



**universidad
de león**



Memoria del proyecto para SISTEMAS DE INFORMACION DE GESTION Y BUSINESS INTELLIGENCE

YouPlay: Aplicación de un sistema de recomendación



Autor: Víctor de la Fuente Martínez

ÍNDICE

INTRODUCCIÓN AL PROYECTO	3
OBJETIVO Y DESCRIPCIÓN DEL PROBLEMA	4
HERRAMIENTAS UTILIZADAS	5
DESCRIPCIÓN DE LA APLICACIÓN	8
Descripción general	8
Frontend	9
Base de datos	20
EXPLICACIÓN DEL ALGORITMO	23
ANÁLISIS DE RESULTADOS	25
DAFO ANÁLISIS CRÍTICO DE LA APLICACIÓN	36
PROBLEMAS EN EL DESARROLLO	39
LÍNEAS DE FUTURO	40
LECCIONES APRENDIDAS	41
BIBLIOGRAFÍA	42

Introducción al proyecto

Hace unos años, cuando pensábamos o queríamos recomendaciones acerca de algún tema en concreto, como pueden ser música, cine, libros, comida... En definitiva, algo de lo que quisiésemos ganar conocimiento para que se ajustara a lo que buscamos, pensábamos en amigos, familiares o gente famosa que supiera del tema. A partir de esta revolución tecnológica, esta misma tecnología nos ha provisto de todo tipo de materiales para el desempeño de páginas web y aplicaciones que en función de un algoritmo muy sofisticado y por medio de filtros son capaces de recomendarnos lo que sea en función del tipo de tema que trate la página. Algún ejemplo sería la aplicación Spotify, que en función de nuestra cuenta y los gustos de música que hayamos escuchado o introducido previamente, los recoge y nos empieza a recomendar música la cual piensa que nos puede llegar a gustar, acertando en muchas de las ocasiones.

El tema del que trata este proyecto es de ser capaz de recomendar los mejores videojuegos a la gente que lo desee y, a mayores, ver cuáles son los que más gustan y los que menos para que la gente se haga una idea de si a la gente le gusta un videojuego o no de manera previa a comprarlo o jugarlo.

Esta idea surgió de una incipiente demanda tanto en el mundo de la tecnología como en el mundo de los videojuegos donde, antiguamente, se tachaba a la gente de paria o rara por jugar o tocar un ordenador y actualmente están a la orden del día y cada vez es más seguro que cada persona haya jugado a un videojuego aunque sea una vez en su vida.

Objetivo y descripción del problema

El objetivo principal, como se menciona brevemente en la introducción, es crear un recomendador de videojuegos el cual sea capaz de decirte cuales son los más populares dentro de la página, a mayores de cuales son más o menos gustados, en función de lo que la gente vote.

Esto permitirá a la gente el hecho de ahorrarse bastante tiempo y dinero a la hora de comprar un videojuego completamente a ciegas y bastantes horas de búsqueda por páginas dentro de Google.

La idea que me ha llevado a hacer esto es por esa misma razón, ayudar a la gente de manera que sea todo de una manera más eficiente y poniéndome también en su lugar ya que veo una falta de este tipo de recomendación en ciertas páginas web.



Herramientas utilizadas

Dentro de la aplicación encontramos las partes del frontend (por así decirlo la vista de la página) y la base de datos (los datos que guardamos y como los guardamos para usarlos en la página).

Para el frontend se han utilizado lenguajes como HTML, SCSS (un sucedáneo del CSS) y Typescript. También se ha hecho uso de un framework llamado Angular. Esto hace que no sea necesario el uso de backend, si no que accedemos directamente a la base de datos por medio de un servicio auxiliar.



Todo esto se ha sido construido por medio del editor de código Visual Studio Code, donde, para crear un nuevo proyecto en angular, se hace por medio de esta instrucción:

```
PS E:\VICTOR\Escritorio\UNI\ULTIMO AÑO COSITAS\SIBI\SIBIVideojuegos> ng new proyecto-sibi
```

Al crear el proyecto, accedemos a la carpeta que acabamos de crear y hacemos uso de la siguiente instrucción para acceder a la aplicación:

```
PS E:\VICTOR\Escritorio\UNI\ULTIMO AÑO COSITAS\SIBI\SIBIVideojuegos> cd .\proyecto-sibi\  
PS E:\VICTOR\Escritorio\UNI\ULTIMO AÑO COSITAS\SIBI\SIBIVideojuegos\proyecto-sibi> npm start
```

A partir de ahí es donde he empezado a crear las distintas funcionalidades a través de las librerías que me ofrece este framework.

El uso de este framework, a pesar de que no lo había utilizado con anterioridad y ha sido algo difícil “empezar de 0”, la verdad

es que ha habido partes bastante intuitivas en las cuales había muchas páginas de ayuda para las distintas librerías y foros de ayuda para tratar el código.

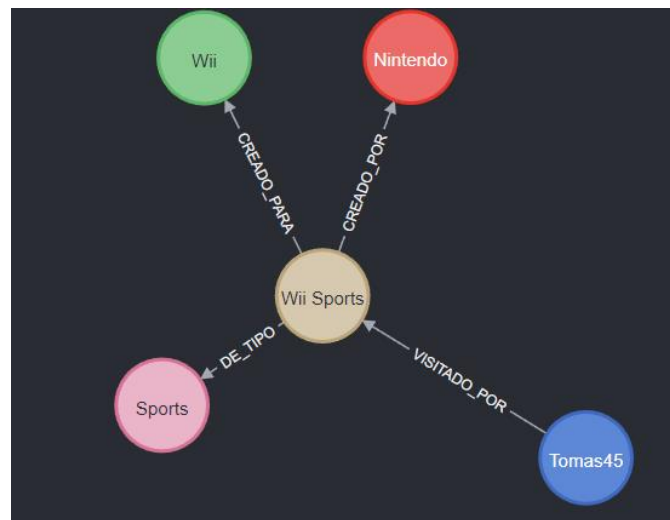
Con esto, y tras instalar el driver de neo-4j para reconocer y tratar los datos de la base de datos, estaba listo para empezar.



Para la base de datos se ha hecho uso de Neo4j, donde se ha usado el servicio de AuraDB para almacenar y tratar la base de datos, así como de su propio lenguaje de programación para grafos, Cypher. Ha sido de bastante utilidad la cantidad de cursos, foros y recursos de los que la página está dotada, algo que de primeras era bastante abrumador se convirtió en horas de buscar información sin apenas mucho tiempo entre problemas, aunque esto no quiere decir que no llevará su dificultad añadida de aprender algo nuevo.

Con esto, he sido capaz de crear mi base de datos la cual tiene como nodos principales videojuegos (donde se guardan los mismos con todos sus atributos), géneros de los mismos, plataformas, empresas y usuarios a usar la aplicación.

A continuación nos podemos encontrar con un grafo bastante reducido de lo que supone mi base de datos, con sus relaciones y sus nodos:



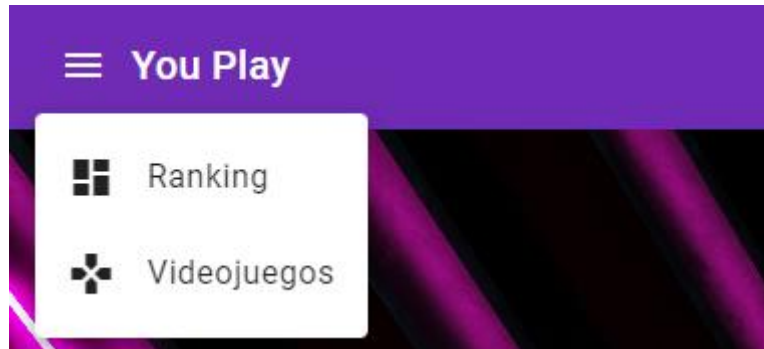
También, hablando de aplicaciones y programas utilizados, no nos podemos olvidar de GitHub, la plataforma en la cual he guardado tanto el código como los archivos necesarios para la realización de este proyecto. Esta me ha servido para guardar cambios y almacenarlos en la nube de manera tanto principal como posible backup en caso de que se diera un problema, no perder todo lo que tenía hecho de manera local.



Descripción de la aplicación

Descripción general

Como nos podemos encontrar en el inicio, estas son las páginas o vistas a las cuales se puede acceder en la aplicación:



En esta nos encontramos con un Ranking, el cual se utiliza para mostrar los juegos más populares de la página en función del algoritmo creado.

Como podemos ver es un ranking con notas del 0 al 10 de forma que en función de lo que vote la gente es lo que aquí se muestra, considerando el número de likes, dislikes y favoritos que recibe un videojuego.

