



MEMORIA DE LA APLICACIÓN

Sistemas de Información de Gestión y BI

Se presentará la aplicación, su funcionamiento y estructura, además de su objetivo, el por qué y para qué fue creada, para terminar con un pequeño análisis crítico.

Andrea Fernández Sánchez

Universidad de León

Índice

Contenido

Índice.....	2
1. Introducción.....	3
2. Objetivos.....	4
3. Herramientas	5
4. Aplicación	8
4.1 Base de datos	8
4.2 Backend	10
4.3 Frontend	12
4.4 Funcionamiento	15
5. Análisis crítico	23
5.1 DAFO.....	23
6. Líneas de futuro	25
7. Qué he aprendido	27
8. Bibliografía y Referencias	29

1. Introducción

Es curioso cómo el sector del videojuego ha ido escalando en la última década. Son millones los jugadores en todo el mundo, y cada vez más se unen a esta forma de diversión. Dicho esto, mi punto de mira durante estos meses ha sido una parte muy concreta dentro de este mundillo: la elección de un videojuego. No es sólo la elección de una caja que contiene un cartucho o disco, sino la elección de horas de entretenimiento, las cuales muchas veces son escasas y por ello se quieren emplear en algo que realmente divierta. Hay mucho más allá de un videojuego de lo que muchos ven: una historia conmovedora, unos personajes carismáticos, un mundo que explorar... Todo ello para conseguir que el jugador se sumerja en ese universo, y para llevar esto a cabo, se necesitan empresas que lo logren, como las ya asentadas Sony o Nintendo. Así es como el sector ha ido cogiendo fuerza, a base de crearse un gigantesco club de fans que sigue creciendo día a día. Una vez establecido el contexto, comencemos con la memoria.

2. Objetivos

El problema que quise tratar tiene su origen en lo que representa un videojuego para nosotros, los jugadores. Invertir el dinero en un videojuego significa invertirlo en horas y horas de diversión, además de que últimamente son más sofisticados y, por lo tanto, más caros. Es por eso que me planteé la posibilidad de crear una aplicación que recomendase un videojuego según ciertos parámetros que introdujera el



usuario para que su compra fuera lo más satisfactoria posible y esa inversión se tradujera en muchas horas de entretenimiento. Esos parámetros de los que hablo pueden ser la plataforma o consola que posea el usuario, su género favorito o si prefiere fiarse de la crítica o de las ventas, de esta forma, la

toma de decisión será más fácil: en vez de elegir entre cientos de videojuegos, se reducirá la cifra en 10 o menos.

3. Herramientas

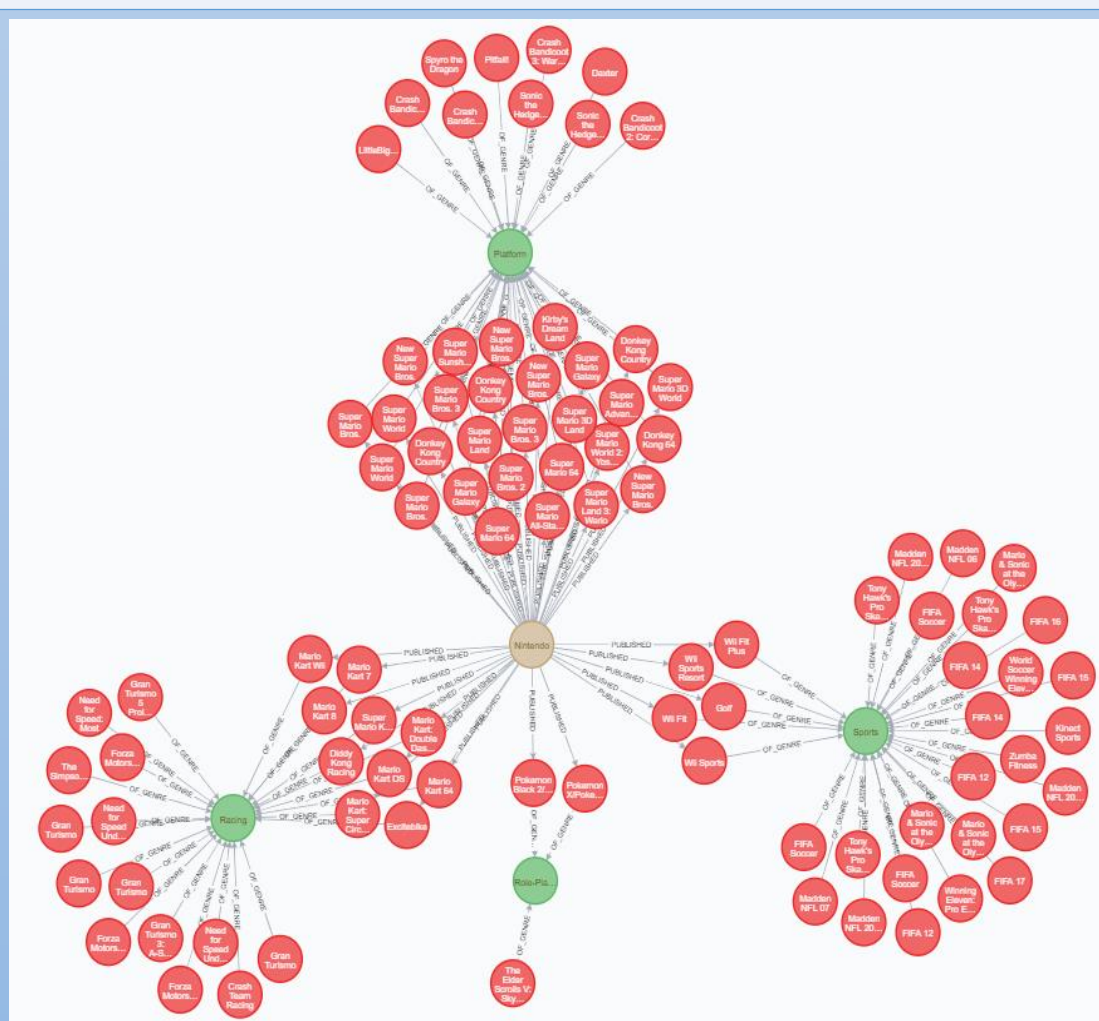
Antes de comenzar el proyecto, me aseguré de tener todo lo que necesitaba. El primer paso fue llevar a cabo una investigación sobre NEO4J, nos enfrentábamos a algo totalmente nuevo y, en ese momento, era lo único que sabíamos que iba a incluir nuestra aplicación. NEO4J ofrece recursos y documentación muy útiles, me descargué su libro “Graph Algorithms”¹ y le eché un vistazo a su canal de YouTube². Reuní toda la información en mi cuaderno de trabajo para regresar a ella en cuanto lo necesitase.

El siguiente paso fue hacerme con una buena base de datos, y que los videojuegos dentro de ella tuvieran atributos que me sirvieran y que pudieran usar los usuarios para encontrar lo que necesitaban sin usar términos demasiado complicados. Después de contrastar varias bases de datos, todas ellas de Kaggle³, me quedé con una que consideraba que tenía lo que necesitaba, además, los datos estaban muy completos, no había muchos campos nulos, aun así, corregí algún que otro parámetro.

