

MEMORIA

GameRecommender

Sistemas de Información de Gestión y BI

David Ondicol García

Universidad de León

Índice

1. Introducción

Breve descripción sobre el sector de los videojuegos así como su importancia en la actualidad

2. Objetivo

Problema el cual se intenta tratar .

3. Herramientas

Herramientas utilizadas para la realización de la aplicación.

4. Aplicación

Código de la aplicación explicado, así como una pequeña vista del resultado final de ese código funcionando en la aplicación

4.1 Backend

4.2 Frontend

4.3 Aplicación final

5. Algoritmo

Algoritmo de recomendación utilizado para recomendar los videojuegos.

6. Análisis resultados

Muestra funcional de la aplicación.

7. Análisis crítico

Debilidades, fortalezas, amenazas y oportunidades de la aplicación.

8. Lineas de futuro

Posibles cambios a futuro de la aplicación para una posible optimización o mejora.

9. Lecciones aprendidas

Valores y conocimientos obtenidos durante la realización de la aplicación.

10. Bibliografía y Referencias

1. Introducción.

El sector de los videojuegos ha ido escalando a una velocidad increíble en las últimas dos décadas haciendo casi irreconocible si son juegos o películas. De la misma forma han cambiado sus consumidores los cuales hasta hace pocos años estaban en el concepto de que solo los niños y los 'frikis' juegan videojuegos. Como se sabe esa afirmación ya no es cierta y se pueden encontrar un alto surtido de edades entre los jugadores desde niños pequeños hasta adultos trabajadores cerca de su jubilación, y de la misma forma que hay una gran variedad de jugadores también hay una gran variedad de gustos y por lo tanto de videojuegos.

2. Objetivos.

Como menciona la introducción el problema de la actualidad es que hay demasiados tipos distintos de jugadores y de juegos, lo cual hace que sea muy difícil encontrar un juego para ti y acabes navegando en mareas de títulos sin encontrar nada a lo que jugar, después de todo los videojuegos son caros y muy poca gente puede permitirse el comprar todos los juegos que le llamen la atención.

De ese problema nace esta idea, GameRecommender, un recomendador de videojuegos con el cual puedes encontrar grandes títulos de diversas categorías a lo largo de los últimos años.



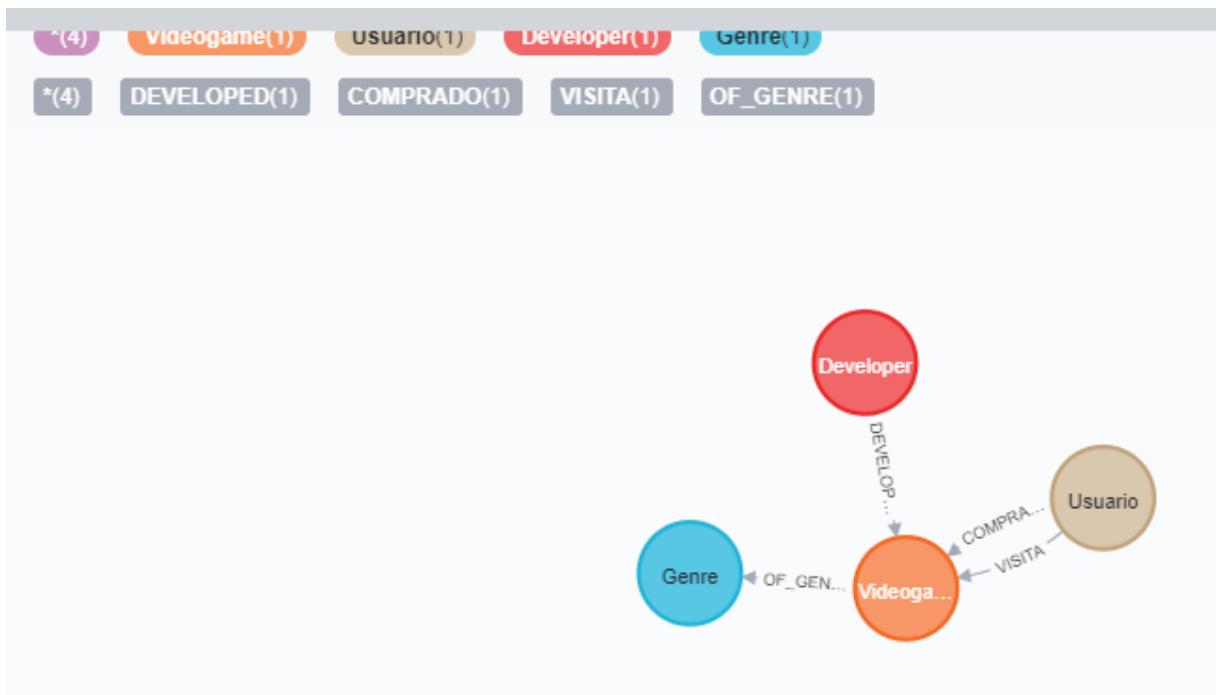
3. Herramientas.

Fue toda una experiencia la hora de elegir qué herramientas utilizar, lo único que tenía claro fue con que hacer la base de datos al saber que haría un recomendador, Neo4J.

Neo4J es una plataforma de bases de datos de grafos , al contrario que las bases de datos más comunes de sql esta se centra más en las relaciones, lo cual la hace perfecta para un sistema de recomendación. Para aprender a trabajar con esta plataforma y su lenguaje Cypher tuve que adentrarme en sus múltiples cursos los cuales puedes encontrar en su página web, los cuales aunque solo se puedan encontrar en inglés están muy bien explicados y contienen múltiples imágenes esquemáticas y problemas que te mantendrán atento a todo momento. También me ayudó mucho la guia de Cypher creada por Andrea Fernández , la cual es un gran complemento a los cursos ya que te ayuda a recordar código cypher mientras trabajas en su plataforma, en mi opinión si lo combinas con mi wiki de código de ejemplo dejado en el OSF en la bibliografía deberías tener todas las líneas para moverte cómodamente en cualquier plataforma de Neo4J.

La base de datos cargada en Neo4J fue descargada desde Kaggle y posteriormente modificada por mi, esta modificación se puede encontrar en mi github, así como otra más pequeña desarrollada para usuarios.

También dejo en el github el código cypher necesario para crear la base de datos, quiero remarcar que aunque el código debería funcionar también en la aplicación de Neo4J-desktop fue desarrollada en Neo4J-Sandbox, no debería haber ningún problema pero aun así recomiendo utilizar más el Sandbox para probar la aplicación, debido a que al ser por servidor online funciona bien desde cualquier monitor y al no necesitar instalación no deberás preocuparte por añadir programas innecesarios.



La imagen superior muestra el esquema final de la base de datos relacional de videojuegos, como se puede ver consta de Videojuegos, Usuarios, Developer y Géneros, así como sus relaciones.

Tras haber mencionado la creación de la base de datos pasemos a la aplicación como tal, la página web.

Para la creación de la página web se utilizó el framework de vue , utilizar vue me permitió más libertad a la hora de desarrollar la interfaz, una de mis actividades favoritas y a la vez más estresante debido a que al principio nada sale nunca como uno quiere. La verdad aunque diga que una página web fuese mi parte favorita no quiere decir que fuese facil, sobre todo la parte que más me preocupaba fue conectar la base de datos con el controlador de la aplicación.

No me centraré mucho ahora en ello ya que haré una explicación muy exhaustiva del código mas adelante , solo mencionar que una aplicación web se compone de dos partes:

- Backend - El apartado de la aplicación en la que se conecta con la base de datos y se trata la información que sera transmitida al frontend
- Frontend - Es la interfaz de la página web , lo que se ve cuando usas la aplicación, en este apartado se usa la información obtenida en el backend para mostrar los datos o usar en diversas funciones.

Las tecnologías utilizadas para el Backend fueron:

- JavaScript - Lenguaje con el que se escriben las funciones del Backend
- Driver de Neo4J - Permite conectar y hacer consultas con las bases de datos Neo4J
- Node.js - El Node.js se usa para crear la estructura del Backend.