La otra vista con la que cuenta la página es el listado de todos los videojuegos, Los cuales se podrán ordenar, buscar el que más te interese por el campo que quieras y realizar las acciones anteriormente mencionadas.

Frontend

Habiendo hablado un poco en general de la página, vamos a pasar a ver las vistas una por una con sus acciones.

Estas acciones serán poder ordenar los elementos según queramos en los que campos que encontramos dentro del nodo videojuegos, así como las acciones de like, dislike y favorito a la derecha de la misma.

Dentro de la pestaña videojuegos nos encontramos esto:

Nombre	Plataforma	Fecha de Lanzamiento	Género	Empresa	Puntuación de la crítica	Puntuación del usuario	Pegi
Wii Sports	Wii	2006	Sports	Nintendo	76	8	936
Super Mario Bros.	NES	1985	Platform	Nintendo			
Mario Kart Wii	Wii	2008	Racing	Nintendo	82	8.3	
Wii Sports Resort	Wii	2009	Sports	Nintendo	80	8	
Pokemon Red/Pokemon Blue	G	1996	Role-Playing	Nintendo			
Tetris	G	1989	Puzzle	Nintendo			906
New Super Mario Bros.	DS	2006	Platform	Nintendo	89	8.5	
Wii Play	Wii	2006	Misc	Nintendo	58	6.6	151
New Super Mario Bros. Wii	Wii	2009	Platform	Nintendo	87	8.4	851
Duck Hunt	NES	1984	Shooter	Nintendo			

Donde ya explicamos anteriormente los campos, estos podrán ser ordenador de manera alfabética o numérica por cada campo:

Nombre ↓	Plataforma
wwe Smackdown vs. Raw 2006	PS2
uDraw Studio	Wii

También podemos utilizar el buscador para encontrar el videojuego que queramos de manera rápida y eficiente, dependiendo de lo específicos que queramos ser:

Videojuegos

Buscar
Wii P
 🔍

Nombre	Plataforma	Fecha de Lanzamiento
Wii Play	Wii	2006
Wii Party	Wii	2010
Wii Party U	WiiU	2013
Wii Play: Motion	Wii	2011

Videojuegos

Buscar
Wii Sports Resort
 🔍

Nombre	Plataforma	Fecha de Lanzamiento
Wii Sports Resort	Wii	2009

También, como vemos en la primera imagen, nos encontramos con el sistema de likes, dislikes y favoritos, los cuales al hacer clic se guardan los datos en la base de datos y nos muestra en la propia página el propio impacto que tiene cada uno.

Nombre	Plataforma	Fecha de Lanzamiento	Género	Empresa	Puntuación de la crítica	Puntuación del usuario	Pegi			
Wii Sports	Wii	2006	Sports	Nintendo	76	8		👍 937	👎 1	❤️
Super Mario Bros.	NES	1985	Platform	Nintendo				👍	👎 151	❤️
Mario Kart Wii	Wii	2008	Racing	Nintendo	82	8.3		👍 926	👎 1	❤️
Wii Sports Resort	Wii	2009	Sports	Nintendo	80	8		👍 921	👎 1	❤️

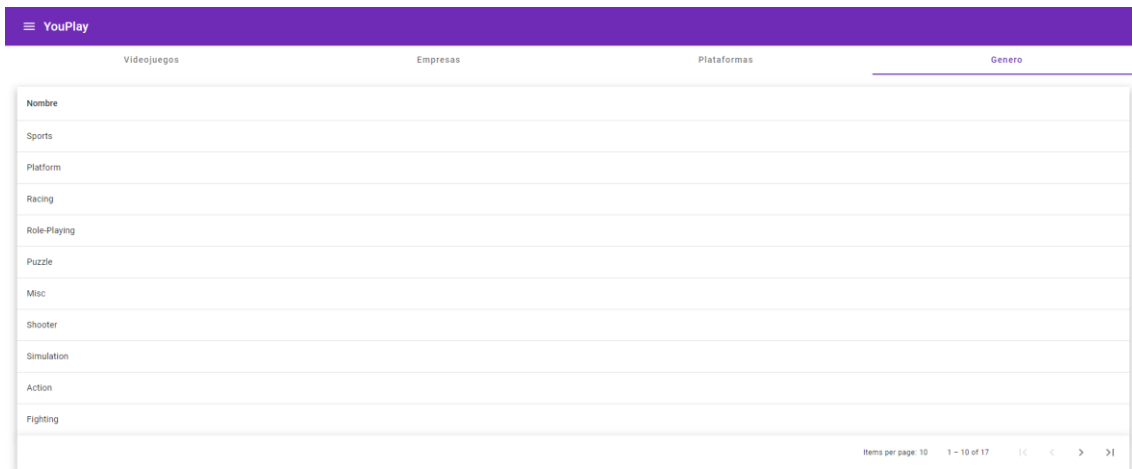
A continuación nos encontramos en el campo de las distintas empresas con las cuales tenemos que crearon ciertos videojuegos:

YouPlay			
Videojuegos	Empresas	Plataformas	Genero
Nombre			
Nintendo			
Microsoft Game Studios			
Take-Two Interactive			
Sony Computer Entertainment			
Activision			
Ubisoft			
Electronic Arts			
Bethesda Softworks			
Sega			
SquareSoft			
Items per page: 10 1 - 10 of 144 < > >			

Lo siguiente serían todas las plataformas con las que se cuenta en la base de datos para los distintos videojuegos:

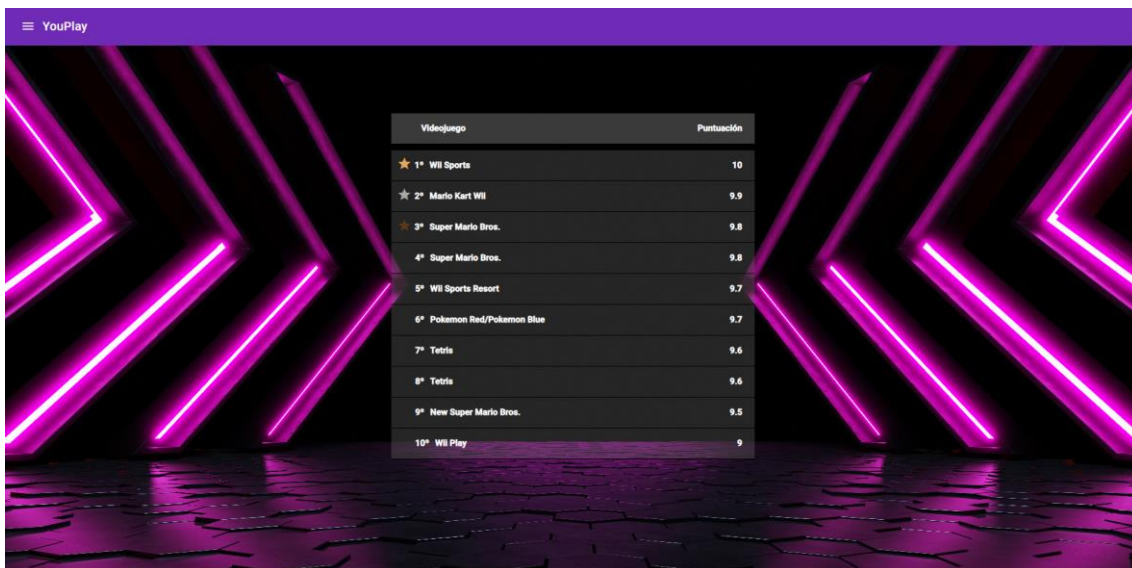
YouPlay			
Videojuegos	Empresas	Plataformas	Genero
Nombre			
Wii			
NES			
G			
DS			
X360			
PS3			
PS2			
SNES			
GBA			
3DS			
Items per page: 10 1 - 10 of 31 < > >			

Así como los distintos géneros o tipos de videojuegos que tratamos en la aplicación:



Finalmente, nos encontramos la vista del ranking, el cual en función del algoritmo utilizado que se basa en el número de likes, dislikes y favoritos que guardamos en cada videojuego nos calcula la repercusión que tiene este dentro de la página.

Aquí podemos observar la vista:



Donde tenemos clasificados los videojuegos del 1 al 10 los videojuegos más populares.

Ahora vamos a pasar a analizar el código y las funcionalidades que este implementa.

Al seguir los pasos de como iniciar el proyecto en angular, nos encontramos que este por defecto nos lo crea en el puerto 4200 para que se abra en el navegador:

```
"version": "0.2.0",
"configurations": [
  {
    "name": "ng serve",
    "type": "pwa-chrome",
    "request": "launch",
    "preLaunchTask": "npm: start",
    "url": "http://localhost:4200/"
  },
]
```

Al llamar a esta instrucción de npm start en la terminal, iniciaremos el proyecto en el puerto indicado y el navegador indicado, en este caso es para Chrome.