	A	B	C	D	E	F	G	H	I	J	K	L	M
1	Name,Platform,Year_of_Release,Genre,Publisher,NA_Sales,EU_Sales,JP_Sales,Other_Sales,Global_Sales,Critic_Score,Critic_Count,User_Score,User_Count,Developer,Rating												
2	Wii Sports,Wii,2006,Sports,Nintendo,41.36,28.96,3.77,8.45,82.53,76.51,8.322,Nintendo,E												
3	Super Mario Bros.,NES,1985,Platform,Nintendo,29.08,3.58,6.81,0.77,40.24,.....												
4	Mario Kart Wii,Wii,2008,Racing,Nintendo,15.68,12.76,3.79,3.29,35.52,82.73,8.3,709,Nintendo,E												
5	Wii Sports Resort,Wii,2009,Sports,Nintendo,15.61,10.93,3.28,2.95,32.77,80,73,8,192,Nintendo,E												
6	Pokemon Red/Pokemon Blue,GB,1996,Role-Playing,Nintendo,11.27,8.89,10.22,1,31.37,.....												
7	Tetris,GB,1989,Puzzle,Nintendo,23.2,2.26,4.22,0.58,30.26,.....												
8	New Super Mario Bros.,DS,2006,Platform,Nintendo,11.28,9.14,6.5,2.88,29.8,89,65,8.5,431,Nintendo,E												
9	Wii Play,Wii,2006,Misc,Nintendo,13.96,9.18,2.93,2.84,28.92,58,41,6.6,129,Nintendo,E												
10	New Super Mario Bros. Wii,Wii,2009,Platform,Nintendo,14.44,6.94,4.7,2.24,28.32,87,80,8.4,594,Nintendo,E												
11	Duck Hunt,NES,1984,Shooter,Nintendo,26.93,0.63,0.28,0.47,28.31,.....												
12	Nintendogs,DS,2005,Simulation,Nintendo,9.05,10.95,1.93,2.74,24.67,.....												
13	Mario Kart DS,DS,2005,Racing,Nintendo,9.71,7.47,4.13,1.9,23.21,91,64,8.6,464,Nintendo,E												
14	Pokemon Gold/Pokemon Silver,GB,1999,Role-Playing,Nintendo,9.6,18.7,2.0,71.23,1,.....												
15	Wii Fit,Wii,2007,Sports,Nintendo,8.92,8.03,3.6,2.15,22.7,80,63,7.7,146,Nintendo,E												
16	Kinect Adventures!,X360,2010,Misc,Microsoft Game Studios,15.4,89,0.24,1.69,21.81,61,45,6.3,106,Good Science Studio,E												
17	Wii Fit Plus,Wii,2009,Sports,Nintendo,9.01,8.49,2.53,1.77,21.79,80,33,7.4,52,Nintendo,E												
18	Grand Theft Auto V,PS3,2013,Action,Take-Two Interactive,7.02,9.09,0.98,3.96,21.04,97,50,8.2,3994,Rockstar North,M												
19	Grand Theft Auto: San Andreas,PS2,2004,Action,Take-Two Interactive,9.43,0.4,0.41,10.57,20.81,95,80,9,1588,Rockstar North,M												
20	Super Mario World,SNES,1990,Platform,Nintendo,12.78,3.75,3.54,0.55,20.61,.....												
21	Brain Age: Train Your Brain in Minutes a Day,DS,2005,Misc,Nintendo,4.74,9.2,4.16,2.04,20.15,77,58,7.9,50,Nintendo,E												
22	Pokemon Diamond/Pokemon Pearl,DS,2006,Role-Playing,Nintendo,6.38,4.46,6.04,1.36,18.25,.....												
23	Super Mario Land,GB,1989,Platform,Nintendo,10.83,2.71,4.18,0.42,18.14,.....												
24	Super Mario Bros. 3,NES,1988,Platform,Nintendo,9.54,3.44,3.84,0.46,17.28,.....												
25	Grand Theft Auto V,X360,2013,Action,Take-Two Interactive,9.66,5.14,0.06,1.41,16.27,97,58,8.1,3711,Rockstar North,M												
26	Grand Theft Auto: Vice City,PS2,2002,Action,Take-Two Interactive,8.41,5.49,0.47,1.78,16.15,95,62,8.7,730,Rockstar North,M												
27	Pokemon Ruby/Pokemon Sapphire,GBA,2002,Role-Playing,Nintendo,6.06,3.9,5.38,0.5,15.85,.....												
28	Brain Age 2: More Training in Minutes a Day,DS,2005,Puzzle,Nintendo,3.43,5.35,5.32,1.18,15.29,77,37,7.1,19,Nintendo,E												
29	Pokemon Black/Pokemon White,DS,2010,Role-Playing,Nintendo,5.51,3.17,5.65,0.8,15.14,.....												
30	Gran Turismo 3: A-Spec,PS2,2001,Racing,Sony Computer Entertainment,6.85,5.09,1.87,1.16,14.98,95,54,8.4,314,Polyphony Digital,E												
31	Call of Duty: Modern Warfare 3,X360,2011,Shooter,Activision,9.04,4.24,0.13,1.32,14.73,88,81,3.4,8713,"Infinity Ward, Sledgehammer Games",M												
32	Pokémon Yellow: Special Pikachu Edition,GB,1998,Role-Playing,Nintendo,5.89,5.04,3.12,0.59,14.64,.....												
33	Call of Duty: Black Ops 3,PS4,2015,Shooter,Activision,6.03,5.86,0.36,2.38,14.63,.....												
34	Call of Duty: Black Ops,X360,2010,Shooter,Activision,9.7,3.68,0.11,1.13,14.61,87,89,6.3,1454,Treyarch,M												
35	Pokemon X/Pokemon Y,3DS,2013,Role-Playing,Nintendo,5.28,4.19,4.35,0.78,14.6,.....												
36	Call of Duty: Black Ops II,PS3,2012,Shooter,Activision,4.99,5.73,0.65,2.42,13.79,83,21,5.3,922,Treyarch,M												
37	Call of Duty: Black Ops II,X360,2012,Shooter,Activision,8.25,4.24,0.07,1.12,13.67,83,73,4.8,2256,Treyarch,M												
38	Call of Duty: Modern Warfare 2,X360,2009,Shooter,Activision,8.52,3.59,0.08,1.28,13.47,94,100,6.3,2698,Infinity Ward,M												

La estructura de la base de datos ‘en crudo’

Una vez elegida, tendría que introducirla en NEO4J, el sistema que debíamos usar para guardarla. NEO4J tiene su propio lenguaje, CYPHER, el cual tuve que aprender. Mientras estaba haciendo pruebas y escribiendo consultas, se me ocurrió la idea de realizar mi propia guía de CYPHER, con código que probablemente me serviría en el futuro, y así fue. La guía recoge prácticamente todas las operaciones que se pueden realizar en los nodos (añadir, quitar, crear relaciones, obtener datos...), además de algunos ejemplos que yo misma había realizado y probado. Normalmente, cuando se programa, ciertas sentencias se suelen repetir, así que recurría a la guía para no tener que volver a formular de nuevo una consulta que ya había hecho anteriormente o a la que había que cambiarle un par de palabras.



Parte de la base de datos en NEO4J

Cuando ya tenía la base de datos en NEO4J, lo siguiente era pensar cómo hacer la aplicación en sí. Mi primera opción era hacerlo como aplicación de Java, pero, a decir verdad, no me llamaba lo suficiente la atención, además Java tampoco me permitiría

```
Node {
  identity: Integer { low: 793, high: 0 },
  labels: [ 'Videogame' ],
  properties:
    { rating: 'E10+',
      name: 'Portal 2',
      metascore: '95',
      user_score: '90',
      year: Integer { low: 2011, high: 0 },
      platform: 'PC' } }
Node {
  identity: Integer { low: 794, high: 0 },
  labels: [ 'Videogame' ],
  properties:
    { name: 'The Elder Scrolls V: Skyrim',
      rating: 'M',
      metascore: '94',
      year: Integer { low: 2011, high: 0 },
      user_score: '82',
      platform: 'PC' } }
Node {
  identity: Integer { low: 796, high: 0 },
  labels: [ 'Videogame' ],
  properties:
    { rating: 'T',
      name: 'Batman: Arkham City',
      metascore: '91',
      user_score: '87',
```

Así empezaron siendo los datos devueltos

desarrollar mi creatividad en lo que a interfaz se refiere. Entonces tomé la decisión de hacerlo como una aplicación Web. Tenía mucha práctica, me permitiría mejorar y podría hacer una interfaz llamativa, lo único que me faltaba era el Driver de NEO4J⁴ para hacerlo de esta manera. Un driver es el conjunto de librerías que permite obtener los datos de la base de datos para llevarlos, en mi caso, a la aplicación Web. Por suerte, NEO4J dispone de una gran cantidad de Drivers⁵ y no tuve mucho problema en comprender su funcionamiento, así que pude empezar a hacer pequeños test de devolución de datos con sentencias simples.

Una aplicación Web se compone de dos partes:

- **Backend** – Hace las consultas a la base de datos, y envía los resultados al Frontend.
- **Frontend** – Pide al Backend que haga consultas para poder mostrar los datos. Es la interfaz como tal.

Las tecnologías que usé para el Backend fueron:

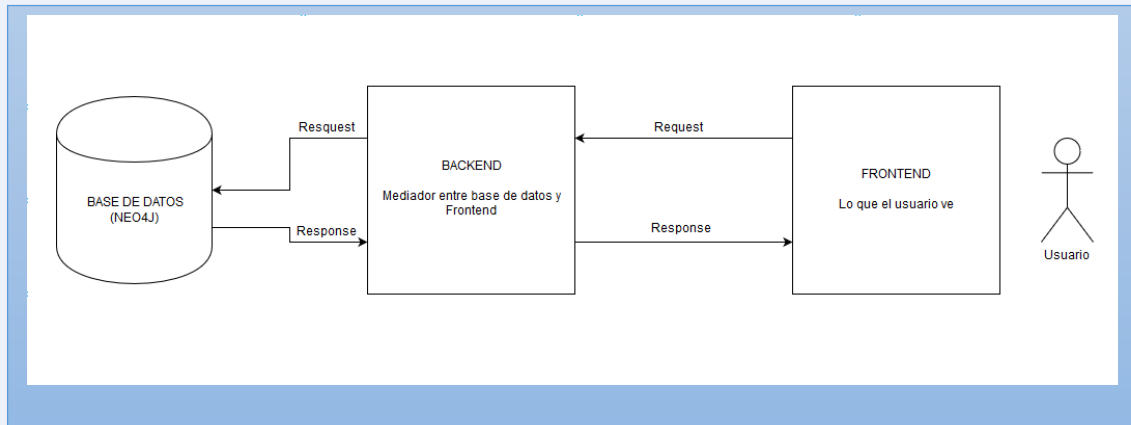
- **Node.js** – Establece la estructura del Backend⁶.
- **Express** – Permite recibir consultas del Frontend⁷.
- **JavaScript** – Le da funcionalidad al Backend.
- **Driver de NEO4J** – Permite hacer consultas a la base de datos de NEO4J⁴.

Las que usé para el Frontend fueron:

- **JavaScript** – Da funcionalidad al Frontend.
- **Vue.js** – Aporta la estructura y parte de la funcionalidad de la interfaz⁸.
- **Vuetify** – Es una extensión de Vue.js centrado en la parte visible de la interfaz⁹.
- **Vue-Resource** – Permite hacer consultas al Backend¹⁰.
- **Electron** – Se usa para transformar la aplicación Web en una aplicación instalable, es decir, se podría acceder a ella mediante un archivo ejecutable y no en el navegador¹¹.

