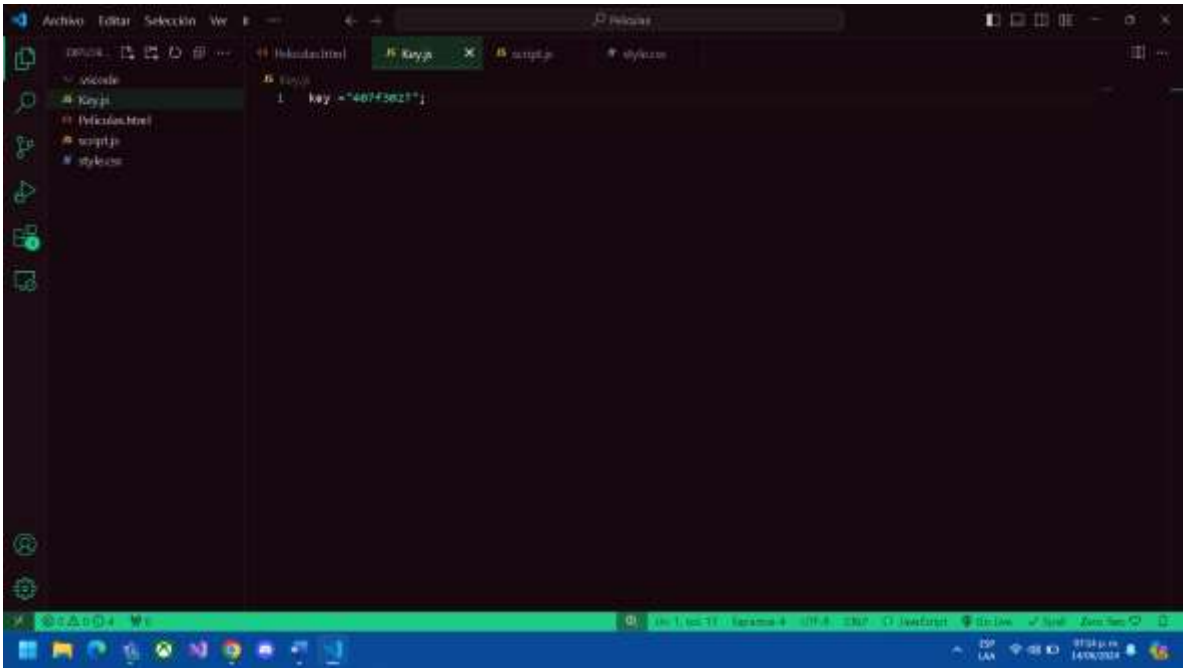
```
        result.innerHTML = `<h3 class="msg">Error Occured</h3>`;
    });
```



```
fetch(url)
    .then((resp) => resp.json())
    .then((data) => {
        // If movie exists in database
        if (data.Response == "True") {
            result.innerHTML = `
                <div class="info">
                    <img src=${data.Poster} class="poster">
                    <div>
                        <h2>${data.Title}</h2>
                        <div class="rating">
                            <img src="star-icon.svg">
                            <h4>${data.imdbRating}</h4>
                        </div>
                        <div class="details">
                            <span>${data.Rated}</span>
                            <span>${data.Year}</span>
                            <span>${data.Runtime}</span>
                        </div>
                        <div class="genre">
                            <div>${data.Genre.split(",").join("</div><div>")
}</div>
                        </div>
                    </div>
                </div>
                <h3>Plot:</h3>
                <p>${data.Plot}</p>
```