Posteriormente, vamos al archivo donde realizo las queries para conectar con la base de datos en neo4j, ya que así tendremos acceso a los datos y podremos trabajar con dichas queries:

```
export class DbConnectorService {
  public driver: any;
  constructor() {
    const uri = 'neo4j+s://71f265a1.databases.neo4j.io';
    const user = 'neo4j';
    const password = 'WnRif4GLOsQwEpRoGy4jWevlV5F6E_660b7jlqu_FQg';
    this.driver = neo4j.driver(uri, neo4j.auth.basic(user, password));
  }
}
```

Al ingresar los datos de la misma tendremos acceso.

Mostrando el ejemplo de los videojuegos, así es como mostramos la información en la página web y trabajamos con ella:

```
public async getVideojuegos(): Promise<videojuego[]> {
  return new Promise(async (ful) => {
    const session = this.driver.session({ database: 'neo4j' });
    let data: videojuego[] = [];
    const writeQuery = `MATCH(juego:Videojuego)
    | | | | | | | | | | RETURN juego`;
    const writeResult = await session.executeWrite((tx: any) =>
    tx.run(writeQuery, {}));
  });

  // Check the write results.
  writeResult.records.forEach((record: { get: (arg0: string) => any }) => {
    const videojuego: videojuego = record.get('juego')
    .properties as videojuego;
    data.push(videojuego);
  });
  session.close();
  ful(data);
});
}
```

Donde, con dicha query, cogemos toda la información retenida en videojuegos, para posteriormente llamar a la constante videojuegos y mostrarla en la página web. Esta lo único que hace es coger el nodo videojuego con sus atributos.

Este sería el otro ejemplo más notorio de query en el código, donde seteamos y cogemos el nombre del videojuego y los valores de like, dislike y favoritos en función de las variables del código para así mostrarlo. Este valor luego lo cogeremos para mostrar los valores, guardarlos y calcular el ranking.

```

public async setVideojuego(videojuego: videojuego): Promise<any> {
  return new Promise(async (ful) => {
    const session = this.driver.session({ database: 'neo4j' });
    const writeQuery =
      `MATCH(juego:Videojuego {nombre: `` +
        videojuego.nombre +
        ``})
      | SET juego.like = `` +
        videojuego.like +
        ``
      | SET juego.favorito = `` +
        videojuego.favorito +
        ``
      | SET juego.dislike = `` +
        videojuego.dislike +
        ``
      | RETURN juego.like`;
    const writeResult = await session.executeWrite((tx: any) =>
      tx.run(writeQuery, {}))
    );
    session.close();
    ful(true);
  });
}

```

Los demás valores de empresa, plataforma, género y usuario serían por el estilo del primero enseñado del nodo videojuego.

Ahora vamos a mostrar el caso de que en la lista de videojuegos, ya que en la de género, empresa y plataforma son por el estilo, lo que se quiera mostrar en las tablas y las acciones pertinentes.

Con la clase videojuegos.ts cogemos los datos de los campos que nos hagan falta del nodo y les asignamos un tipo:

```

export interface videojuego {
  nombre: string;
  plataforma: string;
  lanzamiento: string;
  genero: string;
  empresa: string;
  puntuacionCritica: string;
  puntuacionUsuario: string;
  pegi: string;
  like: string;
  dislike: string;
  favorito: string;
  ranking: number;
  liked: boolean;
  disliked: boolean;
  fav: boolean;
}

```

Con esto, en la clase game-list.componente.ts, creamos las columnas:

```
export class GameListComponent implements OnInit {
  public displayedColumns: string[] = [
    'nombre',
    'plataforma',
    'lanzamiento',
    'genero',
    'empresa',
    'puntuacionCritica',
    'puntuacionUsuario',
    'pegi',
    'likes',
    'dislikes',
    'favorito',
  ];
}
```

Donde se las pasamos a su HTML para que las muestre:

```
<div class="game-list">
  <mat-form-field appearance="fill">
    <mat-label>Buscar</mat-label>
    <input matInput placeholder="..." [formControl]="search" />
    <mat-icon matSuffix>search</mat-icon>
  </mat-form-field>
  <table>
    <mat-table>
      <mat-sort>
        [dataSource]="games"
        class="mat-elevation-z8"
        *ngIf="games"
      >
        <!-- Name Column -->
        <ng-container matColumnDef="nombre">
          <th mat-header-cell *matHeaderCellDef mat-sort-header>Nombre</th>
          <td mat-cell *matCellDef="let element">{{ element.nombre }}</td>
        </ng-container>
        <ng-container matColumnDef="plataforma">
          <th mat-header-cell *matHeaderCellDef mat-sort-header>Plataforma</th>
          <td mat-cell *matCellDef="let element">{{ element.plataforma }}</td>
        </ng-container>
        <ng-container matColumnDef="lanzamiento">
          <th mat-header-cell *matHeaderCellDef mat-sort-header>Lanzamiento</th>
          <td mat-cell *matCellDef="let element">{{ element.lanzamiento }}</td>
        </ng-container>
        <ng-container matColumnDef="genero">
          <th mat-header-cell *matHeaderCellDef mat-sort-header>Genero</th>
          <td mat-cell *matCellDef="let element">{{ element.genero }}</td>
        </ng-container>
        <ng-container matColumnDef="empresa">
          <th mat-header-cell *matHeaderCellDef mat-sort-header>Empresa</th>
          <td mat-cell *matCellDef="let element">{{ element.empresa }}</td>
        </ng-container>
        <ng-container matColumnDef="puntuacionCritica">
          <th mat-header-cell *matHeaderCellDef mat-sort-header>Puntuacion Critica</th>
          <td mat-cell *matCellDef="let element">{{ element.puntuacionCritica }}</td>
        </ng-container>
        <ng-container matColumnDef="puntuacionUsuario">
          <th mat-header-cell *matHeaderCellDef mat-sort-header>Puntuacion Usuario</th>
          <td mat-cell *matCellDef="let element">{{ element.puntuacionUsuario }}</td>
        </ng-container>
        <ng-container matColumnDef="pegi">
          <th mat-header-cell *matHeaderCellDef mat-sort-header>Pegi</th>
          <td mat-cell *matCellDef="let element">{{ element.pegi }}</td>
        </ng-container>
        <ng-container matColumnDef="likes">
          <th mat-header-cell *matHeaderCellDef mat-sort-header>Likes</th>
          <td mat-cell *matCellDef="let element">{{ element.likes }}</td>
        </ng-container>
        <ng-container matColumnDef="dislikes">
          <th mat-header-cell *matHeaderCellDef mat-sort-header>Dislikes</th>
          <td mat-cell *matCellDef="let element">{{ element.dislikes }}</td>
        </ng-container>
        <ng-container matColumnDef="favorito">
          <th mat-header-cell *matHeaderCellDef mat-sort-header>Favorito</th>
          <td mat-cell *matCellDef="let element">{{ element.favorito }}</td>
        </ng-container>
      </mat-sort>
    </mat-table>
  </table>
</div>
```

Y en la clase del typescript seguimos haciendo las demás funciones, como por ejemplo, que pasa al darle clic en el botón de like y que pasa cuando vuelves a hacer clic. Para el primer caso incrementas en 1 su valor y lo muestras y para el segundo todo lo contrario, lo decrementas y desclicas:

```
public likeGame(game: videojuego) {
  game.liked = true;
  game.like = (parseInt(game.like) + 1).toString();
  if (game.disliked) {
    game.dislike = (parseInt(game.dislike) - 1).toString();
    game.disliked = false;
  }
  this.service.setVideojuego(game).then((res) => {
    // this.getVideojuegos();
  });
}
```

Lo mismo pasa con las funciones de dislike y favoritos.