4. Aplicación

El funcionamiento de la aplicación se basa en este esquema:



El usuario interactúa con el Frontend, el cual contiene los ficheros necesarios que componen la interfaz de la aplicación y su funcionalidad. Cuando el usuario quiere que se le muestren los videojuegos, es decir, cuando clique en “Buscar”, el Frontend enviará una petición de datos hacia el Backend, que se encargará de obtener esos datos requeridos de NEO4J. En cuanto obtenga todo lo que necesite, los enviará al Frontend, que los mostrará al usuario. Todo esto se produce en un máximo de 1 segundo, es proceso es muy rápido (también depende del rendimiento del ordenador que se tenga). Podríamos decir que este procedimiento crea una cadena de dependencia, es decir, el Frontend depende del Backend, y éste de la base de datos, por eso es muy importante no olvidarse de comprobar que NEO4J esté activo, error que cometí en numerosas ocasiones.

Ahora que ya está aclarado el funcionamiento del sistema, creo conveniente dividir este bloque en tres apartados: Base de datos, Backend y Frontend. Al final mostraré casos de funcionamiento que servirán de ejemplo para observar cómo trabajan estos componentes de forma perfectamente coordinada. Dicho esto, empezaremos con el Backend.

4.1 Base de datos

Mi base de datos está compuesta por un total de 1067 nodos y 1999 relaciones. Los nodos están repartidos entre videojuegos, publishers y géneros, y las relaciones

pueden ser de tipo “PUBLISHED” o “OF_GENRE”. Los videojuegos son los nodos más numerosos que hay, siendo 1000 de 1067. Decidí que incluiría 1000 porque no iba a necesitar más de la base de datos ‘en crudo’, es decir, del .CSV, de hecho, podría haber incluido menos. Cada nodo ‘Videojuego’ tiene 9 atributos o propiedades:

- **criticScore** – Puntuación dada por MetaCritic
- **global_sales** – Ventas globales.
- **name** – Nombre del videojuego.
- **platform** – Plataforma para la que se publicó (PC, PlayStation, Nintendo DS...)
- **rank** – Número que ocupa el videojuego dentro de los más vendidos.
- **rating** – Puntuación PEGI de edad.
- **user_score** – Puntuación dada por los usuarios.
- **year** – Año en el que se publicó.

Tanto los publishers como los géneros sólo tienen un atributo de nombre, ya que no tenía más datos, aunque tampoco los necesité.

Para transformar la base de datos en crudo en un grafo de conocimiento, construí una sentencia de CYPHER que tomaría los valores que yo especificaría, lo que se iba a traducir como atributos de un nodo. Después solo tendría que hacer las relaciones videojuego-género y videojuego-publisher, y cuando creí que estaba la sentencia completa, me topé con un error días después. La base de datos no tenía datos completos, por lo que algunos videojuegos estaban relacionados con publishers en blanco, nulos. Volví a construir la sentencia, esta vez teniendo en cuenta el caso en que podría tener valores nulos, y mi problema se solucionó.

```
LOAD CSV WITH HEADERS FROM 'file:/// Video_Games_Sales_as_at_22_Dec_2016.csv' AS line
WITH line LIMIT 1000
CREATE (v:Videogame { name: line.Name, year: toInteger(line.Year_of_Release),
platform: line.Platform, rating: line.Rating, criticScore: line.Critic_Score,
user_score: line.User_Score, global_sales: toFloat(line.Global_Sales)})

FOREACH (n IN (CASE WHEN line.Publisher IS NULL THEN [] ELSE [1] END) |
MERGE (d:Publisher {name: line.Publisher})
CREATE (d)-[:PUBLISHED]->(v)
)
FOREACH (n IN (CASE WHEN line.Genre IS NULL THEN [] ELSE [1] END) |
MERGE (g:Genre {name: line.Genre})
CREATE (v)-[:OF_GENRE]->(g)
)
```

Esta imagen contiene el código que usé para transformar los datos en nodos. Primero, accedo al archivo .CSV e indico que voy a llamar ‘line’ a cada línea que contiene, además, le pongo un límite de 1000. Después, creo cada nodo ‘Videogame’

con todos sus atributos, y por último, en los 'FOREACH', compruebo que no sea NULL, o sea, datos nulos.

4.2 Backend

El Backend, como expliqué antes, desarrolla el papel de mediador entre base de datos y Frontend, procesando las peticiones de este último, obteniendo datos de la base de datos y respondiendo al Frontend con lo que necesitaba.

Mi Backend se compone, principalmente, de un archivo llamado "app.js", que es quien realiza el 100% de las funciones del Backend. En él, sólo existe un tipo de petición: la de obtener videojuegos de la base de datos. En realidad, la aplicación no necesita más funciones, ni eliminar juegos ni editarlos, simplemente obtenerlos, de hecho, sería un error permitir que un usuario editase o borrara un videojuego, porque podría cambiar completamente la base de datos y éstos dejarían de ser verídicos. Ahora, vamos a analizar lo que hace el app.js y qué orientación le di a la hora de hacer peticiones.

Primero, tenemos que obtener los datos que nos ha enviado el Frontend, es decir, la

```
app.post("/getGames", function(req, res) {  
  var platform = req.body.platform;  
  var search = req.body.búsqueda;  
  var critic = req.body.critic;  
  var limit = req.body.limit;  
  var genre = req.body.genre;  
  var modern = req.body.modern;
```

Obtención de datos

plataforma, la palabra de la barra de búsqueda, qué tipo de criterio se usó (crítica de usuario, de Metacritic o ventas globales), el límite de videojuegos a mostrar, el género y si se quiere ordenar por año. En la imagen, está cada uno de estos

parámetros en ese orden. En este punto, ya tendríamos todo listo para hacer la petición a base de datos con los datos que acabamos de obtener. Para ello, necesitamos crear una "query". Las queries son las sentencias necesarias para trabajar con bases de datos, en nuestro caso, NEO4J, cuyo lenguaje sería CYPHER. Ahora, hay que construir una, pero teniendo en cuenta que puede haber diferentes caminos. ¿Y si el usuario no ha especificado un género concreto? ¿O si no ha escrito nada en el campo de búsqueda? Hay que tener en cuenta todas las variables. Cuantas más haya, más problemas podría tener, así que se me ocurrió una idea, la idea que me iba a resolver todos los problemas: crear la query por partes. De esta manera, podría hacer las comprobaciones poco a poco, e ir añadiéndole palabras según los diferentes casos.

```

if (genre != "Cualquiera") {
  query = "MATCH(v: Videogame) -[* 1] - (: Genre { name: '" + genre + "' })";
} else {
  query = "MATCH (v: Videogame)";
}
if (platform != "Cualquiera") {
  query = query.concat("WHERE v.platform = '" + platform + "'");
}

if (search != "" && platform != "Cualquiera") {
  query = query.concat(" AND (v.name =~ '(?i).*' + search + '.*') ");
} else if (search != "") {
  query = query.concat(" WHERE (v.name =~ '(?i).*' + search + '.*') ");
}

if (modern) {
  query = query.concat(
    "RETURN v AS videogame ORDER BY v.year DESC, v." +
    critic +
    " DESC LIMIT " +
    limit
  );
} else {
  query = query.concat(
    "RETURN v AS videogame ORDER BY v." +
    critic +
    " DESC, v.year DESC LIMIT " +
    limit
  );
}

```

En la imagen se puede ver cómo desarrollé mi idea: cada 'if' y 'else' representan los diferentes casos que podrían ir ocurriendo (el género o la plataforma no estaban especificados, el campo de búsqueda estaba vacío, se especificó ordenar por año...) y por otra parte, los 'query.concat' añaden las palabras correctas a la query. Todo ello consigue generar una query correcta y funcional, así que se envía a NEO4J.

```

const resultPromise = session.run(query).subscribe({
  onNext: function(record) {
    var element = record.get("videogame").properties;

    if (element.year == undefined) {
      element.year = { low: "--" };
      result.push(element);
    } else {
      result.push(element);
    }
  },
  onCompleted: function() {
    if (result.length < 1) result = [{ message: "Error" }];
    res.send(result);
    session.close();
  },
  onError: function(error) {
    console.log(error);
  }
});

```

Abrimos una sesión de NEO4J y le enviamos la query, como se puede ver en `'session.run(query)'`. A partir de aquí, tenemos tres procedimientos:

`onNext` – Cada vez que la base de datos retorna un elemento, se ejecuta esta función, es decir, si retornara 10 videojuegos, se ejecutaría 10 veces. Yo la aproveché para obtener las propiedades de los nodos 'Videojuego' y añadiéndolas a una lista para después mandarlas al Frontend. También hago ciertas comprobaciones por si el año del videojuego no estuviera definido en base de datos.