Las tecnologías utilizadas para el Frontend fueron:

- Vue.js - Facilita la creación de una página web aportando una estructura sencilla para developers
- JavaScript - Creación de funciones para el Frontend
- Vuetify - Extensión de Vue.js que amplía el desarrollo de la interfaz, dando un mayor abanico de posibilidades al diseño.

4. Aplicación.

En este apartado nos dispondremos a ver el código de la aplicación de forma intensiva así como el resultado final de este.

4.1 Backend

Para la creación de este apartado he creado un pequeño backend si quisiera separarlo, consta de 2 archivos js , uno es el conector con la base de datos y otro será el controlador o handler donde recogeremos los datos de la BD.
Para esto se han instalado el driver de Neo4J y express.

BDConector.js

```
26 // npm install neo4j-driver
27 var neo4j = require('neo4j-driver'); //Driver de Neo4J
28 var driver = neo4j.driver('bolt://52.91.157.39:33058', neo4j.auth.basic('neo4j', 'wiggles-aids-extenuations
29 /*
```

Con estas dos primeras líneas hacemos lo siguiente, la primera carga el driver de neo4J alojado en los módulos, este lo necesitaremos para cargar los métodos que conectarán la base de datos con nuestro back-end .

La segunda es el método que conecta con la base de datos de neo4J, en este caso conectamos con el sandbox por lo que el bolt difiere, si fuese en desktop sería el localhost, la segunda parte serían el usuario y la contraseña para poder conectarse.

```
var session = driver.session(); // Creamos la sesión en la que pasaremos la query
module.exports = session; // La exportamos para que el controlador la pueda usar
```

Con el session creamos una sesión por la que enviaremos la query a Neo4J y el module.exports nos permite decirle que exporte el session creado en el conector a otra clase.

videojuegosControl.js

```
var session = require('../Conector/BDConector'); //Cargamos la sesion creada en el conector
```

Primero importamos los datos de BDConector para disponer de la sesión.

```
function getAll() {
    // Creo la query que buscaremos en neo4j
    var query = "MATCH (g:Genre)-[]-(v:Videogame)-[]-(d:Developer) RETURN collect(g.name) AS genre, v, col
```

Una vez cargada la sesión creamos la función getAll que usará para llamar a los videojuegos de la base de datos. Nada más hacer la función crearemos la query que se usará para decirle a Neo4J lo que quieras que te devuelva, esta query debe estar escrita en lenguaje Cypher, si quieras ver claramente lo que devuelve puedes experimentar con mi base de datos en un sandbox de Neo4J sin ningún problema.

```
const promise = new Promise(function (resolve, reject) { //Creo una promesa para la query
    session.run(query) // corremos la query en neo4j
        .then(result => {
            var Videogames = []; // creo un array donde almacenare los datos retornados
            result.records.forEach(function (record) { //con esto da la vuelta por cada resultado obtenido
                var game = record.get("v");
                var developer = record.get("d");
                var genre = record.get("genre");
                var videogame = {
                    name: game.name,
                    developer: developer.name,
                    genre: genre.name
                };
                Videogames.push(videogame);
            });
            resolve(Videogames);
        })
        .catch(error => {
            reject(error);
        });
}
```

Ahora que tenemos la query creamos la promesa para la base de datos y corremos la query en la sesión, una vez hecho esto creamos un array de videojuegos donde almacenare cada videojuego y recorro cada nodo devuelto en la sesión para almacenar sus datos.

```

    Videogames.push({ //aqui lo almacena y le da nombres a las propiedades
      name: record.get('v').properties.name,
      release_date: record.get('v').properties.release_date,
      user_score: record.get('v').properties.user_score,
      plataforma: record.get('v').properties.plataforma,
      number_players: record.get('v').properties.number_players,
      metascore: record.get('v').properties.metascore,
      genres: record.get('genre'),
      developer: record.get(['developer'])
    })
  })
}

```

Esta es la información que obtiene de cada nodo y almacena por cada videojuego, un apunte importante en esto es notar que para el nodo videojuegos para sacar sus datos es necesario el .properties , sin el no sera capaz de sacar el item correcto y solo te devolverá un error, esto viene mas explicado en las propiedades del driver de Neo4J.

```

    console.log(Videogames);
    resolve(Videogames); // la resolucion de la promesa seran los videojuegos

  }).catch(err => {
    console.log('Error inesperado ' + err);
    reject(new Error('Error inesperado ' + err.data));
  })
}
return promise // Devuelve la promesa
}

module.exports = { getAll }; // exporta la funcion getAll , estos deberian ser todos los nuevos cambios

```

Finalmente escribe todos los videojuegos por la consola y da la resolución de la promesa como Videogames. El catch esta por si sucede cualquier tipo de error podamos ver que sucede. Luego devuelve la promesa como resultado de la función.

El module exports sirve para que podamos llamar a la función getAll desde el front-end

```
[Running] node "c:\Users\david\RecomendadorVideojuegos\recomendador\src\Controlador\VideojuegosControl.js"
[
  {
    name: 'Renegade Ops',
    release_date: 'oct. 26, 2011',
    user_score: '75',
    plataforma: 'PC',
    number_players: undefined,
    metaescore: undefined,
    genres: [ 'Action' ],
    developer: [ 'Avalanche Studios' ]
  },
  {
    name: 'Brink',
    release_date: 'may. 9, 2011',
    user_score: '60',
    plataforma: 'PC',
    number_players: '16 Online',
    metaescore: undefined,
    genres: [ 'Action' ],
    developer: [ 'Splash Damage' ]
  },
  {
    name: 'Dead Space 2',
    release_date: 'Jan 25, 2011',
    user_score: '84',
    plataforma: 'PC',
    number_players: '16 Online'
  }
]
```

In 5 Col 1 Spaces:4 UTF-8

Este sería el ejemplo de los datos recibidos de la BD en neo4J

4.1 Frontend

Para la creación de este apartado he usado un solo archivo para la página, aunque el principal sea el app.vue por defecto solo ha sido modificado el HelloWorld.vue , Para hacer la pagina mas bonita se ha instalado el plugin the vuetify, el resto se puede hacer con el vue por defecto.

HelloWorld.vue

No me centraré mucho en el apartado del template html debido a que será mejor cuando se visualice más adelante en el resultado final, solo voy a enseñar los fragmentos de código para enseñar sobre el apartado visual sobre el cual he estado trabajando.

```
  <!--Este v-card es la barra de navegacion superior-->
<v-card
  app
  color="cyan"
  dark
  height="110"
> <!-- Esto podria ponerlo como una v-app-bar y se moveria , me gusta mas pero crea un espacio vacio q
<!-- Con v-card no se mueve pero no hay espacio -->
<v-toolbar dark height="110" color="cyan">
  <v-app-bar-nav-icon @click="drawer = !drawer"></v-app-bar-nav-icon>
  <v-toolbar-title id="game">
    <span style="font-size: 30pt; font-family: 'Helvetica'; "> <em>GameRecommender</em></span>
  </v-toolbar-title>
  <v-spacer></v-spacer>
  <v-btn icon>
    <v-icon>mdi-heart</v-icon>
  </v-btn>
  <v-btn icon>
    <v-icon>mdi-dots-vertical</v-icon>
  </v-btn>
  <v-btn icon>
    <v-icon>mdi-export</v-icon>
  </v-btn>
</v-toolbar>
```

Con este fragmento de código se creó la barra de cabecera de la aplicación, así como todo lo que contiene, el submenú y los emoticonos.

```
<!-- Este es el menú lateral-->
<v-navigation-drawer temporary absolute v-model="drawer" class="cyan" dark>
  <v-list-item>
    <v-list-item-avatar>
      <v-icon large>mdi-menu</v-icon>
    </v-list-item-avatar>

    <v-list-item-title>Menú</v-list-item-title>

    <v-btn icon @click.stop="drawer = !drawer">
      <v-icon>mdi-chevron-left</v-icon>
    </v-btn>
  </v-list-item>

  <v-divider></v-divider>

  <v-list dense class="pa-0" flat>
    <v-list-item-group>
      <v-list-item v-for="item in items" :key="item.title" :to="item.link">
        <v-list-item-action>
          <v-icon>{{ item.icon }}</v-icon>
        </v-list-item-action>
        <v-list-item-content>
          <v-list-item-title>{{ item.title }}</v-list-item-title>
        </v-list-item-content>
      </v-list-item>
    </v-list-item-group>
  </v-list>
</v-navigation-drawer>
```