Aquí, podemos encontrar la disposición de la barra de búsqueda, donde buscamos los valores de los videojuegos en función de las letras que se van escribiendo y revisando el valor del nombre de los videojuegos o el campo que busques:

```
public applyFilter(search: string) {
    this.games.filter = search.trim().toLowerCase();
}

ngOnInit(): void {
    this.getVideojuegos();
    this.search.valueChanges.subscribe((res) => {
        this.applyFilter(this.search.value ? this.search.value : '');
    });
}
```

Y posteriormente lo creamos en el HTML para que nos lo muestre en la página llamando a la función anterior:

```
<mat-form-field appearance="fill">
  <mat-label>Buscar</mat-label>
  <input matInput placeholder="..." [formControl]="search" />
  <mat-icon matSuffix>search</mat-icon>
</mat-form-field>
```

Finalmente, las clases dashboard, pertenecen a la funcionalidad y vista del ranking. En esta lo primero que hacemos es llamar al videojuego con la función:

```
export class DashboardComponent implements OnInit {
    public videogames: videojuego[] = [];
    constructor(private service: DbConnectorService) {}
    ngOnInit(): void {
        this.service.getVideojuegos().then((res) => {
```

Una vez llamado, y para ese videojuego en específico, calculamos el algoritmo con las funciones anteriores que cogen sus valores de sus queries:

```

this.videogames = res.map((game: videojuego) => {
  game.ranking =
    2 * parseInt(game.favorito) +
    parseInt(game.like) -
    parseInt(game.dislike);
  if (!game.ranking || game.ranking < 0) {
    game.ranking = 0;
  }
  if (game.ranking > maxValue) {
    maxValue = game.ranking + 1;
  }
  return game;
});

```

Finalmente, este valor lo ponemos en base 10, para que nos muestre los valores del ranking del 0 al 10, y los comparamos entre ellos, para que se muestren por orden en función del valor del ranking:

```

this.videogames = res.map((game: videojuego) => {
  game.ranking = Math.round((game.ranking / maxValue) * 100) / 10;
  return game;
});
this.videogames = this.videogames.sort((a, b) => {
  if (a.ranking > b.ranking) {
    return -1;
  } else {
    return 1;
  }
});
this.videogames = this.videogames.slice(0, 10);
;

```




Todo esto lo cogemos con el html y, para las tres primeras posiciones del ranking, imprimimos las estrellas de la vista, con distintos colores especificados en el SCSS:

```

<div class="dashboard">
  <div class="ranking">
    <div class="videogame header">
      <div class="name">
        <span class="position marginLeft"></span>
        <span class="label">Videojuego</span>
      </div>
      <!--Igual cambiar aqui a la funcion para que me lo muestre el setRanking-->
      <div class="score">Puntuación</div>
    </div>
    <div class="videogame" *ngFor="let videogame of videogames; let i = index">
      <div class="name">
        <mat-icon
          *ngIf="i < 3"
          [ngClass]="{ gold: i === 0, silver: i === 1, bronze: i === 2 }"
          >star</mat-icon>
        <span class="position" [ngClass]="{ marginLeft: i > 2 }">
          <{{ i + 1 }}</span>
        <span class="label">{{ videogame.nombre }}</span>
      </div>
      <div class="score">{{ videogame.ranking }}</div>
    </div>
  </div>
</div>

```

Para que el ranking sea de color oro, plata y bronce:

```
mat-icon {  
  &.gold {  
    color:  rgb(229, 166, 94);  
  }  
  &.silver {  
    color:  rgb(169, 169, 169);  
  }  
  &.bronze {  
    color:  rgb(92, 57, 22);  
  }  
  height: 30px;  
  margin-right: 5px;  
}
```

Base de datos

Una vez hablado del cuerpo de la aplicación, vamos a pasar a hablar de la base de datos.

Lo primero que he realizado es recolectar los datos de los distintos videojuegos en un csv separado por comas. En las distintas comas encontramos el nombre y valores de las columnas que va a tener nuestra base de datos. Este lo encontré en un proyecto dentro de la página de Kaggle. En esta hay muchísima información y si no encuentras algún proyecto de scrapeo que te llame la atención o te sirva, seguro que encuentras algún foro con alguien que te ayude o alguna cuestión que te resuelva cualquier duda con tu proyecto.

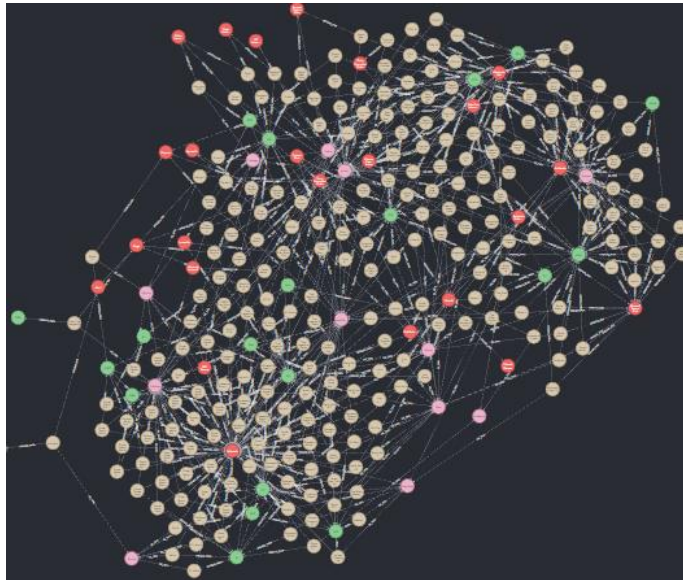
Posteriormente, creamos en el entorno de neo4j. Aquí entramos en el espacio de AuraDB, creamos una base de datos nueva, sin nada dentro, y creamos las queries. Estas serían las siguientes (Explicado en los comentarios de las imágenes):

```
1 LOAD CSV WITH HEADERS //Cargamos el archivo de videojuegos del Github
2 FROM 'https://raw.githubusercontent.com/vdelam00/VideojuegosSIBI/main/DatosVideojuegos.csv'
3 AS row
4 //Con un pequeño limite para que no colapse por problemas propios de hardware
5 WITH row LIMIT 3000
6
7 //Creamos el nodo Videojuego con los campos que estos tienen y los que necesito en un futuro
8 CREATE (juego:Videojuego{nombre:row.Name, plataforma:row.Platform, lanzamiento:row.Year_of_Release, genero:row.Genre, empresa:row.Publisher,
9 puntuacionCritica:row.Critic_Score, puntuacionUsuario:row.User_Score, pegi:row.Rating, like: 0, dislike: 0, favorito: 0, ranking: 0})
10
11 //Lo mismo que para el nodo Videojuego
12 //Creamos el nodo Empresa y su relacion
13 FOREACH(x IN(CASE WHEN row.Publisher IS NULL THEN[]ELSE[1]END) | MERGE(e:Empresa{nombre:row.Publisher}) CREATE(e)←[:CREADO_POR]-(juego))
14
15 //Creamos el nodo Plataforma y su relacion
16 FOREACH(x IN(CASE WHEN row.Platform IS NULL THEN[]ELSE[1]END) | MERGE(pl:Plataforma{nombre:row.Platform}) CREATE(pl)←[:CREADO_PARA]-(juego))
17
18 //Creamos el nodo Genero y su relacion
19 FOREACH(x IN(CASE WHEN row.Genre IS NULL THEN[]ELSE[1]END) | MERGE(g:Genero{nombre:row.Genre}) CREATE(juego)-[:DE_TIPO]→(g))
```

Y lo mismo para los nodos de los usuarios:

```
1 LOAD CSV WITH HEADERS //Cargamos el archivo de usuarios del Github
2 FROM 'https://raw.githubusercontent.com/vdelam00/VideojuegosSIBI/main/Users.csv'
3 AS row
4 WITH row LIMIT 3000
5
6 //Llamamos a los nodos Videojuego
7 MATCH (juego:Videojuego)
8
9 //Creamos los nodos usr con sus propiedades cogiendo los atributos de las rows del csv
10 CREATE (u:Usuario{nombreUsr:row.Usuario,passUsr:row.Pass})
11
12 //Relacion con visualizar un videojuego
13 CREATE (juego)←[:VISITADO_POR]-(u)
14
```

Con estas instrucciones creamos todos sus nodos y relaciones, las cuales se mostrarían de la siguiente manera en forma de grafos:



Este sería un ejemplo de los grafos, donde neo4j limita su visualización a los 300 primeros grafos.

Nos encontramos que cada videojuego tiene los atributos:

- nombre
- plataforma
- lanzamiento
- genero
- empresa
- puntuacionCritica
- puntuacionUsuario
- pegi
- like
- dislike
- favorito

Y las relaciones con los demás nodos Genero, Plataforma, Empresa y Usuario.

Explicación del algoritmo

A la hora de escoger el algoritmo, la verdad es que me decanté por algo no demasiado complejo, ya que debido a la falta de tiempo y el escaso conocimiento previo de los materiales de la asignatura, hicieron que me decantara por algo sencillo que en un futuro pudiera evolucionar en algo bastante más sofisticado. De todas formas, yo desde un principio tenía un boceto o idea de lo que quería para el proyecto final.

En un principio yo quería que el algoritmo me reconociera a los usuarios dentro de la aplicación, que, por razones anteriormente explicadas, me ha sido imposible de implementar. Con esto, cada uno tendría sus valores definidos dentro de su cuenta en la cual se gestionaría que likes, dislikes y favoritos ha marcado el usuario dentro de cada videojuego y las visitas que ha recibido cada uno de esos videojuegos. Todo eso sumado con las puntuaciones preestablecidas del csv, como del usuario y de la crítica, las cuales serían globales, obtendríamos un algoritmo que se basara en lo que le gusta o no al usuario un videojuego, su interacción con él y a mayores una nota imparcial general del propio título.