`onCompleted` – Se ejecuta cuando ya no hay más videojuegos por retornar, entonces es el momento de enviar la lista al Frontend. Yo contemplé el caso en que esa lista estuviera vacía e hice que enviara al Frontend un mensaje de error, el cual posteriormente se mostraría al usuario. Cuando todo está enviado, se cierra la sesión (`session.close()`)

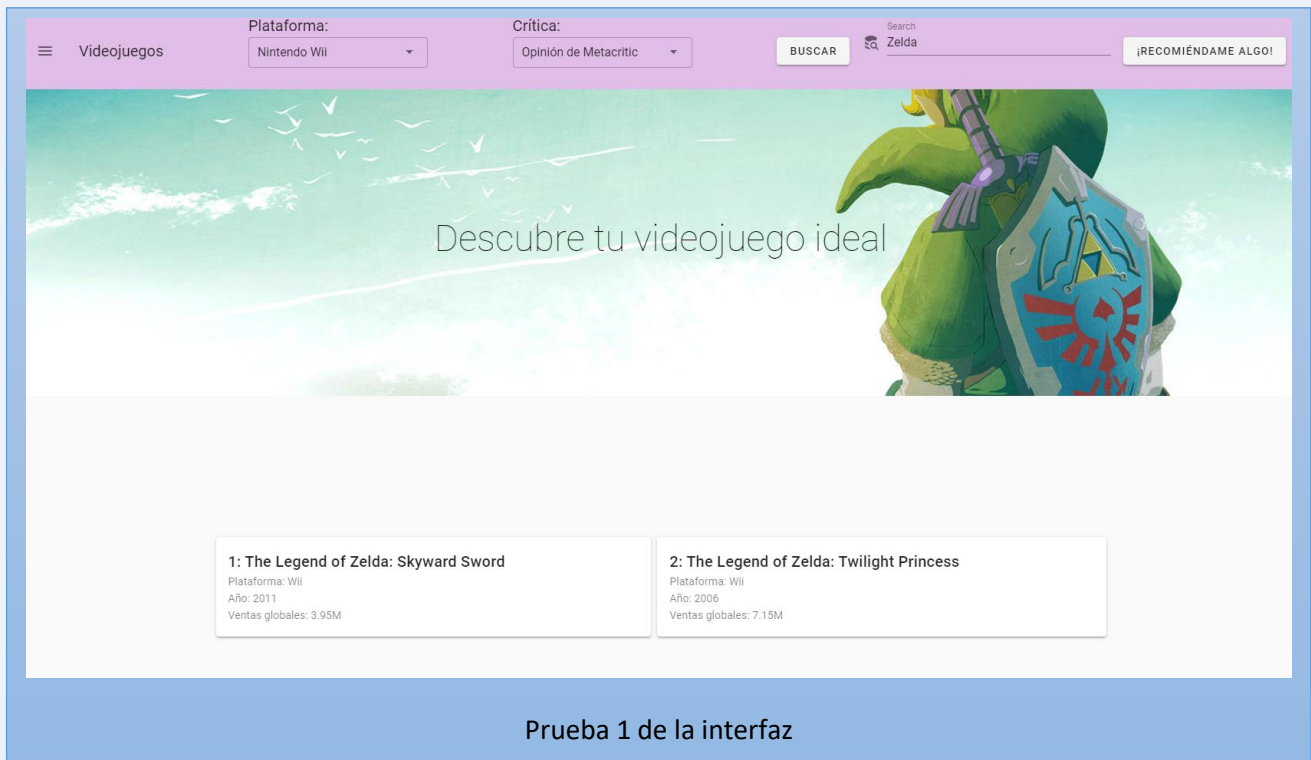
`onError` – Actúa en caso de que se produzca un error, imprimiéndolo en la consola.

Se podría decir que después de enviar los datos, el Backend ha cumplido su función, sólo le queda esperar a otra petición del Frontend para repetir de nuevo este procedimiento.

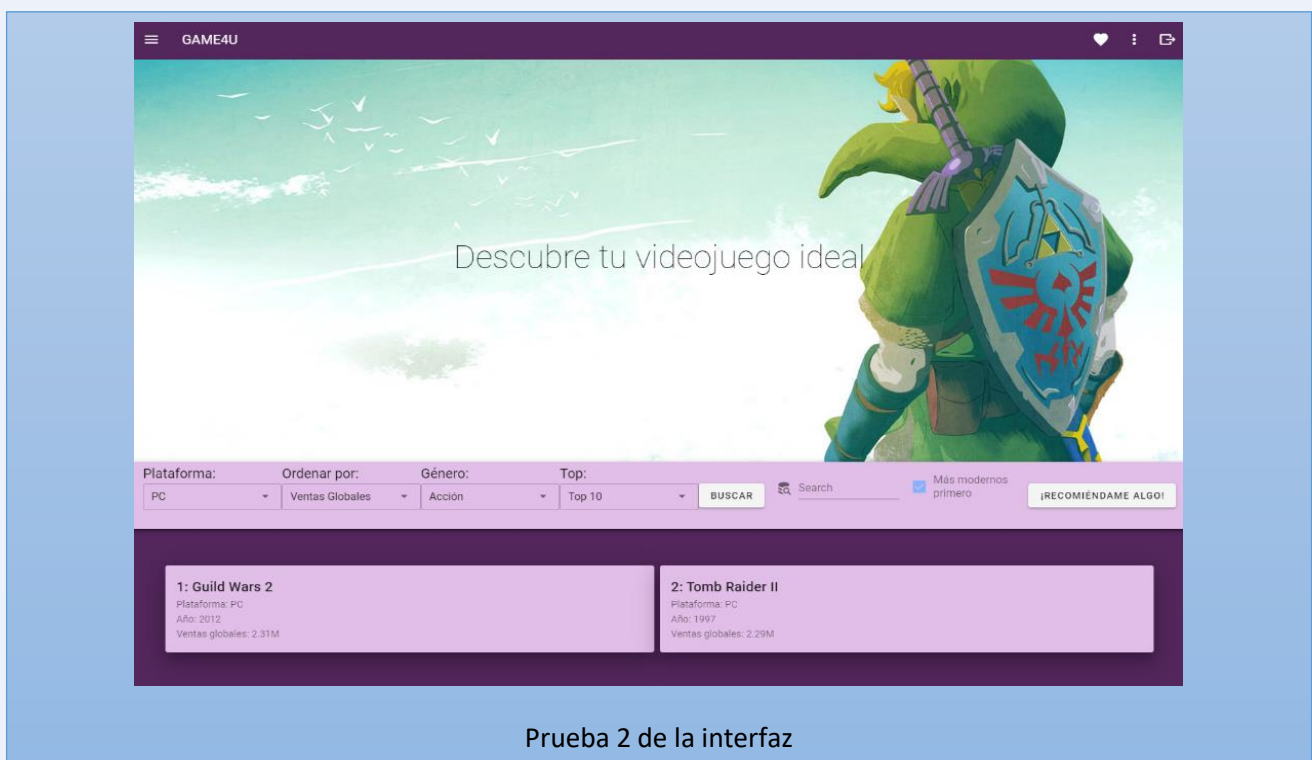
4.3 Frontend

Probablemente el Frontend sea la parte de la aplicación en la que más tiempo empleé programando, y, por lo tanto, la parte en la que más problemas me encontré, aunque también he de decir que todo el tiempo empleado en él dio un resultado extremadamente satisfactorio.

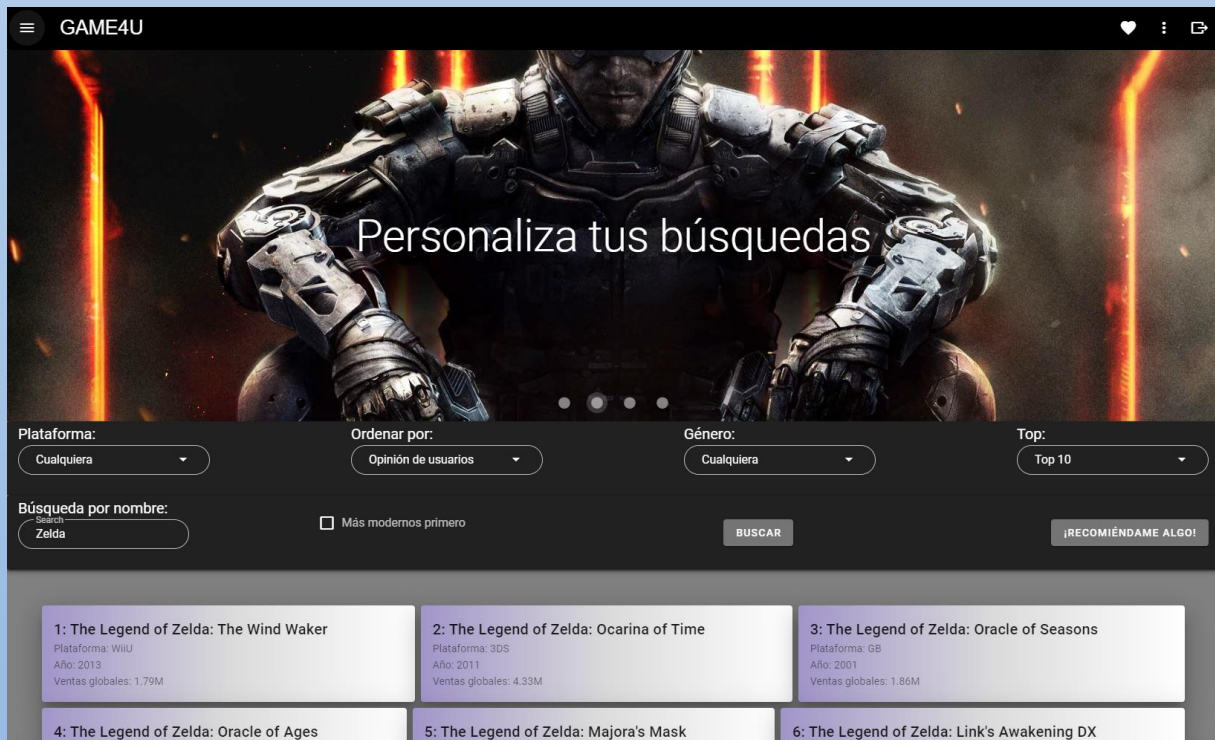
Cuando se crea una interfaz, hay que tener en cuenta que es lo primero que el usuario va a ver. Éste desconoce el código que hay detrás, por lo que se va a guiar por sus ojos, de ahí la importancia de que la interfaz sea atractiva, pero sobre todo accesible. Esto significa que una persona con algún tipo de discapacidad no tenga problemas en usar la aplicación de forma normal. Es por eso que hice mi interfaz muy sencilla, y usando una paleta de colores con el menor contraste posible. Mis primeras pruebas tenían colores atractivos, como morado o violeta, pero no los consideré adecuados y, tras unos días de pruebas y cambios, me decanté por el blanco y negro, colores que todos podemos ver, incluso las personas daltónicas.