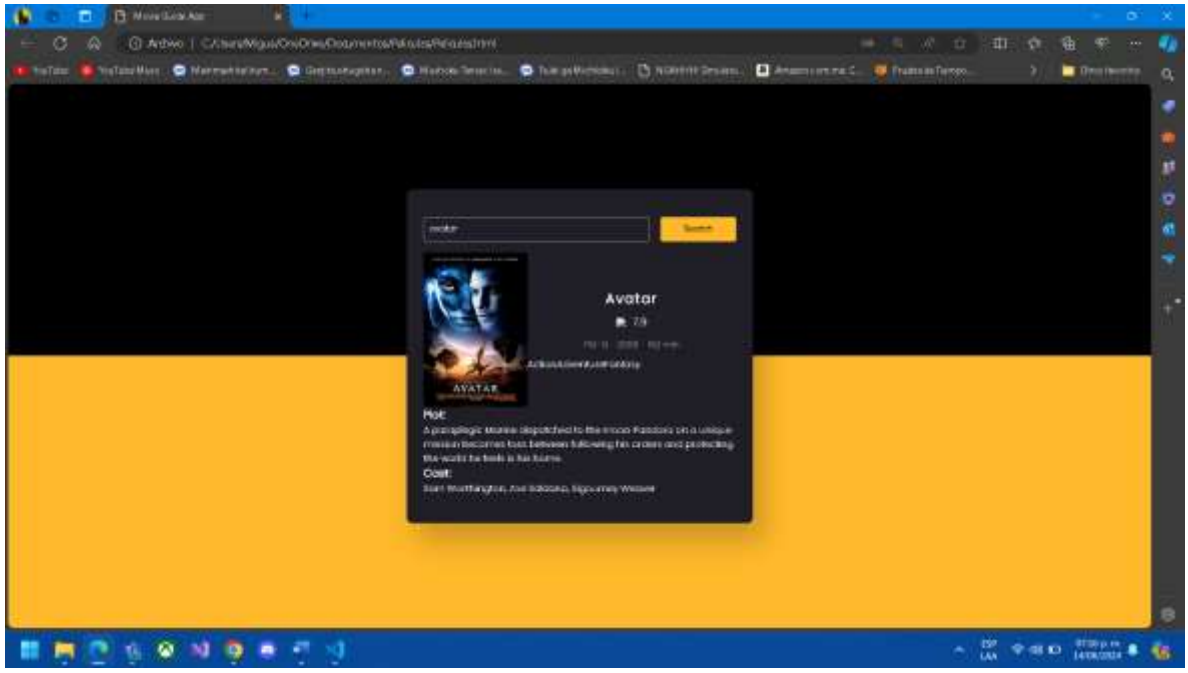
```
                <h3>Cast:</h3>
                <p>${data.Actors}</p>
            `;
        } else {
            // If movie does NOT exist in database
            result.innerHTML = `<h3 class='msg'>${data.Error}</h3>`;
        }
    })
    // If error occurs
    .catch(() => {
        result.innerHTML = `<h3 class="msg">Error Occured</h3>`;
    });
```

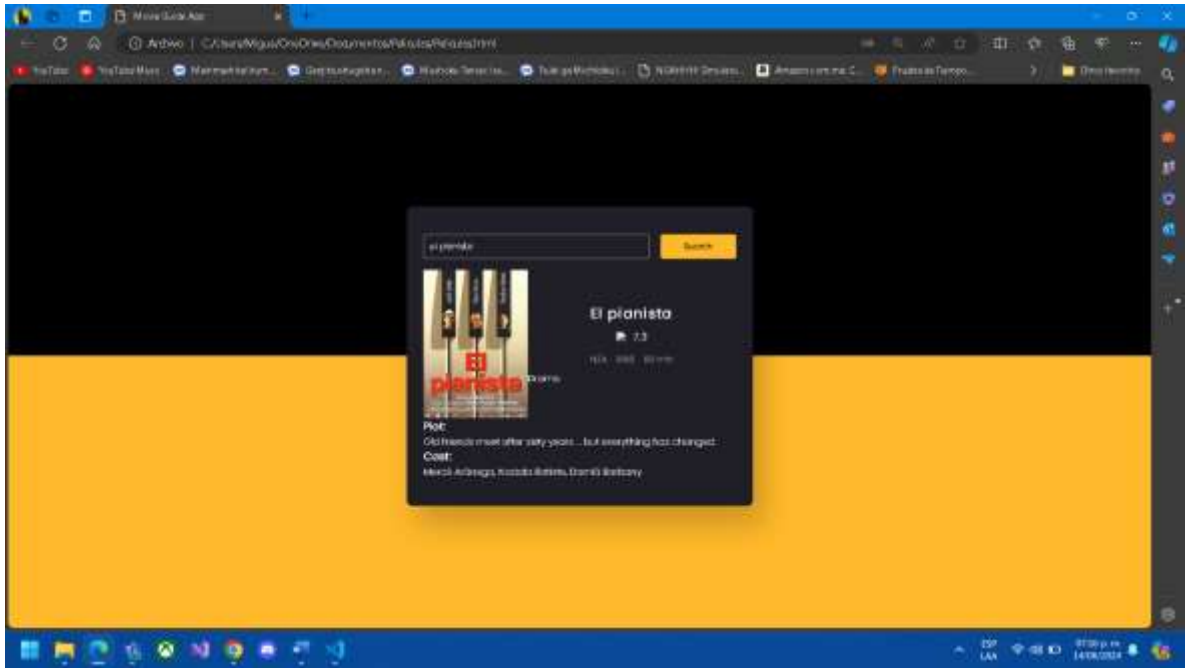
Clave API: En el archivo `key.js`, incluye la clave API necesaria para acceder a la base de datos de películas (OMDb API)