Finalmente, en el algoritmo utilizado simplemente se utilizan los likes, dislikes y favoritos de la siguiente manera:

$\text{Ranking} = 2 * \text{favorito} + (\text{likes} - \text{dislikes})$

Y este algoritmo en base 10 para que nos muestre una nota del 0 al 10 con valores decimales, donde:

- **Favorito:** Es el número de corazones que un videojuego obtiene dentro de la aplicación. Cada vez que un usuario presiona el botón del corazón, este se incrementa en 1,

donde se multiplica por 2 en el algoritmo ya que me parece el valor más importante de todos.

- **Like:** Numero de manos arriba que recibe un videojuego. Cada vez que un usuario hace clic en una mano arriba, el valor dentro del videojuego se incrementa en uno por lo que se puede sumar al favorito para ver que a un usuario le ha gustado dicho videojuego. No se multiplica por ningún valor ya que no se considera tan importante como puede ser favorito.
- **Dislike:** De la misma manera que el Like registra que a un usuario le ha gustado un juego en concreto, el dislike registra todo lo contrario. Este valor se resta al valor like para mostrar el caso de que la gente no esté contenta con el videojuego, no fuese lo que esperaba o, por la razón que sea, no le gustase.
- **Ranking:** Es el valor de la fórmula de todos los valores anteriores. Esto se muestra del 0.0 al 10.0 como se suele hacer al puntuar videojuegos y con esto mostrar cual es más o menos popular dentro de la página o con cuales los usuarios están más contentos y recomiendan.

Este algoritmo no es ni mucho menos perfecto, ya que es una versión inicial y no tiene en cuenta ni las visitas ni a los usuarios registrados, sino a todo tipo de usuarios, sin embargo, tiene mucho espacio para evolución y es un algoritmo que está bastante bien y completo para empezar.

Análisis de resultados

A continuación mostraré todos los resultados que puedan darse al ejecutar la aplicación y navegar por ella:

Ejemplo 1. General

Ejemplo 2. Videojuego específico

Ejemplo 3. Plataforma específica

Ejemplo 4. Fecha de lanzamiento del videojuego específica

Ejemplo 5. Género específico

Ejemplo 6. Empresa específica

Ejemplo 7. Videojuego no encontrado

Ejemplo 8. Recomendación por like

Ejemplo 9. Recomendación por dislike

Ejemplo 10. Recomendación por favorito

Ejemplo 11. Recomendación por todos los factores

Ejemplo 1. General:

Este es el caso general de la página, en la cual nos encontramos la lista con todos los videojuegos y demás campos.

☰ YouPlay

VideoguegosEmpresasPlataformasGenero

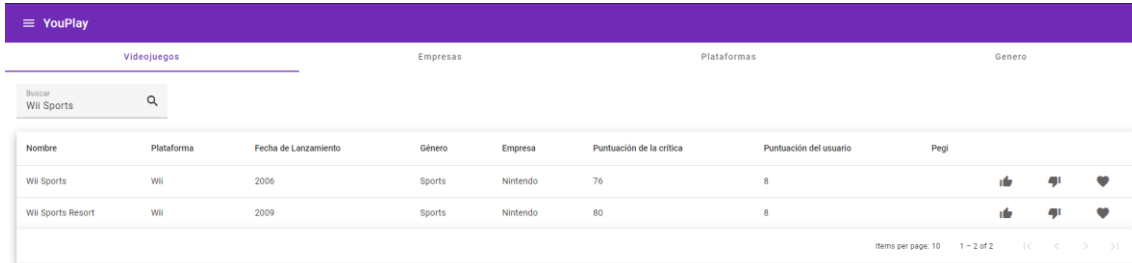
Buscar

Nombre	Plataforma	Fecha de Lanzamiento	Género	Empresa	Puntuación de la crítica	Puntuación del usuario	Pegi			
Wii Sports	Wii	2006	Sports	Nintendo	76	8		👍	👎	❤️
Super Mario Bros.	NES	1985	Platform	Nintendo				👍	👎	❤️
Mario Kart Wii	Wii	2008	Racing	Nintendo	82	8.3		👍	👎	❤️
Wii Sports Resort	Wii	2009	Sports	Nintendo	80	8		👍	👎	❤️
Pokemon Red/Pokemon Blue	G	1996	Role-Playing	Nintendo				👍	👎	❤️
Tetris	G	1989	Puzzle	Nintendo				👍	👎	❤️
New Super Mario Bros.	DS	2006	Platform	Nintendo	89	8.5		👍	👎	❤️
Wii Play	Wii	2006	Misc	Nintendo	58	6.6		👍	👎	❤️
New Super Mario Bros. Wii	Wii	2009	Platform	Nintendo	87	8.4		👍	👎	❤️
Duck Hunt	NES	1984	Shooter	Nintendo				👍	👎	❤️







Items per page: 101 - 10 of 3000<<<>>>

Ejemplo 2. Videojuego específico:

En este caso, realizamos una búsqueda por el nombre de un videojuego.



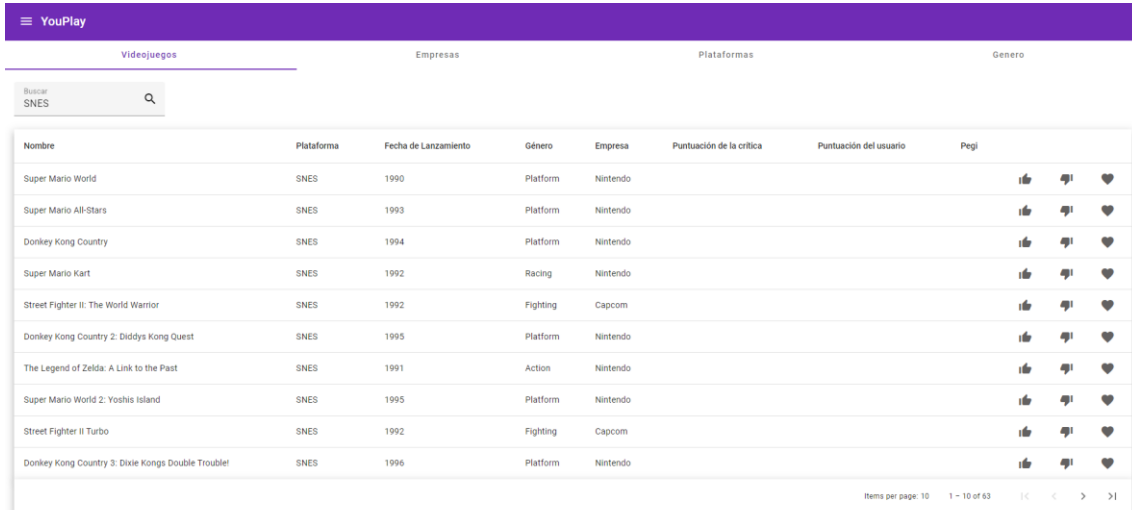
The screenshot shows the YouPlay website interface. At the top is a purple navigation bar with the 'YouPlay' logo and a hamburger menu icon. Below the navigation bar are four tabs: 'Videojuegos', 'Empresas', 'Plataformas', and 'Genero'. The 'Videojuegos' tab is selected. A search bar is located below the tabs, containing the text 'Wii Sports' and a magnifying glass icon. Below the search bar is a table with the following columns: 'Nombre', 'Plataforma', 'Fecha de Lanzamiento', 'Género', 'Empresa', 'Puntuación de la crítica', 'Puntuación del usuario', and 'Pegi'. The table contains two rows of data: 'Wii Sports' and 'Wii Sports Resort'. Each row has a thumbs up icon, a thumbs down icon, and a heart icon to its right. At the bottom right of the table, there is a pagination control showing 'Items per page: 10' and '1 - 2 of 2'.

Nombre	Plataforma	Fecha de Lanzamiento	Género	Empresa	Puntuación de la crítica	Puntuación del usuario	Pegi
Wii Sports	Wii	2006	Sports	Nintendo	76	8	  
Wii Sports Resort	Wii	2009	Sports	Nintendo	80	8	  

Podemos observar que, al realizar la búsqueda por un nombre específico, selecciona los caracteres en función de lo que escribas en el buscador. Dichos caracteres sacarán el nombre del videojuego 'Wii Sports' y todos los que contengan su nombre.

Ejemplo 3. Plataforma específica:

En este caso, vamos a seleccionar por la plataforma del videojuego, para encontrar cuales pertenecen a esta.