Como se puede ver, en esta primera prueba los colores dominantes son el violeta, el blanco y el verde, incluso diría negro. El problema de esta interfaz es que los colores son demasiado claros y la barra de búsqueda está situada en la zona superior por lo que al usuario le exige un esfuerzo extra para acceder a ella y después poder comprobar los resultados en la zona inversa, es decir, en la zona inferior.



Esta segunda prueba ya parece más accesible, sin embargo, no me acabó de convencer por varias razones. La barra está situada en una zona más cercana a los resultados, pero da sensación de que no hubiera suficiente espacio para todos los campos, como si estuvieran demasiado juntos. Respecto a los colores, me gustaban, pero lo veía poco profesional y como si mi aplicación estuviese hecha para niños. En este punto, me planteé la posibilidad de consultar páginas que me ayudasen a decidir los colores, o páginas que estuvieran muy bien diseñadas y pudiera tomar un buen ejemplo.



Interfaz definitiva

Finalmente, y después de observar otras webs, decidí que no saldría del blanco, negro y de la escala de grises. Da un toque moderno, encaja con la temática de videojuegos y es accesible. Si nos fijamos, la barra de búsquedas ha cambiado, ahora ya no están todos los campos dentro de una fila, sino que están divididos en dos filas.

El usuario puede elegir plataforma, orden por crítica, género, top (número de resultados), búsqueda por nombre de videojuego y si quiere ordenarlos de más a menos modernos. Una vez que pulse en buscar, la petición se enviará al Backend, que consultará la base de datos y retornará los resultados, los cuales se muestran en forma de 'tarjetas' en la parte inferior de la aplicación. Dichos campos permiten la elección de:

- **Plataforma** – PC, consolas de Nintendo, consolas PlayStation, consolas XBOX...
- **Ordenar por** – Opinión de usuarios, de Metacritic (web de crítica de videojuegos) o por ventas globales.
- **Género** – Acción, aventura, plataformas, puzzles, deportes...
- **Top** – 3, 5, 10, 15, 20 o 30 resultados.
- **Búsqueda por nombre** – El usuario puede introducir directamente el nombre o parte del nombre de un videojuego para encontrarlo.
- **Más modernos primero** – Si se selecciona, tendrán prioridad los videojuegos más modernos y aparecerán antes.



Respecto a funcionalidad de la interfaz, los componentes más destacables son aquellos en los que se produce una acción al interactuar con ellos: el panel de imágenes, los botones de 'Buscar' y 'Recomiéndame', cada campo de elección... Sin embargo, los únicos que producen llamadas al Backend son los botones. Me gustaría explicar cada zona del código de la interfaz, pero son más de 400 líneas y considero que no vale la pena entretenerse en ello, sino más bien centrarme en explicar el funcionamiento de una forma que cualquier persona pueda entenderlo y no sólo quien ha creado la aplicación.

4.4 Funcionamiento

El punto de interés de la aplicación se encuentra en las búsquedas y la recomendación. Para que un usuario pueda ser recomendado, primero necesitamos

conocer sus gustos, así que se le presentan campos en los que podrán elegir. Cada campo tiene una variable asignada, por ejemplo, si el usuario selecciona como género 'Acción', la variable 'genre' adquiere el valor 'Action', y de esta forma ocurre con toda la barra de búsqueda. El propósito de las variables es, por una parte, ser enviadas al Backend para que realice las consultas, y por otra, ser almacenadas en un historial, cuyo objetivo es poder realizar una recomendación, es la forma de conocer al usuario. Si éste ha elegido muchas veces 'PlayStation 4' como plataforma, es muy probable que posea una o que esté muy interesado en ella, así que conviene tenerlo en cuenta a la hora de recomendarle un videojuego. Por eso, el funcionamiento de la aplicación se basa en analizar el historial y obtener los datos que más se han repetido en las búsquedas de los usuarios. Si no se tienen datos, la aplicación optará por hacer una recomendación al azar, ya que no conoce al usuario aún. De entre todos los juegos que cumplen los parámetros de la recomendación, el algoritmo elige uno de ellos. Ahora, vamos a suponer que somos un usuario, y queremos realizar alguna que otra búsqueda para que se nos recomiende un juego. Las posibilidades son amplias, así que realizaremos varios casos de ejemplo.

Caso 1: Cualquiera

Vamos a hacer una búsqueda predeterminada, es decir, tal y como la página nos la presenta.

The screenshot shows a search interface with the following elements:

- Plataforma:** Dropdown menu set to 'Cualquiera'.
- Ordenar por:** Dropdown menu set to 'Opinión de usuarios'.
- Género:** Dropdown menu set to 'Cualquiera'.
- Top:** Dropdown menu set to 'Top 10'.
- Búsqueda por nombre:** Search bar with the placeholder text 'Search'.
- ☐ Más modernos primero
- BUSCAR** button
- ¡RECOMIÉNDAME ALGO!** button

No especificamos plataforma ni género, lo ordenamos por opinión de usuario, pedimos 10 resultados, sin indicar que se nos muestren los más modernos primero.

1: Pokemon Sun/Moon Plataforma: 3DS Año: 2016 Ventas globales: 7.14M	2: Yokai Watch Busters Plataforma: 3DS Año: 2015 Ventas globales: 2.29M	3: Star Wars Battlefront (2015) Plataforma: PS4 Año: 2015 Ventas globales: 7.98M	4: Star Wars Battlefront (2015) Plataforma: XOne Año: 2015 Ventas globales: 3.66M	5: Need for Speed (2015) Plataforma: PS4 Año: 2015 Ventas globales: 2.43M
6: Call of Duty: Black Ops 3 Plataforma: PS4 Año: 2015 Ventas globales: 14.63M	7: Call of Duty: Black Ops 3 Plataforma: XOne Año: 2015 Ventas globales: 7.39M	8: Monster Hunter X Plataforma: 3DS Año: 2015 Ventas globales: 3.32M	9: Super Smash Bros. for Wii U and 3DS Plataforma: 3DS Año: 2014 Ventas globales: 7.55M	
10: Super Smash Bros. for Wii U and 3DS Plataforma: WiiU Año: 2014 Ventas globales: 4.87M				

El resultado es un conjunto de los 10 videojuegos mejor valorados por los usuarios de 2016 hacia atrás, sin tener en cuenta su año de salida. Hay un par repetidos, pero en realidad se diferencian en su valores: uno de los Call of Duty es para PlayStation4, el otro para XBOX. Ocurre lo mismo con los juegos de Star Wars y Super Smash Bros. Son del mismo año, pero sus ventas, crítica y plataforma son diferentes.

Caso 2: Búsqueda concreta

Ahora, vamos a cambiar ciertos valores. Supongamos que queremos encontrar un juego para la consola Nintendo Wii, con temática de Deportes, ordenándolo por ventas globales y por modernidad.

The screenshot shows a search interface with the following filters and results:

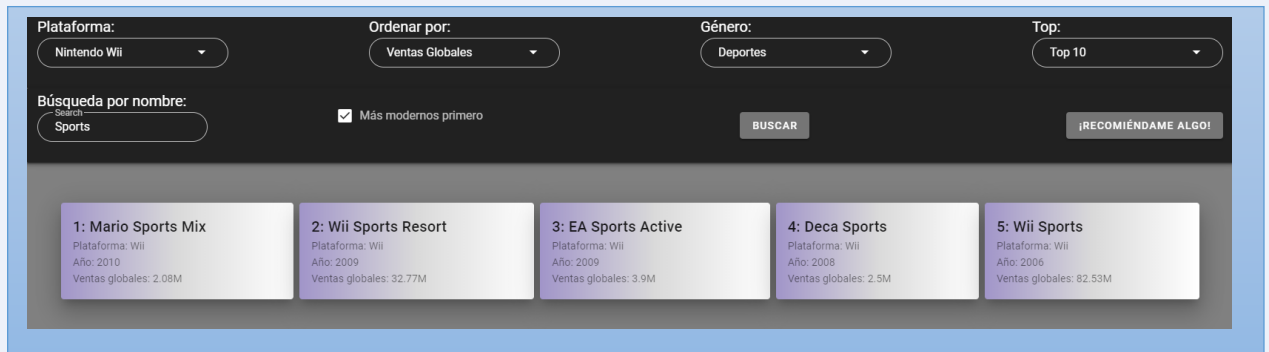
- Plataforma:** Nintendo Wii
- Ordenar por:** Ventas Globales
- Género:** Deportes
- Top:** Top 10
- Búsqueda por nombre:** Search
- ☒ Más modernos primero
- BUSCAR**
- ¡RECOMIENDAME ALGO!**

Rango	Título	Plataforma	Año	Ventas globales
1:	Mario & Sonic at the London 2012 Olympic Games	Wii	2011	3.72M
2:	Zumba Fitness 2	Wii	2011	2.81M
3:	Zumba Fitness	Wii	2010	6.71M
4:	Mario Sports Mix	Wii	2010	2.08M
5:	Wii Sports Resort	Wii	2009	32.77M
6:	Wii Fit Plus	Wii	2009	21.79M
7:	Mario & Sonic at the Olympic Winter Games	Wii	2009	4.53M
8:	EA Sports Active	Wii	2009	3.9M
9:	Your Shape featuring Jenny McCarthy	Wii	2009	2.12M
10:	Deca Sports	Wii	2008	2.5M

Al elegir dos tipos de orden, por ventas y modernidad, tiene prioridad la modernidad. Es decir, dentro de cada año, los videojuegos se ordenarán por ventas. En la imagen, tenemos dos juegos de 2011, dos de 2010, cinco de 2009 y uno de 2008, en ese orden. Dentro de los juegos de 2009, primero se muestra el que más ingresos obtuvo ese año, mientras que el quinto, el que menos. Lo mismo ocurre con el resto de años, y decidí que así sería porque, ya que los juegos que de la base de datos no son especialmente modernos, qué menos que dar la opción de filtrarlos para descartar los más antiguos.

