Universidad Tecnológica Metropolitana		
Código	**INSTRUMENTO DE EVALUACIÓN**	Revisión:
F-SGC-033		**00**

DATOS GENERALES DEL INSTRUMENTO.

División: **Tecnologías de la Información**

FDC*/Carrera: **IDGS**

Asignatura: **Desarrollo de Aplicaciones Web**

Cuat.-Gpo(s): **3A, 3B, 3C, 3D, 3E, 3F** Fecha de aplicación: **Junio 2024**

Unidad(es) de aprendizaje y/o tema(s) a evaluar.

Unidad 3. Backend y consumo de API.

Especificar con una “X” el tipo de instrumento de evaluación a utilizar (señalar sólo uno).

Tec. evaluación para el SABER				Tec. evaluación para el SABER HACER + SER			
	Prueba oral (entrevista)		Otro (Especificar):		Proyectos		Otro (Especificar):
	Prueba escrita			X	Prácticas, ejercicios, demostraciones		
	Trabajo investigación				Rúbrica		
	Ensayo, informe				Lista de cotejo		
					Guía de observación		

Profesor(es) de la asignatura: Ing. Diego May, ISC. Ruth Betsaida Martínez Domínguez

Nombre del alumno: Calificación (puntaje): 2.5 %

PRÁCTICA . Elaboración de ejemplo con consumo de api.

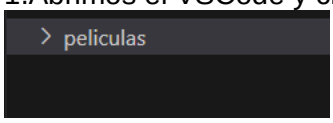
Tema. Consumo de api – Backend

Duración estimada: 00:60:00 minutos.

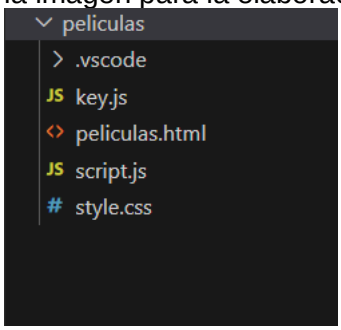
Objetivo: El alumno elaborará un ejemplo funcional de aplicación web con el consumo de una api existente de películas.

INSTRUCCIONES.

1.Abrimos el VSCode y creamos una carpeta llamada películas.



2. Dentro de la carpeta películas creamos 4 archivos un html, 2 js, y un css como se puede apreciar en la imagen para la elaboración de nuestro proyecto.



3.Dentro de nuestro archivo películas.html agregamos el siguiente código.

*FDC: familia de carreras

peliculas > peliculas.html > html > body > div.container > div.search-container > input#movie-name

```html
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <title>Movie Guide App</title>

    <!-- Google Font -->
    <link
      href="https://fonts.googleapis.com/css2?family=Poppins:wght@300;500;600&display=swap"
      rel="stylesheet"
    />
    <!-- Stylesheet -->
    <link rel="stylesheet" href="style.css" />
  </head>
  <body>
    <div class="container">
      <div class="search-container">
        <input type="text" placeholder="Enter movie name heree..." id="movie-name" value="dark knight" />
        <button id="search-btn">Search</button>
      </div>
      <div id="result"></div>
    </div>
    <script src="key.js"></script>
    <script src="script.js"></script>
  </body>
</html>
```

En este ejemplo únicamente necesitamos un botón, un input para buscador y un div con id=”result” que es donde se generara de manera dinámica nuestro front.

4. En nuestro archivo css agregamos el siguiente código para la parte del diseño visual de nuestro ejemplo.

peliculas > # style.css > *

```css
* {
  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;
}
```

```css
.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: #a0a0a0;
  margin: 0.6em 0;
  font-weight: 300;
}
.genre {
  display: flex;
  justify-content: space-around;
}
.genre div {
  border: 1px solid #a0a0a0;
  font-size: 0.75em;
  padding: 0.4em 1.6em;
  border-radius: 0.4em;
  font-weight: 300;
}
h3 {
  font-weight: 500;
  margin-top: 1.2em;
}
p {
  font-size: 0.9em;
  font-weight: 300;
  line-height: 1.8em;
  text-align: justify;
  color: #a0a0a0;
}
.msg {
  text-align: center;
}
@media screen and (max-width: 600px) {
  .container {
    font-size: 14px;
  }
```

```
    .info {
      grid-template-columns: 1fr;
    }
    .poster {
      margin: auto;
      width: auto;
      max-height: 10.8em;
    }
}
```

Con este código ya tendríamos la parte de diseño css de nuestra aplicación web.