Con este otro fragmento se crea el submenú que sale de la barra principal, actualmente no cumplen ninguna función en la aplicación pero son igualmente funcionales si se quisieran utilizar.

```
<!-- El slide de la imagen-->
<v-carousel
  cycle
  height="400"
  hide-delimiter-background
  show-arrows-on-hover
  align="top"
  justify="top"
>
  <v-carousel-item
    v-for="(slide, i) in this.slides"
    :key="i"
    :src="slide.src"
    :position="slide.position"
  >
    <v-row class="fill-height" align="center" justify="center" style="opacity:8 ; color: white; " >
      <div :class="slide.class">{{ slide.text }}</div>
    </v-row>
  </v-carousel-item>
</v-carousel>
```

Aquí es donde se muestra la imagen que aparece en la página principal, actualmente es un slide así que podríamos meter diferentes imágenes y textos , solo hay una debido a que es la que más me gustaba para la propia página y lo deje como un slide para darle más potencial y la posibilidad de dejar una imagen única o una variante

```
<!-- El buscador-->
<v-card color="grey lighten-4" height="100px" >
  <v-toolbar height="100px" color="grey darken-4" dark class="centrado2" >

    <v-toolbar-title class="centrado">
      Búsqueda por nombre:
      <v-text-field
        class="caja"
        rounded
        outlined
        dense
        label="Search"
        v-model="busqueda"
        v-on:keyup.enter="send()"
        color="cyan"

      ></v-text-field>
    </v-toolbar-title>

    <!--El boton de enviar-->
    <v-btn @click="send" class="mov" color="grey darken-1" >Buscar</v-btn>

  </v-toolbar>
</v-card>
```

Esta es la barra que funciona como buscador de los videojuegos, junto con su botón de búsqueda.

```
<!-- los combobox y los botones de filtro-->
<v-card color="grey lighten-4" height="100px">

  <v-toolbar height="100px" color="grey darken-4" dark>

    <v-toolbar-title>
      Plataforma:
      <v-combobox :items="platforms" v-model="platform" outlined dense rounded @change="cambiar"></v-combobox>
    </v-toolbar-title>

    <v-spacer></v-spacer>

    <v-toolbar-title>
      Género:
      <v-combobox :items="genres" v-model="genre" outlined dense rounded @change="cambiar"></v-combobox>
    </v-toolbar-title>

    <v-spacer></v-spacer>
    <v-spacer></v-spacer>

    <v-checkbox label="Primero los mas nuevos" v-model="modern" @change="novedad"></v-checkbox>

    <v-spacer></v-spacer>
    <v-spacer></v-spacer>

    <v-btn @click="mejor" color="grey darken-1">¡Muestrame Lo Mejor!</v-btn>

  </v-toolbar>
</v-card>
```

En este fragmento se vería todo lo que hay debajo del buscador en la página web, el filtro de plataforma, el filtro por género , el tag para que muestre los más modernos primero y finalmente el botón para recomendar juegos.

```

<!-- Este es el contenedor para los videojuegos-->
<v-content>
  <v-container grid-list-md text-xs-center fluid pa-12>
    <v-layout row wrap fill-height fill-width>
      <v-flex v-for="(videogame, index) in this.Videogames" v-bind:key="videogame.id"> <!-- Facil de en
        <v-card
          elevation="18"
          style="background: #3A1C71;
background: -webkit-linear-gradient(to right, #A195C9, #DADADA, white);
background: linear-gradient(to right, #A195C9, #DADADA, white);> class="cajaVideo"
        >
          <v-card-title>{{index + 1}}: {{videogame.name}}</v-card-title>
          <v-card-subtitle>
            Plataforma: {{videogame.plataforma}}
            <br />
            Año: {{videogame.release_date}}
            <br />
            Genero: {{String(videogame.genres)}}<!--M-->
            <br />
            Multijugador: {{videogame.number_players}}
            <br />
            <v-btn color="grey darken-1" class="botonLado" rounded @click="cuenta(index)" >¡Visitame!</v-btn>
          </v-card-subtitle>
        </v-card>
      </v-flex>
    </v-layout>
  </v-container>
</v-content>

```

Finalmente este contenedor es donde se muestra los resultados de cada juego, como se puede ver recorre un array al que llame Videogames y por cada objeto de este array mostrará su plataforma, su año de salida, el género al que pertenece , de cuantos jugadores es y un botón de visita, hablaré de la función de este botón más adelante ya que será importante para el algoritmo de recomendación.

Ahora que ya hemos mencionado los contenidos que se muestran en la pagina llega la parte más importante , el script , el apartado de data solo se mostrará para enseñar las variables de la página en la que se almacena la información, una vez hecho pasamos a las funciones que que suceden tras ejecutarse los eventos de los distintos componentes de la página.

```
<script>
import VideoJuegosControl from "../Controlador/VideojuegosControl"
export default {
  data() {
    return {
      name: "home",
      busqueda: "",
      Videogames: [],
      videogamesBack: [],
      buscadorS: [],
      drawer: false,
      genre: "Cualquiera",
      platform: "Cualquiera",
      modern: false,
      slides: [
        {
          position: "center",
          text: "¡Un nuevo mundo te espera!",
          src: "https://prommorp.ru/wp-content/uploads/2020/11/kons.jpg",
          class: "display-3"
        }
      ],
      items: [
        { title: "Inicio", icon: "mdi-home", link: "/" },
        { title: "Ayuda", icon: "mdi-help", link: "/help" }
      ],
      platforms: [
        "Cualquiera",
        "PC",
        "PlayStation",
        "PlayStation 2"
      ]
    }
  }
}
```

```

        "GameBoy Advance",
        "Nintendo 64",
        "Nintendo Wii",
        "Nintendo DS",
        "Nintendo 3DS",
        "Wii U",
        "Xbox 360",
        "Xbox One"
    ],
genres: [
    "Cualquiera", // si, todos
    "Deportes", //si Sports
    "Primera Persona", // si First-Person

```

Decidí cortar un poco por no llenar el documento con demasiada información, bastará con saber que estas son las variables que se usarán, importante el import de Videojuegos control pues necesitaremos usarlo para cargar los videojuegos de la base de datos.

```

mounted: async function(){ //async le dice que espere a q pase los datos
    this.Videogames= await VideoJuegosControl.getAll(); // await le dice que espere tambien como el async
    this.videogamesBack = this.Videogames ; //un respaldo con todos los videojuegos

    // Si no tiene el numero de jugadores les da 1 por defecto que es el minimo

    for (var v of this.Videogames) {
        if(v.number_players==undefined){
            v.number_players="1 player";
        }
        // añado las visitas a 0 de cada videojuego
        v.visitas=0;
    }

    //console.log("Videojuego numero 52: "+this.Videogames[52].plataforma ); //prueba para ver por nodo
    this.$forceUpdate(); // fuerzo la pagina a updatear para mostrar los juegos ahora cargados
    //console.log("Genero actual: " +this.genre)
},

```

Ahora empezamos con las funciones, la carne de la materia que hace que todo funcione como debe.

Esta primera función es un poco distinta de las demás y eso se debe a que no hay ningún evento que la ejecute como los demás salvo el arranque de la propia página web, primero almacena los datos obtenidos en videoJuegosControl en Videogames, despues hago un back-safe para que pueda manipular el array de videojuegos y no perder nada de información.

En ese bucle for lo que hago son dos cosas sencillas pero imprescindibles para la aplicación, asignó un número de jugadores individual para todos aquellos juegos que no lo tengan definido, esto se debe a que hay algunos juegos que no lo especifican y eso se debe a que son singleplayer, es decir de un único jugador. Lo otro es asignar el número de visitas que recibe ese videojuego , como es un valor temporal todos empiezan a 0 , como ya explique en el apartado del html esto servirá para el algoritmo de recomendación.