Caso 3: Búsqueda por nombre

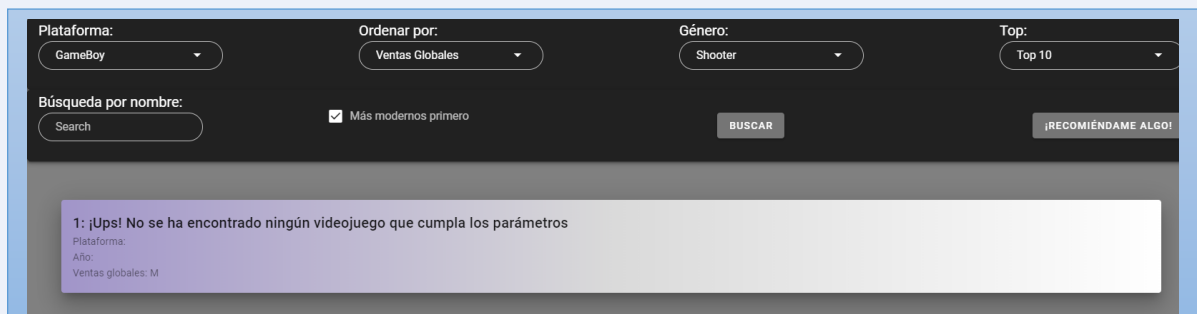
A la búsqueda anterior, vamos a añadirle un título de videojuego en la búsqueda por nombre, por ejemplo, “Sports”. Dado que estamos dentro del género “Deportes”, es muy probable que existan videojuegos que contengan esta palabra en su nombre.



El criterio de orden se conserva, simplemente se han descartado los juegos que no contenían “Sports” en su nombre, pero se sigue ordenando primero por año y después por ventas globales.

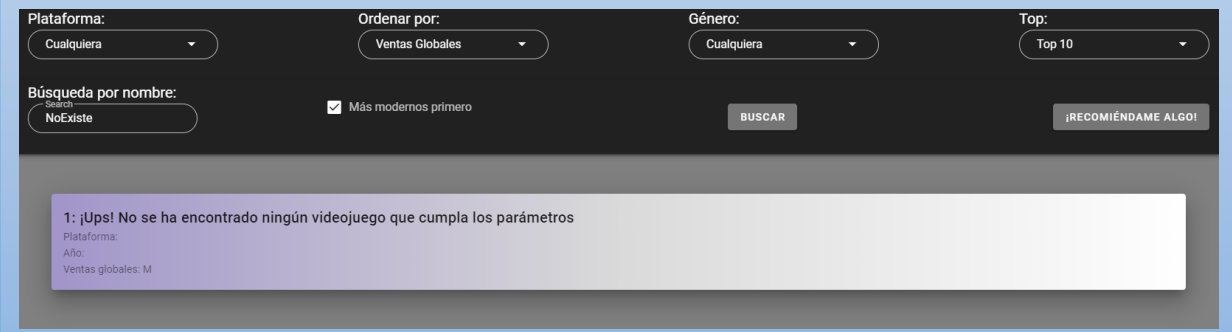
Caso 4: Error 404, ¡Not Found!

Ahora vamos a intentar encontrar una combinación de género y plataforma que no exista, así comprobaremos que ocurre cuando la aplicación no encuentra lo que deseamos en la base de datos. Lo ideal sería seleccionar una consola más bien antigua y un género moderno, por ejemplo, GameBoy y Shooter. Normalmente los juegos de GameBoy eran de plataformas o rol, y los juegos tipo ‘Shooter’ (de disparos) están en auge actualmente.



“¡Ups! No se ha encontrado ningún videojuego que cumpla los parámetros” es el texto que se muestra cuando la búsqueda no ha tenido éxito.

La forma en que me percaté de que esto podía ocurrir, fue, por desgracia, mediante mi aplicación congelándose. En ese momento supe que no había tenido en cuenta esta posibilidad, pero no me costó mucho ideármelas para mostrar esta tarjeta. Simplemente, comprobé que el Backend no me retornara ningún videojuego, sólo en ese caso mostraría la advertencia. Lo mismo ocurre si escribimos un nombre inexistente para un videojuego.

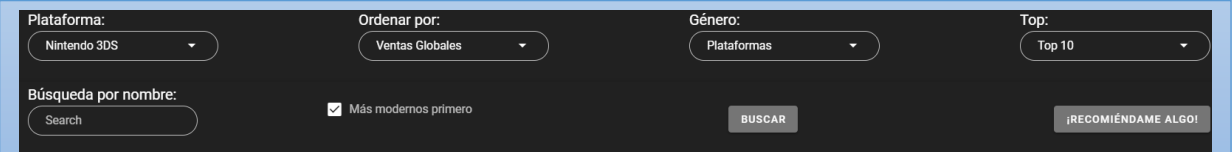


The screenshot shows the search interface with the following settings: Plataforma: Cualquiera, Ordenar por: Ventas Globales, Género: Cualquiera, Top: Top 10. The search bar contains 'NoExiste'. A message box displays: '1: ¡Ups! No se ha encontrado ningún videojuego que cumpla los parámetros'. Below the message, it lists: Plataforma: Año: Ventas globales: M.

Caso 5: ¡Recomiéndame algo!

Creo que ya es hora de poner a prueba las recomendaciones. Como ya expliqué, funcionan por 'moda', es decir, la elección que más se ha repetido de cada parámetro es la que se usará para recomendar. Primero, vamos a realizar varias búsquedas.

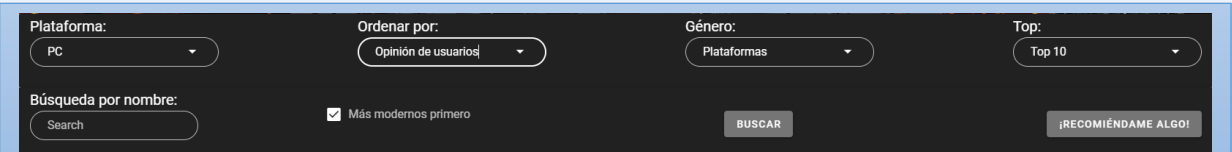
Búsqueda 1:



The screenshot shows the search interface with the following settings: Plataforma: Nintendo 3DS, Ordenar por: Ventas Globales, Género: Plataformas, Top: Top 10. The search bar contains 'Search'. A button labeled '¡RECOMIÉNDAME ALGO!' is visible.

Nintendo 3DS, Ventas Globales, Plataformas, Top 10, Más modernos primero

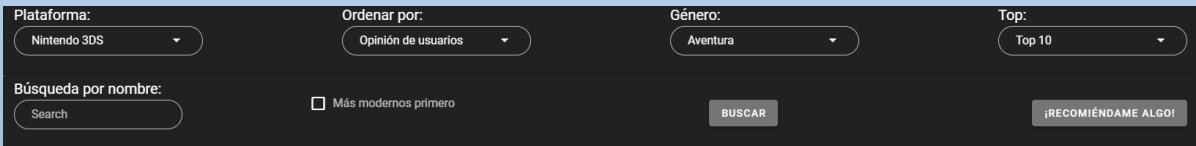
Búsqueda 2:



The screenshot shows the search interface with the following settings: Plataforma: PC, Ordenar por: Opinión de usuarios, Género: Plataformas, Top: Top 10. The search bar contains 'Search'. A button labeled '¡RECOMIÉNDAME ALGO!' is visible.

PC, Opinión de usuarios, Plataformas, Top 10, Más modernos primero

Búsqueda 3:

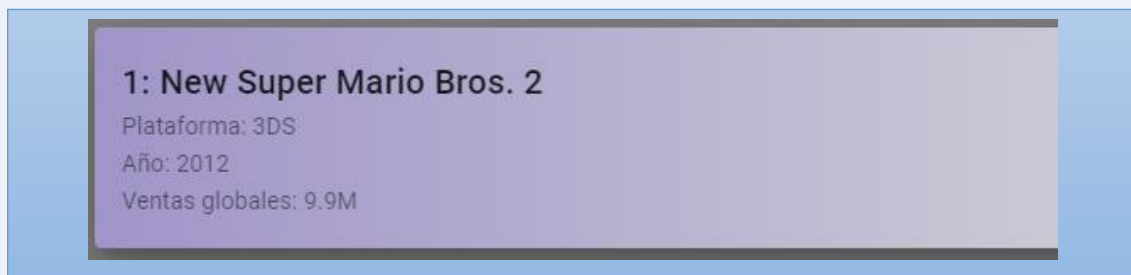


Plataforma: Nintendo 3DS Ordenar por: Opinión de usuarios Género: Aventura Top: Top 10

Búsqueda por nombre: Search ☐ Más modernos primero BUSCAR ¡RECOMIÉNDAME ALGO!