5. Posteriormente en nuestro archivo script.js agregamos el siguiente código.

peliculas > JS script.js > getMovie > then() callback

```
//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>`;
  }
  //If input field is NOT empty
  else {
    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>

        `;
        }
        //If movie does NOT exists in database
        else {
          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);
```

Con este código generaremos los eventos javascript necesarios para la creación dinámica de nuestro front y para el consumo del api que necesitamos para obtener la información a mostrar en nuestra página.

El código que realiza la petición por medio del fetch es el siguiente.

```
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>

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

En e cual se realiza la petición de la información por medio de fetch a la url del api a consumir, y la respuesta del api se parsea en un json para un uso mas fácil y sencillo y poder ir armando nuestro código html de manera dinámica con la información devuelta de la consulta del api.

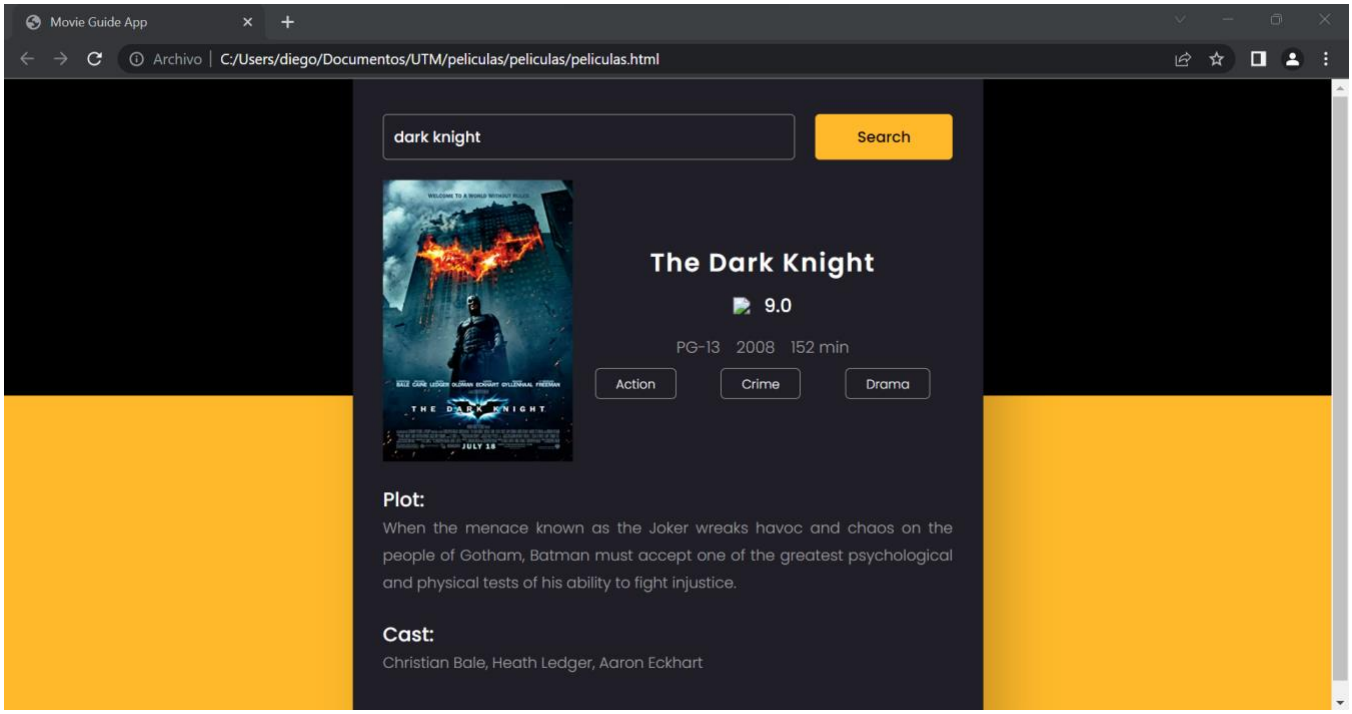
6. Posteriormente en nuestro archivo key.js agregamos la siguiente línea de código.

peliculas > JS key.js