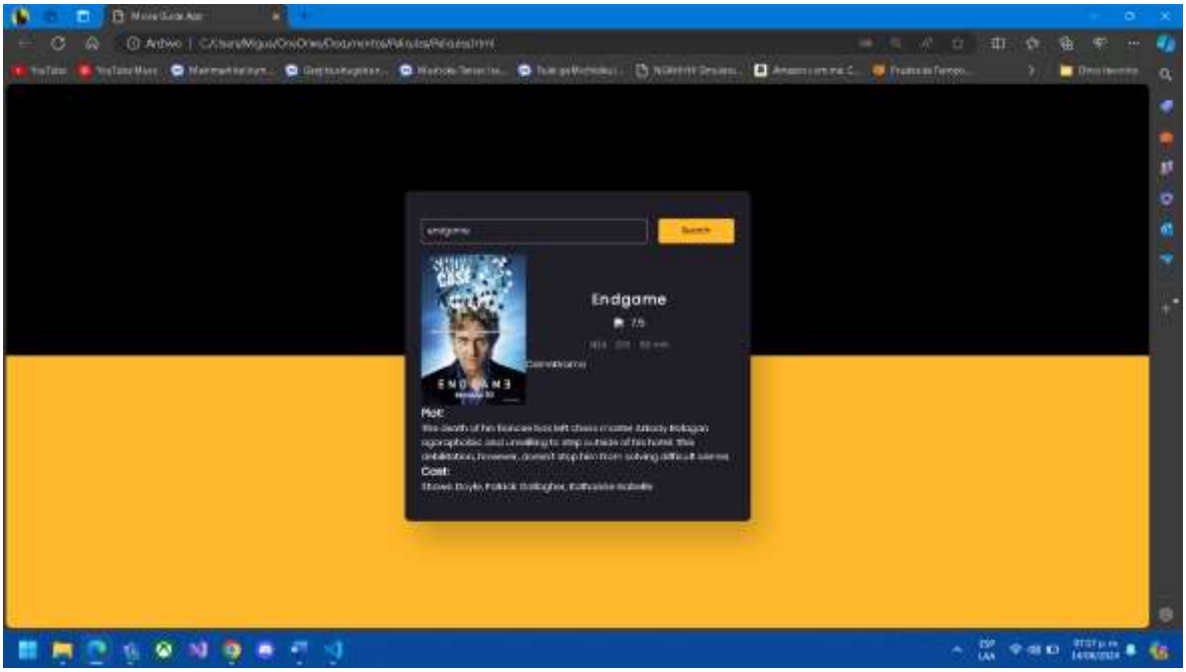


Este último punto es muy importante, por lo general, la mayoría de apis están bloqueadas para que puedan controlar el manejo del mismo, sin esta key, no podremos usar el script y mucho menos usar la api

Probar: Ejecuta el proyecto y verifica que funciona correctamente, mostrando información sobre películas al realizar búsquedas.






endgame
Search
Endgame
7.5
Plot:
Cast:

Conclusión

Las APIs son herramientas esenciales en el desarrollo de software moderno, ya que permiten la comunicación e integración entre diferentes sistemas y aplicaciones. Facilitan el acceso a funcionalidades y datos externos, agilizando el desarrollo y promoviendo la reutilización de código.

La adopción generalizada de APIs ha fomentado la innovación y colaboración, permitiendo a empresas de todos los tamaños integrar funciones avanzadas sin necesidad de desarrollarlas desde cero. Sin embargo, es crucial abordar los desafíos de seguridad y gestión de versiones para aprovechar al máximo los beneficios de las APIs y mitigar sus riesgos potenciales.