The screenshot shows the 'YouPlay' website interface. At the top, there's a purple navigation bar with a menu icon and the text 'YouPlay'. Below it, there are four tabs: 'Videojuegos', 'Empresas', 'Plataformas', and 'Genero'. The 'Videojuegos' tab is selected. A search bar is located below the tabs, containing the text 'SNES'. Below the search bar, there's a table with 10 rows of game data. The table has columns for 'Nombre', 'Plataforma', 'Fecha de Lanzamiento', 'Género', 'Empresa', 'Puntuación de la crítica', 'Puntuación del usuario', 'Pegi', and three icons (thumbs up, thumbs down, heart). All games listed are on the 'SNES' platform.

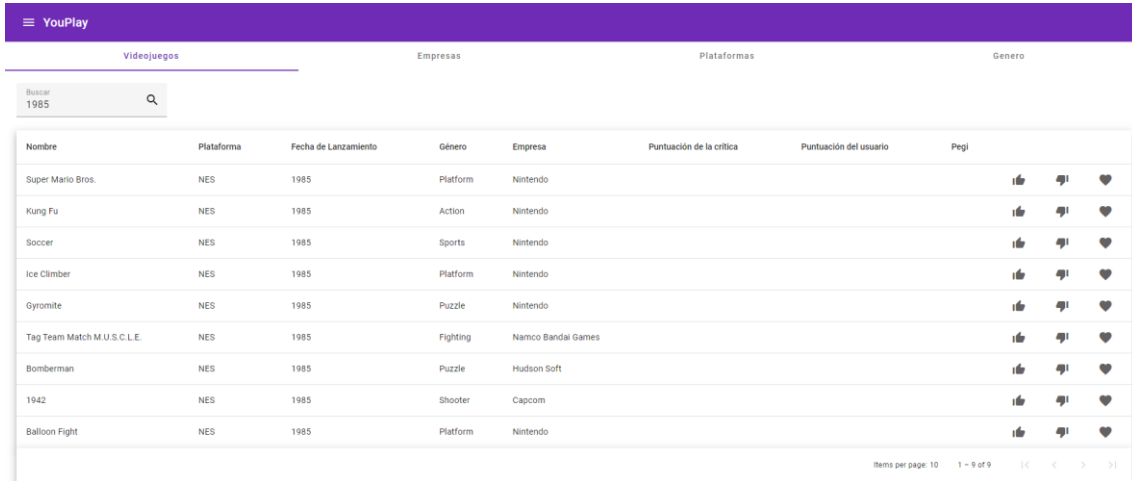
Nombre	Plataforma	Fecha de Lanzamiento	Género	Empresa	Puntuación de la crítica	Puntuación del usuario	Pegi			
Super Mario World	SNES	1990	Platform	Nintendo						
Super Mario All-Stars	SNES	1993	Platform	Nintendo						
Donkey Kong Country	SNES	1994	Platform	Nintendo						
Super Mario Kart	SNES	1992	Racing	Nintendo						
Street Fighter II: The World Warrior	SNES	1992	Fighting	Capcom						
Donkey Kong Country 2: Diddy's Kong Quest	SNES	1995	Platform	Nintendo						
The Legend of Zelda: A Link to the Past	SNES	1991	Action	Nintendo						
Super Mario World 2: Yoshi's Island	SNES	1995	Platform	Nintendo						
Street Fighter II Turbo	SNES	1992	Fighting	Capcom						
Donkey Kong Country 3: Dixie Kongs Double Trouble!	SNES	1996	Platform	Nintendo						

Items per page: 10 1 - 10 of 63 < > >|

Como vemos coge todos los videojuegos que pertenecen a la plataforma 'SNES' con todos los demás campos. Con esto, conseguimos ver que videojuegos pertenecen a esa plataforma por si nos interesa alguno.

Ejemplo 4. Fecha de lanzamiento del videojuego específica:

En este caso, vamos a seleccionar la búsqueda para que nos muestre todos los videojuegos que fueron creados en cierto año.



The screenshot shows the YouPlay website interface. At the top is a purple navigation bar with the 'YouPlay' logo. Below it is a white header with four tabs: 'Videojuegos' (selected), 'Empresas', 'Plataformas', and 'Genero'. A search bar on the left contains the text 'Buscar 1985' and a magnifying glass icon. The main content area displays a table of search results for video games released in 1985. The table has columns for 'Nombre', 'Plataforma', 'Fecha de Lanzamiento', 'Género', 'Empresa', 'Puntuación de la crítica', 'Puntuación del usuario', and 'Pegi'. There are 10 rows of results, each with a thumbs up/down icon and a heart icon on the right. At the bottom right of the table, it says 'Items per page: 10' and '1 - 9 of 9'.

Nombre	Plataforma	Fecha de Lanzamiento	Género	Empresa	Puntuación de la crítica	Puntuación del usuario	Pegi
Super Mario Bros.	NES	1985	Platform	Nintendo			👍👎❤️
Kung Fu	NES	1985	Action	Nintendo			👍👎❤️
Soccer	NES	1985	Sports	Nintendo			👍👎❤️
Ice Climber	NES	1985	Platform	Nintendo			👍👎❤️
Gyromite	NES	1985	Puzzle	Nintendo			👍👎❤️
Tag Team Match M.U.S.C.L.E.	NES	1985	Fighting	Namco Bandai Games			👍👎❤️
Bomberman	NES	1985	Puzzle	Hudson Soft			👍👎❤️
1942	NES	1985	Shooter	Capcom			👍👎❤️
Balloon Fight	NES	1985	Platform	Nintendo			👍👎❤️

Vemos que en este caso nos recoge todos los videojuegos creados en el año '1985'.

Ejemplo 5. Género específico:

En este caso, vamos a seleccionar que nos encuentre a todos los videojuegos de un género en específico:

☰ YouPlay

VideojuegosEmpresasPlataformasGenero

BuscarPuzzle

Q

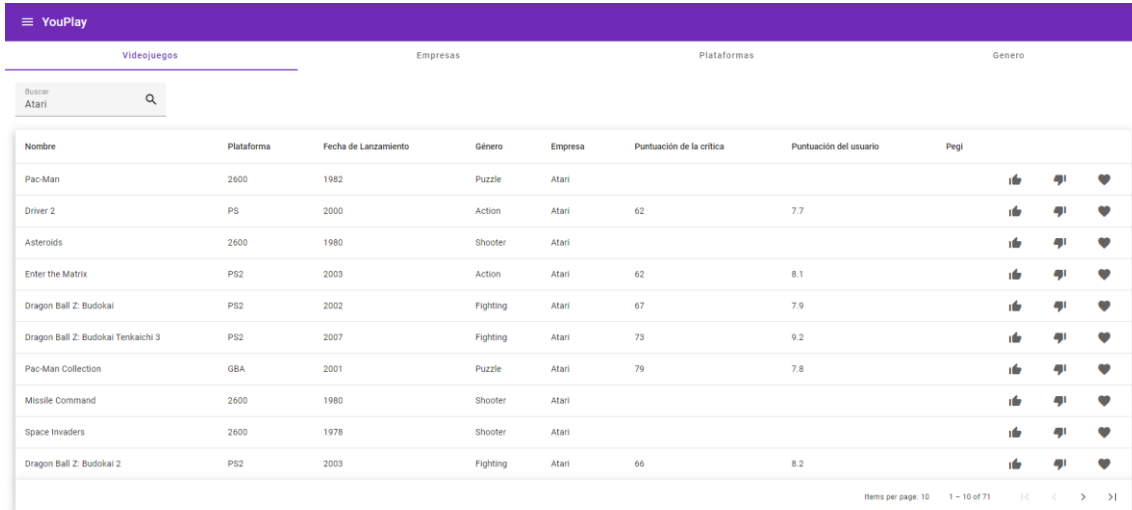
Nombre	Plataforma	Fecha de Lanzamiento	Género	Empresa	Puntuación de la crítica	Puntuación del usuario	Pegi			
Tetris	G	1989	Puzzle	Nintendo				👍	👎	❤️
Brain Age 2: More Training in Minutes a Day	DS	2005	Puzzle	Nintendo	77	7.1		👍	👎	❤️
Pac-Man	2600	1982	Puzzle	Atari				👍	👎	❤️
Tetris	NES	1988	Puzzle	Nintendo				👍	👎	❤️
Dr. Mario	G	1989	Puzzle	Nintendo				👍	👎	❤️
Professor Layton and the Curious Village	DS	2007	Puzzle	Nintendo	85	8.6		👍	👎	❤️
Dr. Mario	NES	1990	Puzzle	Nintendo				👍	👎	❤️
Professor Layton and the Diabolical Box	DS	2007	Puzzle	Nintendo	84	8.8		👍	👎	❤️
Professor Layton and the Unwound Future	DS	2008	Puzzle	Nintendo	86	9.2		👍	👎	❤️
Pac-Man Collection	GBA	2001	Puzzle	Atari	79	7.8		👍	👎	❤️

Items per page: 101 - 10 of 78<<<>>>>

Como vemos, en este caso nos cogería todos los videojuegos del género 'Puzzle'.