```
1   key = "407f3027";
```

Este dato es necesario ya que la api que estamos consumiendo se encuentra protegida y dicha key es el acceso al api para poder consumir, de no agregar esta key, el api devolverá un error de que no se tienen permiso para consumo.

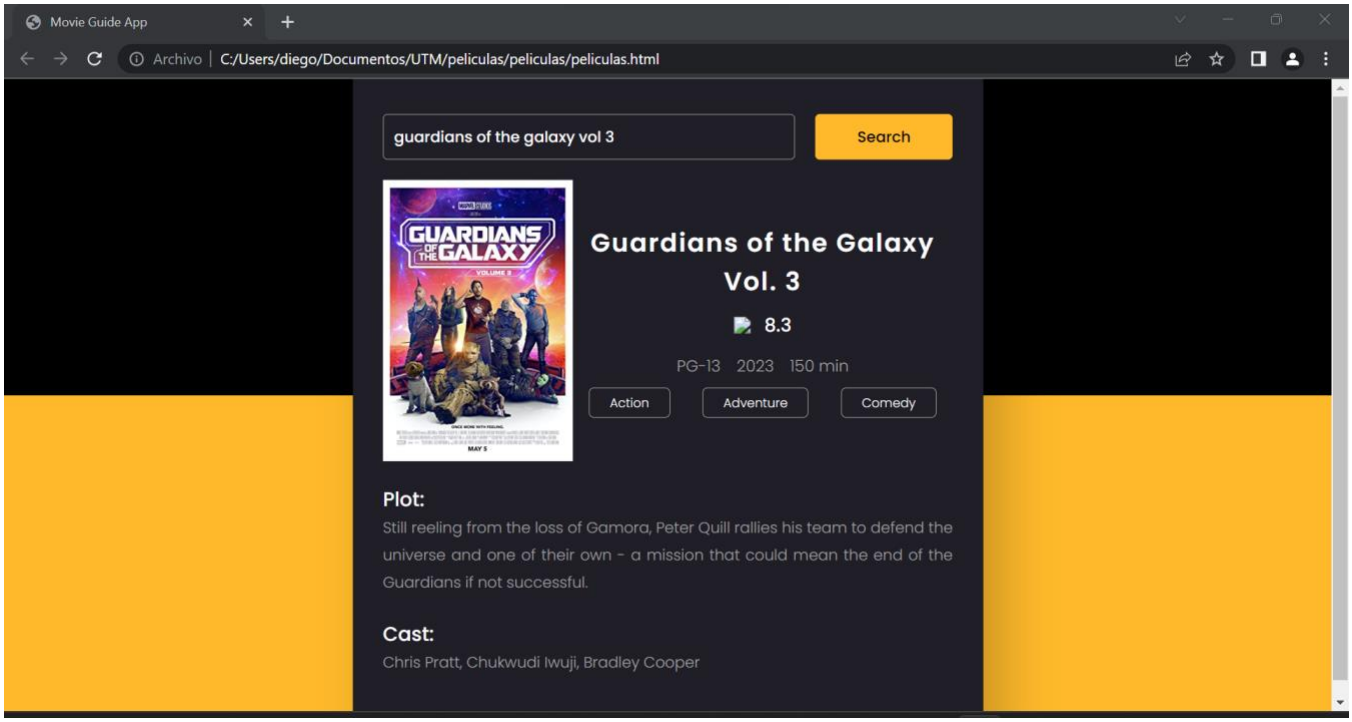
Con esto ya tendríamos el proyecto listo para ejecutar, únicamente probamos y nos debe devolver algo como lo siguiente:



En este ejemplo nos inicializa el proyecto con la película de dark knight ya que es el valor por defecto que tenemos indicado en el value del input de búsqueda.

```
18        <input type="text" placeholder="Enter movie name heree..." id="movie-name" value="dark knight" />
```

Ya teniendo el ejemplo en ejecución podemos probar con otros nombre de películas y nos ira trayendo la información de acuerdo a lo que el api tenga almacenado en la BD.



FORMATO: Proyecto en repositorio Git.

FECHA DE ENTREGA: Subir un el proyecto al repositorio Git y compartir la liga del repositorio.

VALIDACION DE LA ACADEMIA*

Nombre de los integrantes de la academia	Firma
ISC. Ruth Betsaida Martínez Domínguez Ing. Diego May	

* Este apartado solo se llenará para la entrega de este instrumento a la División correspondiente.