Nintendo 3DS, Opinión de usuarios, Aventura, Top 10, Más antiguos primero

Con estas búsquedas será suficiente para tener un pequeño historial donde poder aplicar el algoritmo. La plataforma más buscada ha sido “Nintendo 3DS”, el orden, Opinión de usuarios, el género, plataformas, y en dos ocasiones hemos seleccionado “Más modernos primero”. Y la recomendación sugerida por la aplicación es:

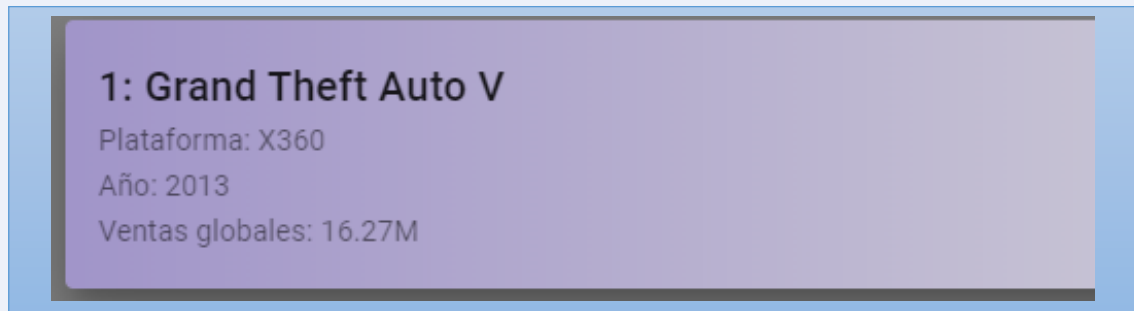


“New Super Mario Bros. 2”, un famoso juego para Nintendo 3DS del año 2012, pero no hay que dejarse engañar por el año: a día de hoy sigue teniendo jugadores. La recomendación se ajusta perfectamente a los gustos del usuario: coinciden la consola, el género y es un juego muy bien valorado. Además, pertenece a la saga Super Mario, ¿a quién no le podría gustar?

Caso 6: Recomendación aleatoria

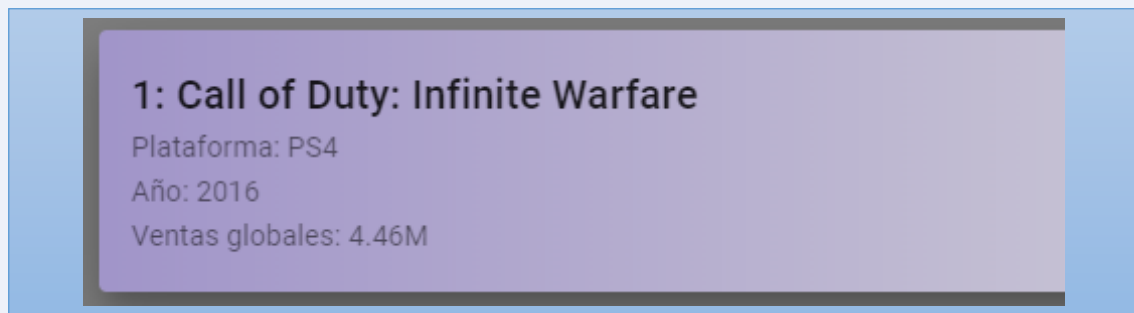
Es muy complicado conocer al usuario si no disponemos ningún dato o ninguna pista de lo que le gusta. Por eso, al principio y con un historial vacío, si el usuario clicara en “¡Recomiéndame algo!” sin haber hecho siquiera una búsqueda, se le sugeriría un juego aleatorio. Hagamos varias pruebas.

Prueba 1:



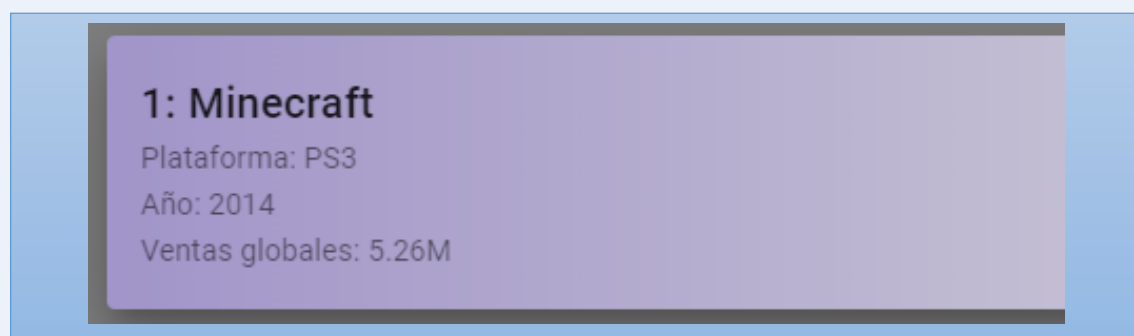
“Grand Theft Auto V”, uno de los juegos que más impacto tuvo en su día. He de decir que me ha sorprendido, ha sido una muy buena recomendación. Continuemos.

Prueba 2:



Uno de los famosos “Call of Duty”. Personalmente no soy muy fan de esta saga, pero soy consciente de la gran cantidad de fans que tiene. Además, es de los más modernos (2016) y para una consola muy habitual, PS4.

Prueba 3:



Minecraft: ese juego que nunca pasa de moda. Hace ya 10 años que este juego salió a la luz y, sin embargo, a día de hoy tiene millones de jugadores (y subiendo) entre los cuales me incluyo. Muy buena recomendación.

Después de unas cuantas pruebas, puedo decir que, generalmente, los videojuegos recomendados de forma aleatoria suelen ser modernos, aunque no es casualidad ya que decidí que, de recomendar un juego de forma aleatoria, mejor que fuese moderno, hoy en día no hay muchos jugadores de GameBoy o PlayStation 1. También excluí el campo “Búsqueda por nombre”, los resultados se reducirían demasiado junto con las posibilidades de que el usuario descubra un nuevo juego, y no uno que ya conozca.

5. Análisis crítico

Si soy sincera, no me esperaba que este trabajo fuera a despertar en mí tanto interés y ganas de programar. Al inicio de curso ni siquiera sabía cómo manejar la situación, tenía el método para hacerlo (NEO4J) y sabía que quería relacionarlo con los videojuegos, pero necesitaba combinar ambos en una misma aplicación. Fue una buena decisión investigar primero qué base de datos utilizar en vez de crear una, de esta manera podría elegir entre más videojuegos que recomendar sin tener que añadir uno a uno. Sin embargo, me encontré con que había ciertos campos nulos, es decir, que durante la recogida de datos, éstos no se especificaron y quedaron en blanco. Eso me llevó a tener que decidir entre 3 o 4 bases de datos, observando cuál de ellas estaba más completa y tenía lo que yo necesitaba.

A la hora de aprender Cypher, he de decir que estoy muy orgullosa de la labor que hice con la guía y ojalá pueda servir a otras personas en el futuro. No fue difícil aprender un nuevo lenguaje, a estas alturas los alumnos de informática tenemos todo un abanico de lenguajes de programación aprendidos y establecemos relaciones entre ellos más fácilmente, en este caso, tenía de precedente a SQL. Mi fallo fue no realizar los cursos de NEO4J, a lo mejor hubiera aprendido un poco más de lo que sé. Quise que fuera de esta manera porque me considero una persona autodidacta y aprendí a base de investigación, prueba y fallo. A diferencia de otras carreras como la Medicina, nosotros podemos hacer pruebas sin arriesgar los sistemas, en entornos de test, y confundirnos tantas veces como necesitemos.

Durante el desarrollo de la aplicación como tal, me concentré mucho en ella, ya que podía permitírmelo en ese momento: no teníamos exámenes cerca, llevaba bien las prácticas de otras asignaturas... Y al final, resulté beneficiada, tanto yo como los compañeros a los que ayudaba, ya que no tenía mucho que hacer, así que empleaba mi tiempo ayudándoles con sus problemas técnicos. No me arrepiento de haberlo hecho, pues gracias a ellos tuve mi idea del algoritmo de recomendación. La conclusión a la que llegué es que, cuando prestas tu ayuda, inconscientemente, eres ayudado/a.

5.1 DAFO

En este punto creo que debería realizar un análisis DAFO: Debilidades, Amenazas, Fortalezas y Oportunidades.

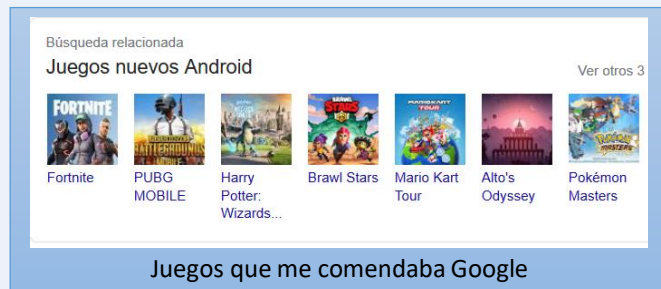
- **Debilidades** – Podría haber mejorado el algoritmo, es cierto que no es complicado ni elaborado, pero me gustó tanto la idea que no me lo pensé dos

veces. Como dije, mi algoritmo se basa en un historial de búsquedas, muy similar a Google, por lo que me fascinó que mi aplicación pudiera implementarlo.

- **Amenazas** – Apenas existía documentación respecto al tipo de aplicación que quería hacer, así que, sobre todo al principio, tuve que recoger muchísima información sobre las metodologías, distintos drivers por si acaso no era posible y soluciones a problemas que la comunidad proponía.
- **Fortaleza** – Mi gran fortaleza es la base de datos y la interfaz. Tengo una base de datos lo suficientemente amplia como para poder recomendar el videojuego ideal a cada persona, o que encuentre mediante búsquedas el juego que tenía en mente. Realicé la interfaz de la forma más intuitiva y accesible posible, es decir, que personas con ciertas discapacidades puedan usar la aplicación sin problemas. Creo que ese es un gran punto que no muchos desarrolladores tienen en cuenta.
- **Oportunidades** – Mi gran oportunidad fue la ayuda que di y, sin quererlo, recibí. Me proporcionó la idea de la recomendación y alguna que otra respecto a la interfaz, además, también pude corregir algunos fallos en mi aplicación al verlos en las de mis compañeros.