Finalmente se le fuerza a recargar la página, de esta forma ahora que ya tiene todos los juegos almacenados podrá mostrarlos por pantalla.

```
    },
    methods: {
        cambiar: function(){ //Metodo para la box de genero y plataforma
            this.Videogames=[]; // Vaciamos las muestras que salen por pantalla
            var indiceGenero = 0; var genero; var indice=0; var indicePlat= 0; var plataforma;

            switch(this.genre){ //un switch para el cambiar el genero de nombre para que la base de datos la
                case "Cualquiera" :
                    indiceGenero = 0;
                    break;
                case "Deportes" :
                    indiceGenero=1;
                    genero="Sports";
                    break;
                case "Primera Persona" :
```

```
//Un nuevo switch para ver la plataforma
switch(this.platform){
    case "Cualquiera":
        indicePlat=0;
        break;
    case "PC":
        indicePlat=1;
        plataforma="PC";
        break;
    case "PlayStation":
```

El método de cambiar, sirve para los filtros de los botones género y plataforma, lo primero de todo es vaciar la lista de Videogames que es lo que se muestra por pantalla , pues solo queremos que aquellos juegos que contengan el género y la plataforma marcadas se muestren. Tras ello inicializamos unas variables temporales que usaremos más tarde y entramos en los switch, dentro de los switch cambiamos los nombres para ajustarlos a los de la base de datos y a la vez le indicamos el índice, este índice servirá para decidir más adelante el tipo de operación que necesitaremos hacer.

```
//Con estos if sabra si cambia el genero, la plataforma o los dos
if(indiceGenero==0 && indicePlat==0){
    indice=0;
}
else if(indiceGenero==1 && indicePlat==0){
    indice=1;
}
else if(indiceGenero==0 && indicePlat==1 ){
    indice=2;
}
else{
    indice=3;
}
```

Aquí el índice general, como se puede ver cambiará dependiendo de las operaciones que necesitemos realizar por los filtros.

```
//En este switch se mete una vez sabe con cuanto debe operar
switch(indice){

    case 0: //En caso de que quieras ver todos vuelve al array original
        this.Videogames = this.videogamesBack;
        break;
    case 1: //Solo muestra los juegos con el genero elegido
        for (var v of this.videogamesBack) {
            if(v.genres == genero){
                //console.log(v.genres);
                this.Videogames.push(v);
            }
        }
        break;
    case 2: // muestra los juegos con la plataforma elegida
        for (var ve of this.videogamesBack) {
            if(ve.plataforma == plataforma){
                this.Videogames.push(ve);
            }
        }
        break;
    case 3: // muestra los juegos con la plataforma y generos deseados
        for (var ver of this.videogamesBack) {
            if(ver.plataforma == plataforma && ver.genres == genero){
                this.Videogames.push(ver);
            }
        }
}
```

Este será el filtro real para videojuegos, una vez sabemos las operaciones que debemos hacer se procederá a ejecutarse, en el caso 0 se cargara todo el VideojuegosBack de nuevo pues este caso se da cuando los dos filtros de

género y plataforma son cualquiera, lo cual quiere decir que nos mostrará todos los juegos.

El caso 1 se ejecuta solo cuando el género es distinto a cualquiera y la plataforma se mantiene en cualquiera, filtra únicamente por género añadiendo a videogames los videojuegos que coincidan con el género marcado. El caso 2 funciona igual que el caso 1 pero para las plataformas. El último caso es el 3 , en el cual filtrara únicamente los juegos en los que coincide la plataforma y el género.

```
novedad: function (){ //Por algun motivo al hacerlo en el caso de cualquiera cualquiera se queda asi
    //Ordena los videojuegos por fecha de mas nuevo a mas viejo

    //Solo muestra los mas nuevos si esta activo .
    if(this.modern==true){
        this.Videogames.sort(function(a,b) {
            var KeyA = new Date(a.release_date), //primer argumento
            KeyB = new Date(b.release_date); //segundo argumento
            //Ahora los compara
            if (KeyA > KeyB) return -1;
            if (KeyA < KeyB) return 1;
            return 0;
        });
    }
},
```

Esta es la función para novedad, si está marcada como verdadero realizará un sort en el que recogiendo todos los años de salida de los videojuegos los compara entre ellos , devolviendo 1 y -1 como resultados si son mayores y 0 si son iguales, esto será comprendido por la función sort de forma que reorganizará los objetos en un orden de mayor a menor fecha

```
mejor: function(){ // La funcion que usa para recomendar los juegos por nota

    //Crea la nota de cada videojuego, la cual almacena en cada videojuego y las usa para ordenarlos
    this.Videogames.forEach((a)=>{ var nota = (Number(a.user_score) + Number(a.metaescore) + Number(a.visitas)) / 3;
        a.nota = nota;
    });

    //El sort para ordenar los videojuegos por nota
    this.Videogames.sort(function(a,b) { // Coloca por nota los juegos de mayor a menor
        var KeyA = new Number(a.nota), //primer argumento
            KeyB = new Number(b.nota); //segundo argumento
        //Ahora los compara
        if (KeyA > KeyB) return -1;
        if (KeyA < KeyB) return 1;
        return 0;
    });
},
```

En esta función es donde se aplica el algoritmo de recomendación, no me pararé mucho a explicarlo ya que dejaré un apartado más adelante en el que hablare de el, solo mencionar que como se ve se utiliza un sort similar al de novedad para organizar los objetos.

```

send: function(){ // Funcion del boton send que coge los datos de la barra, es el buscador

    //Invocando antes a la funcion cambiar logro que Videogames cargue los juegos de la categoria y pl
    this.cambiar()
    //Almacena el buscador actual en un nuevo array para que busque en la categoria actual
    this.buscadorS = this.Videogames;
    //Vaciamos el nodo
    this.Videogames=[];
    //pushea todo aquel juego que contenga las palabras encontradas en el buscador, en el mismo orden
    for (var v of this.buscadorS) {
        if(this.busqueda.toUpperCase() == String(v.name).substring(0,this.busqueda.length).toUpperCase()){
            this.Videogames.push(v);
        }
    }
    //Si no encuentra ningun juego devuelve esto como si fuese un nodo de videojuegos
    if(this.Videogames.length==0){
        this.Videogames.push({
            name: "No se ha encontrado ningun Juego."
        })
    }
},

```

Esta función corresponde a el buscador de videojuegos, nada más entrar llama a cambiar para ver en qué género y plataforma deberá buscar los videojuegos, después almacena los videojuegos en buscadorS y vacía Videogames , paso que como ya explique antes sirve para cambiar los videojuegos que se mostrarán por pantalla, tras ello entra en un bucle en el que compara lo que está escrito en la barra de búsqueda con un fragmento del mismo tamaño de los nombres de los videojuegos, de esta forma aunque no acabes de escribir el nombre completo te lo mostrará en el resultado final.

```

cuenta: function(intValue){

    //Anade 1 visita por cada vez que se le da al boton del juego
    this.Videogames[intValue].visita= this.Videogames[intValue].visita+1;
    //console.log(this.Videogames[intValue].visita);
}

```

La última función, pequeña pero importante, será cargada por todos y cada uno de los botones de cada videojuego, funcionara de contador añadiendo 1 cada vez que se pulse al número de visitas de dicho videojuego.

4.1 Aplicación final

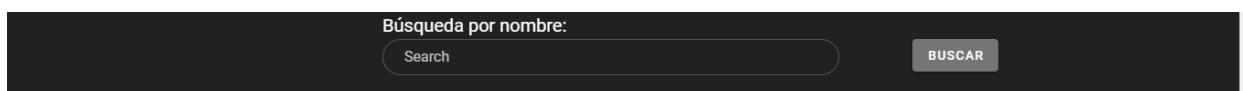
En este subapartado mostraré por encima como se ve la aplicación final, los botones , las barras y mencionare un poco por encima su funcionamiento. Este apartado puede ser observado de una forma más práctica y sencilla si se observa el video posteado junto a esta memoria, ya que podrá observar tanto los resultados como la interfaz.



La cabecera de la aplicación, consta de un submenú plegable lateral, el título/logo de la aplicación y unos iconos al final que no tienen una funcionalidad en el estado actual de la aplicación.