Ejemplo 6. Empresa específica:

En este caso, vamos a seleccionar que nos muestre todos los videojuegos que fueron creados por una empresa en específico.



The screenshot shows the YouPlay website interface. At the top, there's a purple navigation bar with the YouPlay logo and a menu icon. Below the navigation bar, there are four tabs: 'Videojuegos', 'Empresas', 'Plataformas', and 'Genero'. The 'Videojuegos' tab is selected. A search bar is located below the tabs, containing the text 'Atari'. Below the search bar, a table displays the search results. The table has columns for 'Nombre', 'Plataforma', 'Fecha de Lanzamiento', 'Género', 'Empresa', 'Puntuación de la crítica', 'Puntuación del usuario', and 'Pegi'. The table lists 12 games, all of which were created by Atari. Each game entry includes a thumbs up icon, a thumbs down icon, and a heart icon. At the bottom right of the table, there is a pagination control showing 'Items per page: 10' and '1 - 10 of 71'.

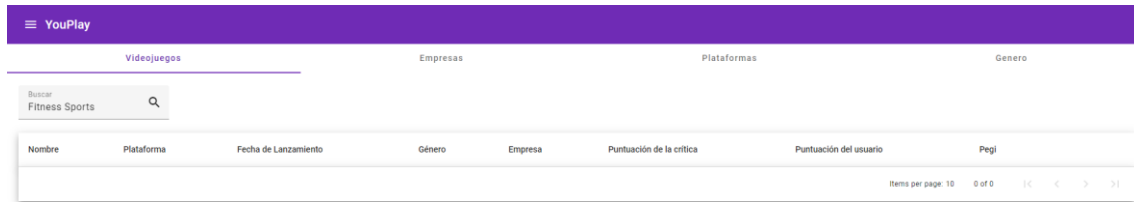
Nombre	Plataforma	Fecha de Lanzamiento	Género	Empresa	Puntuación de la crítica	Puntuación del usuario	Pegi
Pac-Man	2600	1982	Puzzle	Atari			👍 👎 ❤️
Driver 2	PS	2000	Action	Atari	62	7.7	👍 👎 ❤️
Asteroids	2600	1980	Shooter	Atari			👍 👎 ❤️
Enter the Matrix	PS2	2003	Action	Atari	62	8.1	👍 👎 ❤️
Dragon Ball Z: Budokai	PS2	2002	Fighting	Atari	67	7.9	👍 👎 ❤️
Dragon Ball Z: Budokai Tenkaichi 3	PS2	2007	Fighting	Atari	73	9.2	👍 👎 ❤️
Pac-Man Collection	GBA	2001	Puzzle	Atari	79	7.8	👍 👎 ❤️
Missile Command	2600	1980	Shooter	Atari			👍 👎 ❤️
Space Invaders	2600	1978	Shooter	Atari			👍 👎 ❤️
Dragon Ball Z: Budokai 2	PS2	2003	Fighting	Atari	66	8.2	👍 👎 ❤️

Items per page: 10 1 - 10 of 71 < > >|

Podemos observar que simplemente nos coge los videojuegos que han sido creados por la empresa 'Atari'.

Ejemplo 7. Videojuego no encontrado:

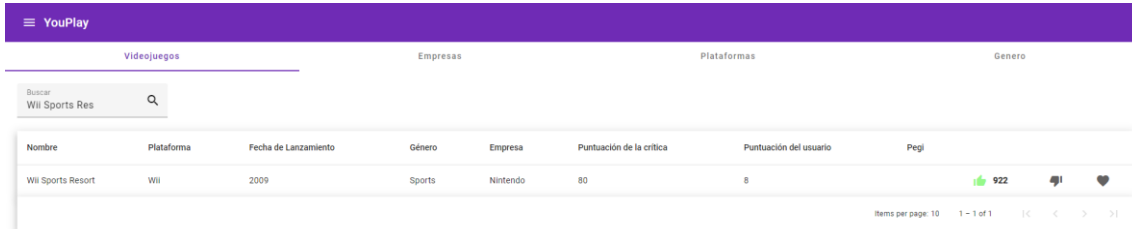
En el caso de que introduzcas un videojuego que no exista o que no contenga la base de datos, simplemente no se mostrará.



Como vemos, simplemente la lista aparecería en blanco ya que dicho videojuego no está listado.

Ejemplo 8. Recomendación por like:

En este caso, daremos like a algún videojuego que encontremos en la aplicación.



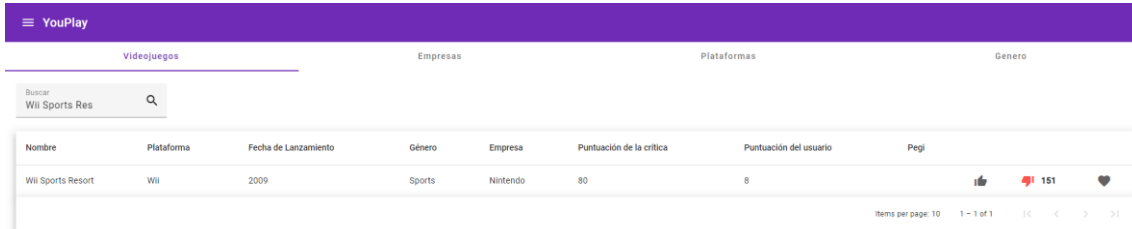
The screenshot shows the 'YouPlay' application interface. At the top, there is a purple header with the 'YouPlay' logo. Below the header, there are four tabs: 'Videojuegos', 'Empresas', 'Plataformas', and 'Genero'. The 'Videojuegos' tab is selected. Below the tabs, there is a search bar with the text 'Buscar' and 'Wii Sports Res'. Below the search bar, there is a table with the following columns: 'Nombre', 'Plataforma', 'Fecha de Lanzamiento', 'Género', 'Empresa', 'Puntuación de la crítica', 'Puntuación del usuario', and 'Pegi'. The table contains one row for 'Wii Sports Resort' with the following values: 'Wii', '2009', 'Sports', 'Nintendo', '80', '8', and 'Pegi'. To the right of the table, there is a green thumbs up icon, the number '922', a speech bubble icon, and a heart icon. At the bottom right of the table, there is a pagination bar with the text 'Items per page: 10', '1 - 1 of 1', and navigation arrows.

Nombre	Plataforma	Fecha de Lanzamiento	Género	Empresa	Puntuación de la crítica	Puntuación del usuario	Pegi
Wii Sports Resort	Wii	2009	Sports	Nintendo	80	8	Pegi

Hemos escogido el videojuego 'Wii Sports Resort', el cual hemos dado like para que estos aumenten en el videojuego y tenga más posibilidades de aparecer en el ranking.

Ejemplo 9. Recomendación por dislike:

Este será el caso contrario al like, donde se da dislike a un videojuego por la razón que sea.



The screenshot shows the YouPlay website interface. At the top is a purple navigation bar with the 'YouPlay' logo and a hamburger menu icon. Below the navigation bar are four tabs: 'Videojuegos', 'Empresas', 'Plataformas', and 'Genero'. The 'Videojuegos' tab is selected. Below the tabs is a search bar with the text 'Buscar' and 'Wii Sports Res' entered. Below the search bar is a table with the following data:

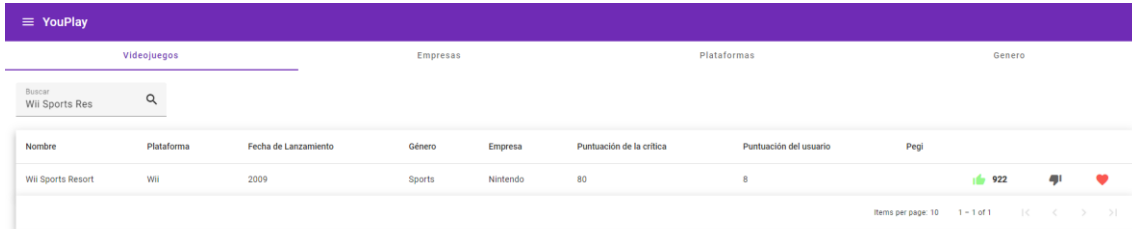
Nombre	Plataforma	Fecha de Lanzamiento	Género	Empresa	Puntuación de la crítica	Puntuación del usuario	Pegi
Wii Sports Resort	Wii	2009	Sports	Nintendo	80	8	

Below the table, there is a pagination bar showing 'Items per page: 10' and '1 - 1 of 1'. To the right of the pagination bar, there are three icons: a thumbs up icon, a thumbs down icon with the number '151' next to it, and a heart icon.