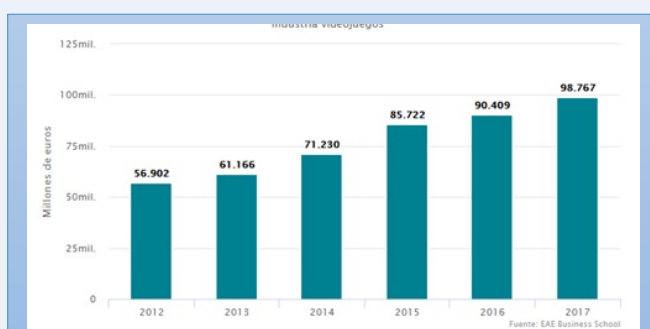
6. Líneas de futuro

Como dije al principio, el sector del videojuego está creciendo constantemente. Es cierto que existen muchas páginas de “recomendación de videojuegos”, tales como 3DJuegos, pero su recomendación es por valoración de jugadores, no recomendación personalizada usando una base de datos. Durante mis búsquedas, Google me mostraba juegos de móvil, interpretando que lo que yo estaba buscando eran aplicaciones (juegos) y quería que me recomendara algunos. Por lo tanto, mi búsqueda de una aplicación similar a la mía



fracasó, siempre me encontraba con titulares del estilo “Los mejores juegos de...”, ¿quiere decir esto que los usuarios no usan este tipo de aplicaciones? ¿o a lo mejor que no existe ninguna lo suficientemente valorada como para que un usuario pueda encontrarla? Sería una buena idea proponer la aplicación a páginas que se dedican a recomendar videojuegos, además, podrían ir ampliando la base de datos, porque siempre van a estar surgiendo nuevos juegos. También habría que mejorarla para que el historial se adaptase a la sesión del usuario y que se guardase al reiniciar la sesión y, dependiendo de la página, que usara sus propias valoraciones o las de usuario como criterio de búsqueda.

Por otra parte, me informé sobre la analítica de datos aplicada a los videojuegos. Me encontré con declaraciones como que “los videojuegos han pasado de ser simples



Gráfica de los ingresos del sector del videojuego

métodos de entretenimiento a convertirse en deportes o incluso herramientas de marketing y publicidad. Es una de las industrias con más y mejor crecimiento en los últimos años, llegando a superar en ingresos a sectores más antiguos y asentados como son el cine, los libros o la música.” La

información que se puede obtener de los videojuegos podría ser respecto a los jugadores, como su comportamiento, o a los datos del sistema, evaluando su rendimiento. Google Analytics ya tiene su propio sistema para juegos creados con el motor Unity. El creador puede recoger los datos de comportamiento de sus

jugadores con ciertas líneas de código, por ejemplo, cuándo o dónde muere el personaje.

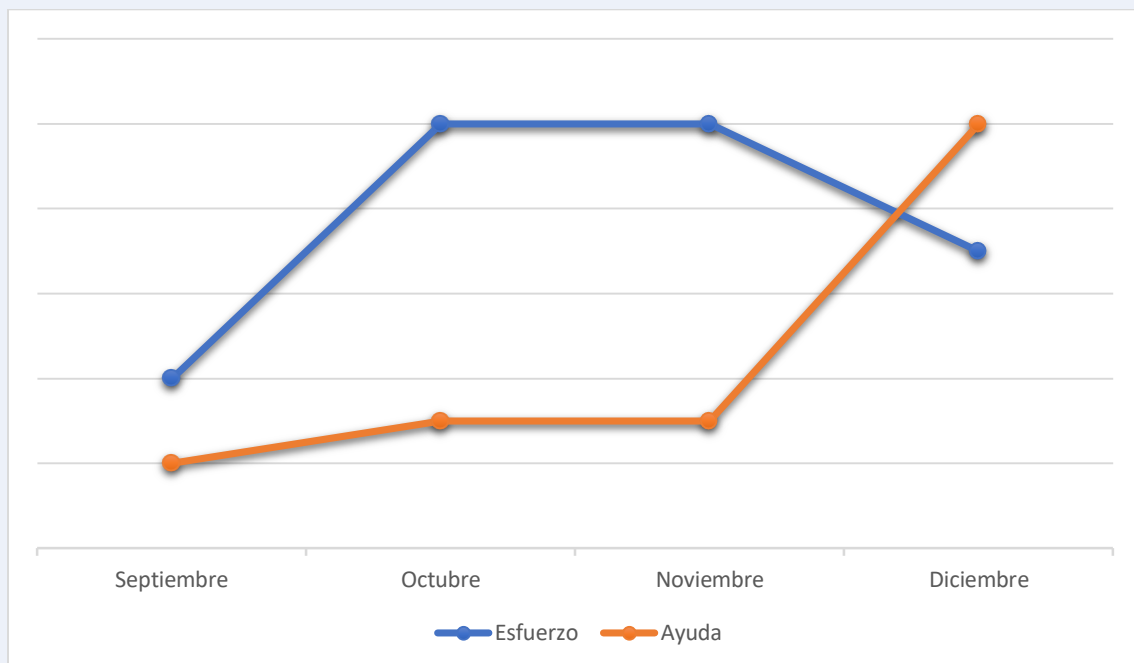
Tomando otro punto de vista y probablemente el más interesante y con más futuro, la analítica digital ha ayudado a mejorar y a conseguir el éxito de algunos de los videojuegos más conocidos del momento. Algunos ejemplos de conceptos que se estudian son la economía interna del juego, el nivel de dificultad, las preferencias de los usuarios hacia los diferentes modos de juego o personajes, la motivación y la participación. Además, el análisis de datos también es útil para detectar jugadores tramposos. Existen algoritmos que miden el rendimiento de los jugadores y les ayudan a mejorar. Sin duda, toda esta tecnología tiene su punto de mira en los conocidos 'eSports', los videojuegos como deporte. Ya se han realizado numerosas competiciones a nivel mundial y se podría decir que aquí hay futuro. Así como se usa Big Data en fútbol, béisbol o Fórmula 1, se usa también en los eSports, de hecho, es más fácil ya que su naturaleza digital facilita la recogida de datos. Se dice que hay mucho campo de desarrollo todavía, así que es una buena idea apostar por el BigData en los eSports.



Competición mundial de League Of Legends 2019

7. Qué he aprendido

La lección más importante que me he dado a mí misma es que por mucho que me asuste al principio un proyecto, siempre le acabo poniendo la suficiente voluntad e ilusión como para acabarlo incluso antes de tiempo. He hecho un gráfico que representa mi esfuerzo durante el semestre. Al principio todo fue investigar y recoger información para los meses que venían. octubre y noviembre los dediqué plenamente a hacer pruebas programando y después a implementarlo según mis ideas. Mi rendimiento bajó hacia diciembre porque acabé el proyecto antes de tiempo, simplemente me quedaba darle algún que otro retoque, así que pude aprovechar para echar una mano a mis compañeros.



Dándole un punto de vista técnico, he aprendido a trabajar con un tipo de base de datos que desconocía y que tiene lenguaje propio. Hasta ahora, no había salido de SQL, pero CYPHER no es muy diferente a él, por eso no me costó aprenderlo. Mi parte favorita era cuando funcionaba lo que quería hacer, pero para llegar hasta ahí tuve que cometer muchos errores, reescribir algunas sentencias de la guía que me hice y probar sentencias diferentes. El hecho de cometer errores me permitió ayudar a mis compañeros, que, al parecer, cometían los mismos errores que yo.

Durante la parte más intensa del semestre, es decir, donde desarrollé la aplicación como tal, me encontré con problemas que me llevaban a tomar unas decisiones u otras: a veces, estaba confiada de que podría resolverse si me fijaba bien en las líneas

de código y encontraba el error, pero otras lo resolvía eliminando lo que ya tenía hecho y haciendo algo nuevo. Con ambas soluciones el resultado era bueno, y me hacía pensar que, a veces, hay que empezar de 0, para salir de un bucle de desesperación por un error que podría o no tener solución, o simplemente apartarse de la pantalla un rato. No sería la primera vez que encuentro al minuto un error que había estado buscando durante horas gracias a dejar el ordenador por tener que ir a clase.

8. Referencias

1. **Graph Algorithms:** <https://neo4j.com/graph-algorithms-book/>
2. **Canal de YouTube de NEO4J:**
<https://www.youtube.com/channel/UCvze3hU6OZBkB1vkhH2IH9Q>
3. **Kaggle:** <https://www.kaggle.com/datasets>
4. **Drivers NEO4J:** <https://neo4j.com/developer/language-guides/>
5. **Driver JavaScript:** <https://neo4j.com/docs/api/javascript-driver/current/>
6. **Node.js:** <https://nodejs.org/es/>
7. **Express:** <https://expressjs.com/es/>
8. **Vue.js:** <https://vuejs.org/>
9. **Vuetify:** <https://vuetifyjs.com/en/>
10. **Vue-Resource:** <https://github.com/pagekit/vue-resource>
11. **Electron:** <https://electronjs.org/>

9. Bibliografía

GitHub del proyecto: <https://github.com/aferns16/Game4U>

Líneas de futuro. Artículos usados:

- ❖ “Análítica digital aplicada a videojuegos”, Guillermo Cano,
<https://www.analiticaweb.es/analitica-digital-aplicada-videojuegos/>
 - ❖ “El Big Data llega a los eSports”, El País Economía,
https://retina.elpais.com/retina/2018/01/25/innovacion/1516897950_287371.htm
- !