La imagen de entrada de la aplicación, junto a un logo que hace referencia a todos las distintas historias de cada videojuego.



La barra buscador, al introducir el nombre o parte del nombre del videojuego en ella y pulsamos el botón o intro mostrará todos los juegos en el formato de Videogames.



Los filtros de plataforma , género , el tag para novedad y el botón de recomendación, al pulsar en plataformas o género mostrará todos las posibles opciones , al pulsar primero los más nuevos aparecerá marcado y muestrame los mejor funciona como un botón.

The image shows a grid of six video game cards, each with a title, platform, release date, genre, and player count. Each card has a 'VISITAME!' button.

1: Gunslugs Plataforma: VITA Año: feb. 18, 2014 Genero: Action Multijugador: 1 player ¡VISITAME!	2: Skylanders Trap Team Plataforma: XONE Año: oct. 5, 2014 Genero: Action Multijugador: 1 player ¡VISITAME!
3: Gabrielle's Ghostly Groove 3D Plataforma: 3DS Año: oct. 4, 2011 Genero: Action Multijugador: 1 player ¡VISITAME!	4: Vessel Plataforma: PC Año: mar. 1, 2012 Genero: Action Multijugador: 1 player ¡VISITAME!
5: Guns of Icarus Online Plataforma: PC ¡VISITAME!	6: Resogun: Defenders Plataforma: PC ¡VISITAME!

Ficha de cada videojuego, con el nombre, la plataforma , el año de salida , el género y el número de jugadores, así como el botón de visita que funcionaría como un hiperlink del videojuego que redireccionará a la pagina con los datos de dicho juego, no se ha implementado esto debido al corto tiempo de desarrollo que se ha tenido, así que solo funciona como un contador de visitas.

The screenshot shows the homepage of the GameRecommender website. At the top, there is a teal header bar with the title "GameRecommendor". Below the header is a large, stylized graphic featuring a hand holding a video game controller against a background of glowing blue and white light rays. Overlaid on the graphic is the text "¡Un nuevo mundo te espera!". Below the graphic is a search bar with the placeholder "Búsqueda por nombre:" and a "Search" button. To the right of the search bar is a "BUSCAR" button.

Below the search bar, there are two dropdown menus: "Plataforma:" set to "Cualquiera" and "Género:" set to "Cualquiera". To the right of these menus are two buttons: "Primero los más nuevos" (with an unchecked checkbox) and "¡MUESTRAME LO MEJOR!".

The main content area displays six game recommendations in a grid:

- 1: Gunslugs**
Plataforma: VITA
Año: feb. 18, 2014
Genero: Action
Multijugador: 1 player
[¡VISITAME!](#)
- 2: Skylanders Trap Team**
Plataforma: XONE
Año: oct. 5, 2014
Genero: Action
Multijugador: 1 player
[¡VISITAME!](#)
- 3: Gabrielle's Ghostly Groove 3D**
Plataforma: 3DS
Año: oct. 4, 2011
Genero: Action
Multijugador: 1 player
[¡VISITAME!](#)
- 4: Vessel**
Plataforma: PC
Año: mar. 1, 2012
Genero: Action
Multijugador: 1 player
[¡VISITAME!](#)
- 5: Guns of Icarus Online**
Plataforma: PC
Año: oct. 29, 2012
[¡VISITAME!](#)
- 6: Resogun: Defenders**
Plataforma: PS4
Año: feb. 17, 2015
[¡VISITAME!](#)

Vista final de la página web

5. Algoritmo.

La elección final de un algoritmo de recomendación fue algo altamente difícil, gracias a haber conocido diversos proyectos de recomendación tuve una idea clara al principio de cómo lo quería hacer, sin embargo debido a su alta dificultad y la falta de tiempo preferí simplificarlo a un nivel aceptable.

Primero hablaré un poco del primer algoritmo pues se usó una parte de la idea para la realización del algoritmo final. El primer algoritmo de recomendación tenía dos cosas en cuenta : similitud entre usuarios y visualizaciones, la similitud entre usuarios servía para recomendar juegos teniendo en cuenta las compras de otros usuarios, cuanto más se parecieran sus compras a las de la persona que queremos recomendar más importancia tendrían los otros juegos que haya comprado y se le recomienda al comprador. Las visualizaciones serían las veces que el comprador mira un juego o muestra interés por él , esto se guardaría y aumentaría el valor del juego a recomendar.

Ahora que ya hemos hablado un poco de la primera idea del algoritmo pasaremos al que hizo el corte final, la idea del algoritmo final fue por nota y visualización. La nota que se le otorga a cada videojuego es la suma de las notas de usuario más la nota de metascore y sus visualizaciones por 2 = user_score + metascore + (visitas*2). Ahora pasaremos a explicar que es cada uno:

- **User_score** : Es la nota global otorgada por los jugadores, se realiza una suma global proporcionada por jugadores en diversas páginas, así se puede saber la opinión pública común sobre el.
- **Metascore** : Es la nota otorgada por un grupo de especialistas en análisis de videojuegos, esta nota ayuda a conocer el juego desde un punto de vista más técnico que la user_score ya que se tienen en cuenta sus cualidades más técnicas
- **Visitas** : El número de veces que un usuario inspecciona un videojuego, se le da un multiplicador para darle más importancia a las visitas , ya que se entiende que esos juegos han llamado la atención de un posible comprador.

Este algoritmo no es ni mucho menos perfecto, ya que no entiende realmente los gustos del usuario, sin embargo permite hacer recomendaciones a todo tipo de usuarios sin ningún tipo de suscripción a ninguna página, además de dar una idea de los gustos por las visitas, resultando en un algoritmo medianamente simple pero muy completo.

6. Análisis resultados.

Ahora mostraré los resultados de la aplicación ejecutando los siguientes casos:

Caso 1: Cualquiera

Caso 2: Genero específico

Caso 3: Plataforma específica

Caso 4: Género y plataforma específicos

Caso 5: Búsqueda por nombre en caso 1

Caso 6: Búsqueda por nombre en caso 4

Caso 7: Búsqueda fallida

Caso 8: Recomendación en caso 4

Caso 9: Recomendación en caso 5

Caso 10: Recomendación en caso 1

Caso 1: Cualquiera

Este es el caso por defecto, nada mas te introduces en la página web te muestra todos los juegos sin ningún tipo de filtro.

The screenshot shows a dark-themed game search interface. At the top, there is a search bar labeled "Búsqueda por nombre:" with a placeholder "Search" and a "BUSCAR" button. Below the search bar are two dropdown filters: "Plataforma:" set to "Cualquiera" and "Género:" set to "Cualquiera". To the right of these filters is a checkbox labeled "Primero los más nuevos" and a button labeled "¡MUESTRAME LO MEJOR!". The main content area displays four game cards:

- 1: Gunslugs**
Plataforma: VITA
Año: feb. 18, 2014
Genero: Action
Multijugador: 1 player
[¡VISITAME!](#)
- 2: Skylanders Trap Team**
Plataforma: XONE
Año: oct. 5, 2014
Genero: Action
Multijugador: 1 player
[¡VISITAME!](#)
- 3: Gabrielle's Ghostly Groove 3D**
Plataforma: 3DS
Año: oct. 4, 2011
Genero: Action
Multijugador: 1 player
[¡VISITAME!](#)
- 4: Vessel**
Plataforma: PC
Año: mar. 1, 2012
Genero: Action
Multijugador: 1 player
[¡VISITAME!](#)

Caso 2: Género específico

Ahora probaremos cambiando la caja de género para que muestre un solo tipo de juego , en este ejemplo los de conducción.

The screenshot shows a search interface for video games. At the top, there are filters: 'Plataforma:' set to 'Cualquiera' and 'Género:' set to 'Conducción'. There is also a checkbox for 'Primero los más nuevos' which is unchecked, and a button '¡MUESTRAME LO MEJOR!' which is greyed out. Below the filters, four game results are listed in a grid:

- 1: Forza Horizon 2**
Plataforma: XONE
Año: Sep 30, 2014
Genero: Driving
Multijugador: Up to 12
[¡VISITAME!](#)
- 2: Ride to Hell: Retribution**
Plataforma: PC
Año: jun. 24, 2013
Genero: Driving
Multijugador: 1 player
[¡VISITAME!](#)
- 3: Project CARS**
Plataforma: XONE
Año: may. 12, 2015
Genero: Driving
Multijugador: Up to 16
[¡VISITAME!](#)
- 4: Forza Motorsport 6**
Plataforma: XONE
Año: Sep 15, 2015
Genero: Driving
Multijugador: Up to 24
[¡VISITAME!](#)

Como se puede observar ahora solo los videojuegos etiquetados como juegos de conducción serán mostrados por pantalla.