Escogiendo el videojuego anterior, al darle a dislike, hacemos que tenga menos probabilidad de que entre en el ranking.

Ejemplo 10. Recomendación por favorito:

En este caso, daremos a favoritos a un videojuego ya que nos ha encantado.



The screenshot shows the 'YouPlay' website interface. At the top is a purple navigation bar with a menu icon and the text 'YouPlay'. Below this is a horizontal menu with four tabs: 'Videojuegos' (selected), 'Empresas', 'Plataformas', and 'Genero'. A search bar is located below the tabs, containing the text 'Wii Sports Res' and a magnifying glass icon. Below the search bar is a table with the following data:

Nombre	Plataforma	Fecha de Lanzamiento	Género	Empresa	Puntuación de la crítica	Puntuación del usuario	Pegi
Wii Sports Resort	Wii	2009	Sports	Nintendo	80	8	

At the bottom right of the table, there is a green thumbs up icon with the number '922', a thumbs down icon, and a red heart icon. Below the table, there is a pagination bar that says 'Items per page: 10' and '1 - 1 of 1'.

Escogiendo el videojuego anterior, y sin necesidad de que entre en conflicto con el botón de like o dislike, vemos que hemos dado al botón de favoritos, lo cual hace que tenga muchas más probabilidades de entrar en el ranking, ya que su valor es bastante más importante que el de like o dislike.

DAFO Análisis crítico de la aplicación

En este apartado vamos a analizar las debilidades, amenazas, fortalezas y oportunidades que encontramos en la página web de una manera imparcial y lo más objetiva posible en relación a este proyecto.

De esta manera, y con la herramienta DAFO, diseñaremos un plan a futuro en función de las características mencionadas anteriormente.



- **Debilidades:** La verdad es que la debilidad más notable que veo es la calidad del algoritmo, el cual no está mal, pero me gustaría que hubiera sido algo más complejo e intrincado, pero, como bien he mencionado anteriormente, me ha sido una tarea bastante difícil dado el tiempo y la dificultad.

- **Amenazas:** La amenaza más importante ha sido la cantidad de nuevo material a la hora de realizar esta aplicación. Desde aprender a scrapear en Kaggle con Python, lo cual había hecho con anterioridad pero en Twitter y de manera algo más liviana, crear la base de datos con Neo4j y Cypher, donde las queries no es que sean complicadas del todo, pero aprender a cómo funcionaban y aprender a hacerlas en función del csv ha llevado bastante tiempo, y aprender a programar en angular, framework que no había utilizado todavía.
- **Fortalezas:** Diría que tanto la base de datos, en referencia a las queries y los datos utilizados en el csv, son bastante buenos y hacen lo que prometen, cosa de la que estoy bastante orgulloso y con unos datos bastante potentes y limpios con los que no hay errores por culpa del csv. Aparte de haber aprendido muchísimo sobre creación de bases de datos en Neo4j, así como creación de páginas web en angular, dos cosas que considero que en un futuro van a ser imprescindibles, ya que considero a Cypher como el lenguaje del futuro para la creación de un nuevo tipo de bases de datos y angular, a mi punto de vista, el mejor framework a la hora de crear una página web.

- **Oportunidades:** Al sector de los videojuegos le está pasando un poco como al sector de la informática en el ámbito laboral, pero en este caso tanto en el ámbito laboral como el de ocio, en ambos está creciendo de una manera desmesurada (con mucha visibilidad desde la expansión de los Esports, las retransmisiones en directo, los videojuegos online...).

De manera que este sector está en pleno crecimiento y páginas como IGN, Metacritic o Eurogamer, son plataformas globales que se utilizan para la búsqueda de información de videojuegos y creo que, un sistema como el mío, podría hacerle falta a páginas como estas, por eso lo veo tan buena idea.

Problemas en el desarrollo

Problema 1: A la hora de crear los usuarios con el csv en un nodo, no me reconocía bien el archivo y a la hora de crearlos no los creaba del todo bien como un nodo propio.

Solución 1: Suprimir los nodos usuarios para una futura implementación.

Problema 2: A la hora de crear filtros específicos, era una tarea algo compleja y difícil de realizar con angular a la cual, empleándole más horas, podría haberse conseguido.

Solución 2: Creador de un buscador específico el cual diferencia por campos, intentando implementar dichos filtros en un futuro.

Problema 3: No conseguir conectar bien con la base de datos.

Solución 3: Problema solventado echándole muchas horas y encontrando tanto el código para conectar bien con neo4j y su driver así como para tratar sus queries en angular. Una tarea bastante ardua.

Problema 4: Creación de atributos en función de las rows del csv los cuales dificultaban la creación del nodo videojuego.

Solución 4: Eliminación de dichos datos del csv. Estos no se iban a utilizar en el modelo final aunque podían ser útiles simplemente por la información que contenían, pero no tenían ningún tipo de funcionalidad final.

Líneas de futuro

El sector de los videojuegos es un sector que está en pleno auge y crecimiento y, como bien he dicho anteriormente, es algo que me apena no haber sido capaz de cumplir todas las ideas que tenía en mente, de todas formas bastantes de ellas se cumplieron y estoy bastante orgulloso de ello.

La creación de usuarios en los cuales se guarden los valores de a quienes dan like, dislike y favoritos, así como los videojuegos que visitan con sus géneros, los que más les interesan, el tipo, la plataforma.... Son cosas que me gustaría implementar para hacer la página más compleja.

Aparte, me gustaría que cada videojuego tuviera su propia pantalla individual en la que yo poder guardar tanto archivos multimedia como descripciones y demás datos que puedan ser de interés, como por ejemplo, un sistema de comentarios que la gente pueda leer y comentar acerca del videojuego que ellos quieran.

Aparte de los comentarios mencionados anteriormente, una especie de comentarios con feedback acerca de la página en la cual los usuarios me transmiten sus deseos e impresiones de la página.

Finalmente, me gustaría expandirme, no solo en el mundo de los videojuegos, si no en el mundo del ocio en general, pudiendo recomendar tanto películas, como libros... Un sinfín de posibilidades.

Lecciones aprendidas

La verdad es que esta puede ser la parte más gratificante del trabajo, ya que apenas entre aquí sabiendo de bases de datos, que se programan con SQL y poco más, sabiendo lo justo de páginas web, algo de HTML y Javascript pero bastante justo, y con un conocimiento bastante básico del scrapeo de datos.

Tras haber finalizado el proyecto y la asignatura, he adquirido conocimientos los cuales considero que son muy útiles para el futuro, como otra manera bastante dinámica, y fuera de la rutina y lo común, de crear bases de datos con Cypher. Crear mis propias páginas web con el framework de angular, cosa que yo creo que no me habría atrevido si no fuera por esta asignatura, ya que algo de pánico sí que les tenía a las páginas web. Y a como ser capaz de obtener información con la ayuda de Python y sabiendo donde buscar.

La verdad es que la parte de creación de la página web la considero la más frustrante pero a la vez satisfactoria de aprender y trabajar en este proyecto la verdad es que me ha servido para darme cuenta de que aunque no se me dé bien algo ni me guste, eres capaz de hacer de todo con mucho tiempo y esfuerzo, algo de lo que no ha faltado.

Bibliografía

- Csv de Kaggle:
<https://www.kaggle.com/code/kendallgillies/video-game-sales-by-genre>
- Foros de Kaggle:
<https://www.kaggle.com/discussion>
- Cursos Neo4j:
<https://graphacademy.neo4j.com>
- Visual Studio Code:
<https://code.visualstudio.com>
- Inicio en Angular:
<https://angular.io/start>
- Páginas de ayuda en Angular:
<https://material.angular.io/>
- Página para la creación de componentes en Angular:
<https://material.angular.io/components/categories>
- Neo4j AuraDB:
<https://neo4j.com/cloud/platform/aura-graph-database/>
- GitHub:
<https://github.com>
- GitHub del trabajo:
<https://github.com/vdelam00/VideojuegosSIBI>

- OSF del proyecto:
<https://osf.io/j7q8y/>
- Foros de ayuda de Neo4j:
<https://community.neo4j.com>
- Recursos y ayuda de SCSS:
<https://sass-lang.com/guide>
- Ayuda en la creación del DAFO:
<https://www.cerem.es/blog/claves-para-hacer-un-buen-dafo-o-foda#:~:text=Los%20>