Caso 3: Plataforma específica

Ahora probaremos cambiando la caja de plataforma para restringir los juegos a una única consola.

The screenshot shows a search results page for video games. At the top, there are filters for 'Plataforma' (set to 'Nintendo 3DS') and 'Género' (set to 'Cualquier'). There is also a checkbox for 'Primero los más nuevos' (Newest first) which is unchecked. A button labeled '¡MUESTRAME LO MEJOR!' (Show me the best!) is visible. The results are displayed in a grid:

- 1: Gabrielle's Ghostly Groove 3D**
Plataforma: 3DS
Año: oct. 4, 2011
Genero: Action
Multijugador: 1 player
[¡VISITAME!](#)
- 2: Fluidity: Spin Cycle**
Plataforma: 3DS
Año: Dec 27, 2012
Genero: Action
Multijugador: 1 player
[¡VISITAME!](#)
- 3: CRUSH3D**
Plataforma: 3DS
Año: mar. 6, 2012
Genero: Action
Multijugador: 1 player
[¡VISITAME!](#)
- 4: Gunman Clive 2**
Plataforma: 3DS
Año: Jan 20, 2015
Genero: Action
Multijugador: 1 player
[¡VISITAME!](#)
- 5: Fractured Soul**
Plataforma: 3DS
Año: Sep 13, 2012
[¡VISITAME!](#)
- 6: Frogger 3D**
Plataforma: 3DS
Año: Sep 20, 2011
[¡VISITAME!](#)

Al igual que el filtro de género se han dejado solo los videojuegos que pertenecen a la plataforma de 3ds.

Caso 4: Género y plataforma específicos

Ya hemos visto los casos individuales ahora veamos cómo se comporta si los dos están activos a la vez

The screenshot shows a user interface for searching games. At the top, there are two dropdown menus: 'Plataforma:' set to 'Nintendo 3DS' and 'Género:' set to 'Conducción'. To the right of these is a checkbox labeled 'Primero los mas nuevos' (First the newest) and a button labeled '¡MUESTRAME LO MEJOR!' (Show me the best). Below the filters, six game entries are listed in a grid:

- 1: Toy Stunt Bike**
Plataforma: 3DS
Año: jun. 26, 2014
Genero: Driving
Multijugador: 1 player
[¡VISITAME!](#)
- 2: Driver: Renegade**
Plataforma: 3DS
Año: Sep 6, 2011
Genero: Driving
Multijugador: 1 player
[¡VISITAME!](#)
- 3: AiRace Xeno**
Plataforma: 3DS
Año: jun. 12, 2014
Genero: Driving
Multijugador: 1 player
[¡VISITAME!](#)
- 4: Asphalt 3D**
Plataforma: 3DS
Año: mar. 22, 2011
Genero: Driving
Multijugador: 1 player
[¡VISITAME!](#)
- 5: Sonic & All-Stars Racing Transformed**
Plataforma: 3DS
Año: feb. 12, 2013
Genero: Driving
- 6: Need for Speed: The Run**
Plataforma: 3DS
Año: nov. 15, 2011
Genero: Driving

Vemos como ambos filtros se ejecutan a la vez sin ningún tipo de problema mostrando solo los juegos que cumplen las condiciones de ser de nintendo 3ds y de conducción.

Caso 5: Búsqueda por nombre en caso 1

Ahora que ya hemos visto cómo funcionan los filtros pasemos a ver como funciona el buscador, para ello utilizaremos el estado de la página por defecto y haremos una búsqueda parcial de un juego

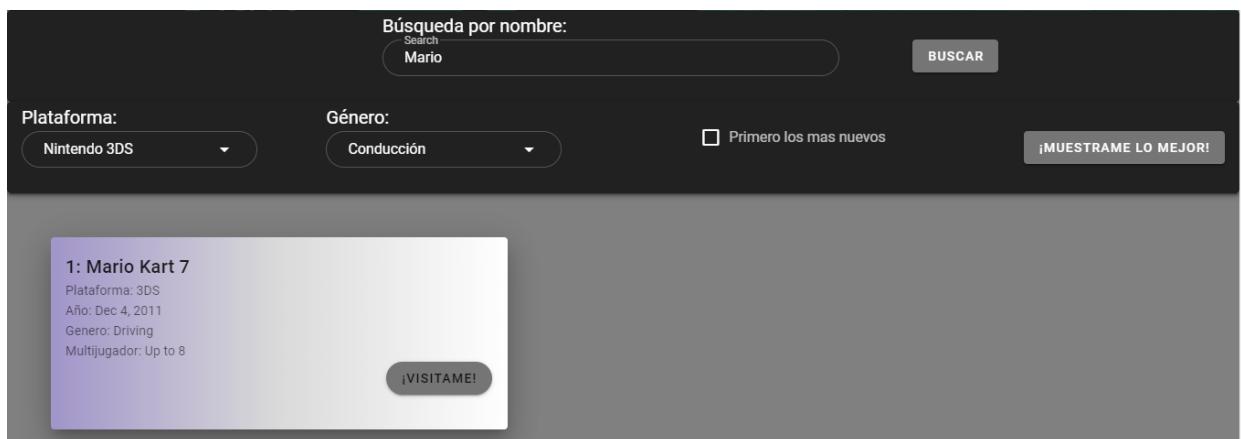
The screenshot shows a search interface with a search bar containing 'Search Mario'. Below it, there are dropdown menus for 'Plataforma' (Any) and 'Género' (Any). A checkbox for 'Primero los más nuevos' is checked. A button labeled '¡MUESTRAME LO MEJOR!' is visible. The search results are displayed in four cards:

- 1: Mario & Luigi: Dream Team**
Plataforma: 3DS
Año: Aug 11, 2013
Genero: Role-Playing
Multijugador: No Online Multiplayer
[¡VISITAME!](#)
- 2: Mario & Sonic at the Sochi 2014 Olympic Winter Games**
Plataforma: WIIU
Año: nov. 15, 2013
Genero: Sports
Multijugador: 1-4
[¡VISITAME!](#)
- 3: Mario Tennis Open**
Plataforma: 3DS
Año: may. 20, 2012
Genero: Sports
Multijugador: Up to 4
[¡VISITAME!](#)
- 4: Mario Golf: World Tour**
Plataforma: 3DS
Año: may. 2, 2014
Genero: Sports
Multijugador: Up to 4
[¡VISITAME!](#)

Como se puede observar al escribir 'Mario' en el buscador nos ha mostrado todos los juegos de cualquier plataforma y genero que empiecen por el nombre Mario.

Caso 6: Búsqueda por nombre en caso 4

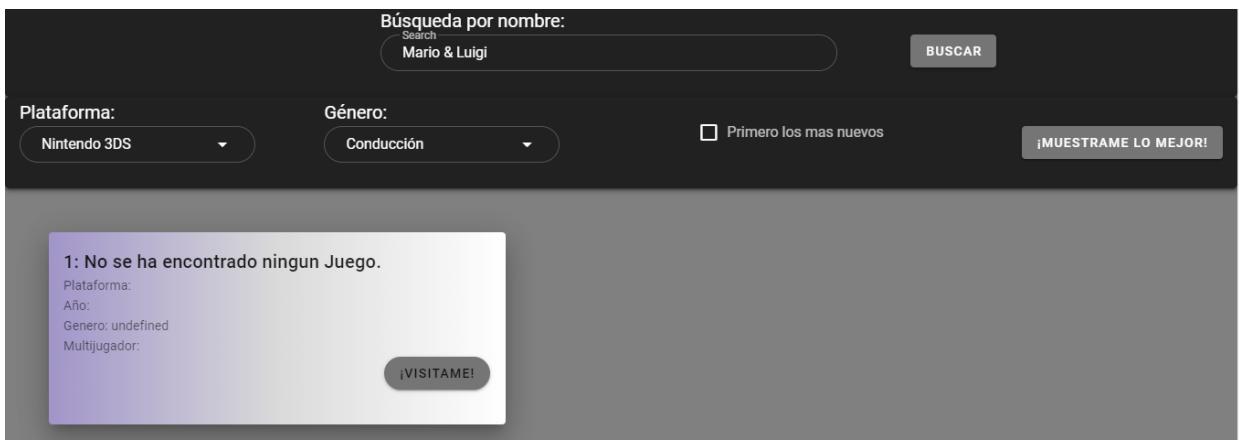
Tras haber probado el buscador en el anterior caso vamos a volver a probarlo pero ahora dándole un par de condiciones más, veamos cómo se comporta entonces



Como se puede ver escribiendo lo mismo en el buscador pero con los filtros activos ahora solo aparece un videojuego, pues es el único que cumple las 3 condiciones de todos los juegos disponibles, pertenece al género de conducción, a la plataforma de nintendo 3ds y empieza por 'Mario'.

Caso 7: Búsqueda fallida

Sería muy fácil que no encontrase nada si simplemente escribimos un nombre aleatorio, pero para ver cómo no muestra los juegos si no coinciden con los filtros escribiremos un juego que sí existe pero no pertenezca a la categoría correcta.

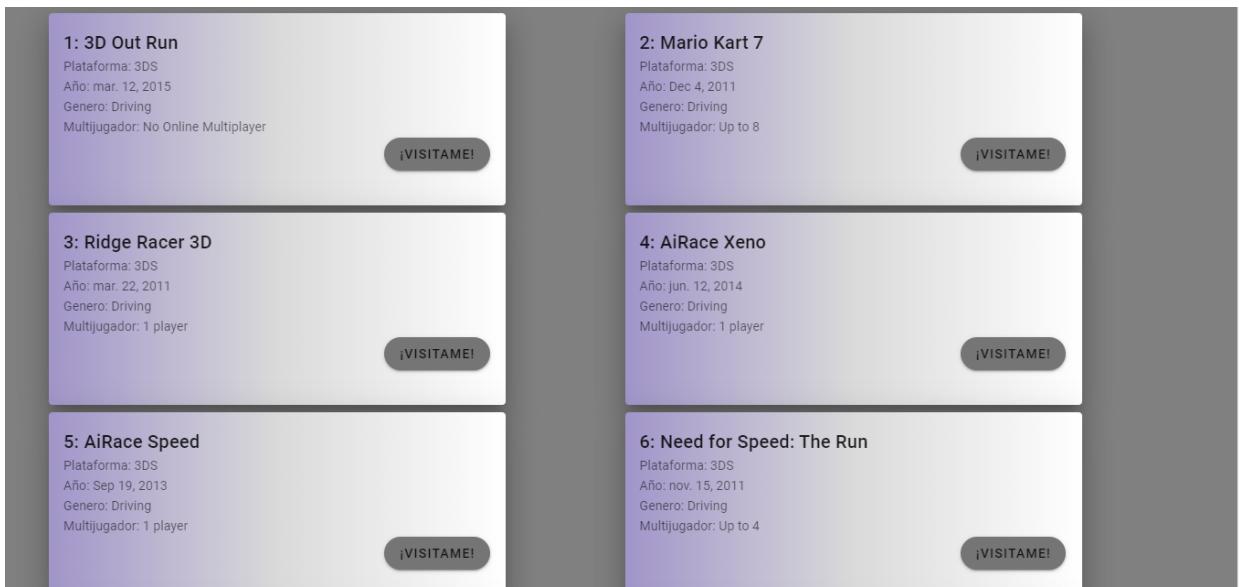


Como podemos ver el juego de Mario & Luigi si existe, pero al no pertenecer a la categoría de conducción no es capaz de encontrarlo, lo mismo si escribimos algo aleatorio



Caso 8: Recomendación en caso 4

Finalmente llegamos a la recomendación, vamos a ver como nos recomienda juegos con las condiciones del caso 4 con y sin visitas.



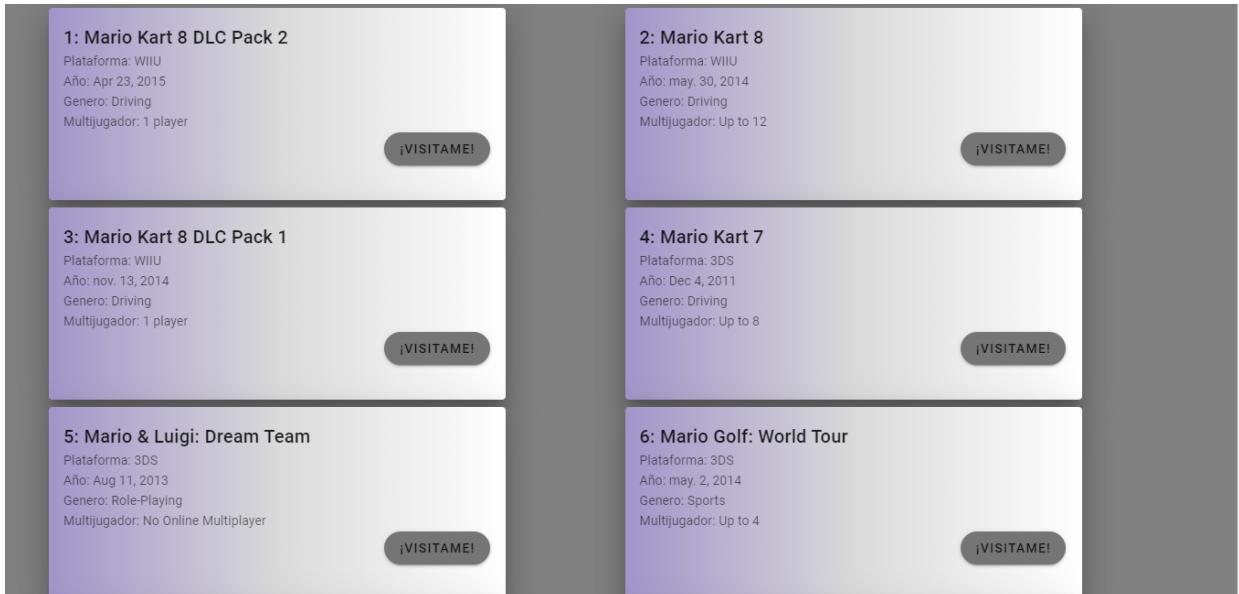
Como se puede observar el orden de los videojuegos ya no es el mismo , esto se debe a que ya son las recomendaciones, no está diciendo que de todos los juegos de esa categoría nos recomienda comprar el 3D Out Run primero, así seria si pulsamos el botón de recomendación sin más, ahora veamos como le puede afectar si hacemos múltiples visitas a los videojuegos.

1: Mario Kart 7 Plataforma: 3DS Año: Dec 4, 2011 Genero: Driving Multijugador: Up to 8 VISITAME!	2: 3D Out Run Plataforma: 3DS Año: mar. 12, 2015 Genero: Driving Multijugador: No Online Multiplayer VISITAME!
3: AiRace Xeno Plataforma: 3DS Año: jun. 12, 2014 Genero: Driving Multijugador: 1 player VISITAME!	4: AiRace Speed Plataforma: 3DS Año: Sep 19, 2013 Genero: Driving Multijugador: 1 player VISITAME!
5: Ridge Racer 3D Plataforma: 3DS Año: mar. 22, 2011 Genero: Driving Multijugador: 1 player VISITAME!	6: Need for Speed: The Run Plataforma: 3DS Año: nov. 15, 2011 Genero: Driving Multijugador: Up to 4 VISITAME!

¿Notas los cambios?, puede ser que al principio nos recomendase primero el 3D Out Run pero tras haber estado visitando los diversos videojuegos eso ha cambiado, siendo por ejemplo en este caso el Mario Kart 7 el que me recomendaria a mi.

Caso 9: Recomendación en caso 5

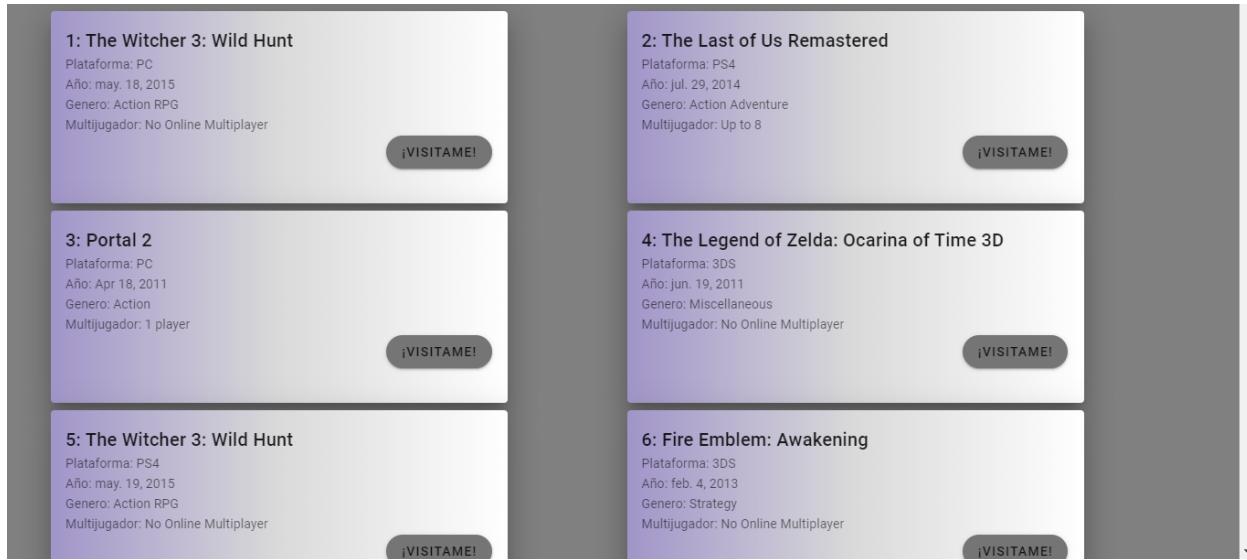
Veamos cómo se comporta el recomendador una vez se usa el buscador



Como se puede ver nos recomienda únicamente los juegos de Mario.

Caso 10: Recomendación en caso 1

Finalmente procederemos a ver que juegos nos recomienda del catálogo general,



como se puede observar nos a recomendado unos juegos increíbles de la categoría general, esto ya entra en el catálogo de la opinión personal pero como alguien que a jugado la mayoría de esos juegos puedo decir que son piezas magistrales que no dejarán a nadie indiferente.

7. Análisis crítico.

Bueno a llegado la hora de analizar esta aplicación desde un punto de vista menos personal, a todo creador le pasa, cuando haces algo desde 0 con tus propias manos siempre es difícil ver lo que no hace bien o podría ser mejor.

Por lo tanto en este apartado se intentará ver desde un punto lo más externo y profesional posible para que podamos ver los ventajas y desventajas de la aplicación, y para ello no podría haber mejor forma que con un análisis dafo.

- **Debilidades** - Me preocupa la calidad del algoritmo de recomendación, tenía planeado uno mucho mejor , sin embargo el corto tiempo y mi falta de conocimiento a la hora de usar las tecnologías hizo que no me diera tiempo a hacerlo y cambiase la idea a una más sencilla a lo largo de la práctica.
- **Amenazas** - Aprender todo lo necesario para realizar la aplicación, al principio fue muy duro pero a medida que empecé a hacer la aplicación la dificultad se suavizó, igualmente fue debido a la poca documentación que había y espero no tener que repetir una experiencia tan agotadora en un campo de tiempo tan corto nunca.
- **Fortalezas** - Diría que tanto la base de datos como la interfaz en si son mis puntos fuertes, estuve mucho tiempo pensando cómo realizar la base de datos, qué atributos necesitaría , etc .. al final todo ese tiempo se vio recompensado con una base de datos amplia y preparada con muchos

datos útiles para nuevas operaciones. En cuanto a la interfaz , estoy acostumbrado a trabajar con ellas e intente un estilo minimalista que fuese ligero y agradable para el visitante.

- ❑ **Oportunidades** - El sector de los videojuegos está en continuo crecimiento y no tiene pinta de que quiera detenerse, esto da una gran oportunidad a los recomendadores y una buena aplicación podría ayudar tanto a los clientes comprando lo que quieren y dejarlos satisfechos como a las empresas vendiendo sus productos.

8. Líneas de futuro.

Como he mencionado anteriormente no solo el sector de los videojuegos crece constantemente , yo mismo tenía una idea más grande de lo que quería hacer y no pudo ser.

Principalmente lo que haría a futuro sería cambiar el algoritmo de recomendación a mi versión anterior, para ello añadiría las funciones de suscripción de usuarios, la capacidad de loguearse, etc ... Y así podríamos usar el apartado de usuarios que al final no se pudo usar, podría hacer recomendaciones más personalizadas y variadas.

También me gustaría cambiar la visualización de los videojuegos, cada videojuego en la lista solo constaría de su nombre y la portada del juego, una vez pulsases el juego te redireccionaría a una página exclusiva de cada juego con su información más extendida.

A mayor futuro todavía se podrían incluir funciones de recomendación y muestra de datos sobre los eGames y eSports, competencias internacionales de videojuegos que cada año reciben más y más seguidores, podría añadir a la base de datos información sobre ellos y streamers , recomendando no solo videojuegos si no canales y competiciones donde se juegue a dichos juegos de manera profesional o casual para el entretenimiento.

9. Lecciones aprendidas.

He aprendido mucho más de lo que pensaba a lo largo de este curso, he aprendido lo duro que es investigar en algo por tu cuenta y en un tiempo limitado, fue estresante y aterrador la mayoría del tiempo , pero a medida que aprendía y las cosas empezaban a funcionar ese miedo se convirtió en ansia, ansia por ver la página final funcionar bien y ser bonita , con esos nuevos sentimientos cambió el enfoque del trabajo radicalmente y empezó a desarrollarse a pasos agigantados .

Aprendí a trabajar con Neo4J, asenté mis bases con Vue y aprendí sobre sistemas de recomendación, viéndolo desde el punto de vista de la experiencia diría que he ganado una gran cantidad de conocimientos. Estoy seguro de que podría haber aprendido mucho de mis compañeros, desgraciadamente la situación actual del Covid-19 a hecho que fuese más difícil comunicarme con ellos de lo que pensaba aun así estoy muy agradecido con ellos debido a que saber el estado de cómo progresaban los demás me empujaba a esforzarme más o a relajarme cuando me estresaba por no avanzar.

Trabajar en este proyecto me ayudó a entender cómo afrontar los desafíos en el futuro, que por mucho que creas haber pensado en un proyecto siempre habrá cambios, el trabajo que te presento aquí es completamente de la idea inicial que tuve, tanto el código de fondo como la interfaz , y cómo a pesar del miedo y la agitación con trabajo constante y esfuerzo todo sale adelante.

Espero que cualquier persona que lea esta memoria llegue a sentir curiosidad por esta experiencia y pruebe a programar con Neo4J, pues ha sido una experiencia invaluable.

10. Bibliografía y referencias.

10.1 Referencias

1. Cursos Neo4J: <https://neo4j.com/graphacademy/>
2. Guia Cypher Andrea Fernández:
<https://github.com/afersns16/Game4U/blob/master/Gu%C3%ADa%20de%20Cypher%20por%20Andrea%20Fern%C3%A1ndez.pdf>
3. Driver Neo4J vue: <https://github.com/adam-cowley/vue-neo4j>
4. Kaggle: <https://www.kaggle.com/datasets>
5. Node.js: <https://nodejs.org/es/>
6. Vue.js: <https://vuejs.org/>
7. Vuetify: <https://vuetifyjs.com/en/>

10.2 Bibliografía

Github del proyecto:

<https://github.com/Dondig00/RecomendadorVideojuegos>

OSF: <https://osf.io/j9dfq/>