SISTEMA RECOMENDADOR DE EVENTOS CULTURALES

MEMORIA DEL PROYECTO

CONTENIDO

1.		INTRODUCCIÓN DEL PROYECTO	2
2.		BASE DE DATOS	2
3.		FRONTEND	5
,	١.	PÁGINA DE INICIO	5
E	3.	PÁGINA DE RESULTADOS	5
(2.	ALERTA DE ERROR	6
4.		BACKEND	6
5.		EJECUCIÓN DE LA APLICACIÓN	6
6.		ANÁLISIS DE RESULTADOS	7
Á	١.	SOLO PROVINCIA	7
7.		ANALISIS DAFO	9
[DΕ	BILIDADES	9
A	۱۸	MENAZAS	9
F	Ю	PRTALEZAS	9
(DΡ	PORTUNIDADES	9
8.		PROXIMOS PASOS	9
9.		REFERENCIAS Y HERRAMIENTAS UTILIZADAS	10

1. INTRODUCCIÓN DEL PROYECTO

En el mundo de los eventos culturales, la planificación y promoción de eventos es la parte más importante para que un espectáculo cuelgue el cartel de aforo completo. En este contexto, el presente trabajo se centra en la concepción e implementación de un sistema recomendado de eventos, diseñado con el propósito de recomendar a la gente los eventos que más les interese.

El tema de este trabajo se ha elegido debido a mi empleo en una empresa de gestión de salas de teatro y producción de eventos y espectáculos en Castilla y León, Baleares y Madrid. Desde mi propia experiencia, es muy difícil que los programadores de eventos culturales puedan atraer al público sin conocer todos los detalles del espectáculo. En la actualidad, no existe ninguna plataforma que junte todos los espectáculos de España y permita a los usuarios filtrar por intereses. Las únicas plataformas que permiten filtrar son las de ventas de entradas, pero en ellas no se encuentran los eventos gratuitos, o con ventas de entradas físicas

Este proyecto es la primera piedra para conseguir crear una aplicación para las salas que gestiona la empresa en la que trabajo.

2. BASE DE DATOS

Los datos para nuestra base de datos han sido obtenidos del portal de Datos Abiertos de la Junta de Castilla y León. Antes de poder meterlos en Neo4J he tenido que limpiar los datos ya que contenían errores de formato y filas incompletas. En el repositorio de GitHub puedes ver el CSV antes y después de ser procesado. Además, el CSV original estaba separado por ";" mientras que Neo4J necesita que este separado por comas. Para introducir los datos del CSV en la base de datos en Neo4J, utilizamos la siguiente Query

```
LOAD CSV WITH HEADERS FROM 'file:///limpiado.csv' AS row

// Crear nodos de Evento

CREATE (:Evento {
    titulo: row.titulo,
    tematica: row.tematica,
    fecha: row.fecha,
    publico: row.publico,
    foto: row.foto,
    link: row.link

}}

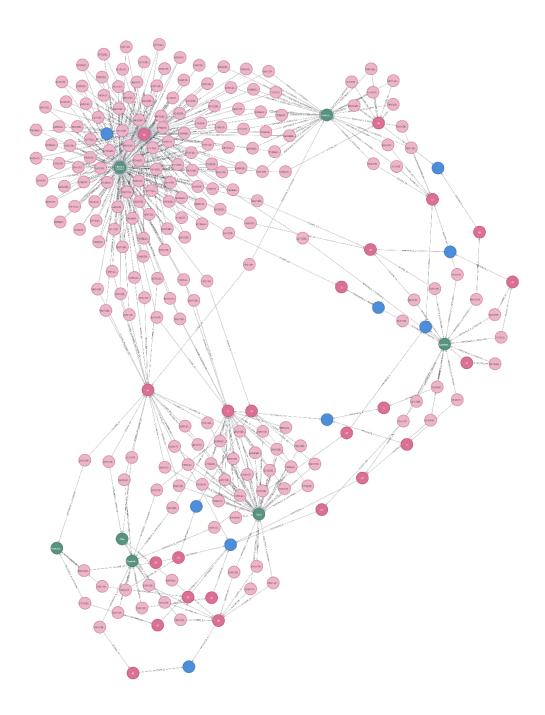
// Crear nodos de Lugar

MERGE (:Lugar {
    nombre: row.lugar,
    localidad: row.localidad,
```

```
idProvincia: row.idProvincia,
  provincia: row.provincia,
  calle: COALESCE(row.calle,"")
})
//Crear nodos provincias
MERGE (:Provincia {
  nombre: row.provincia,
  idProvincia: row.idProvincia
})
// Enlazar nodos de Evento con nodos de Categoría
WITH row
MATCH (evento:Evento {titulo: row.titulo})
WHERE row.categoria IS NOT NULL
MERGE (categoria:Categoria {nombre: row.categoria})
MERGE (evento)-[:PERTENECE_A_CATEGORIA]->(categoria)
//Enlazar nodos de Evento con nodos de Lugar
WITH row, evento
MATCH (I:Lugar {nombre: row.lugar})
MERGE (evento)-[:OCURRE_EN]->(I)
// Enlazar nodos de Lugar con nodos de Provincia
WITH row
MATCH (lugar:Lugar {nombre: row.lugar})
MATCH (provincia: Provincia {idProvincia: COALESCE(row.idProvincia, ")}) // Utilizar COALESCE para manejar valores nulos
MERGE (lugar)-[:PERTENECE_A]->(provincia)
```

Con este comando, creamos un nodo por cada evento, un nodo por cada lugar, un nodo por cada categoría, un nodo por cada provincia y relacionamos todos para tener una base de datos relacional y consistente.

La estructura resultante de la base de datos es la siguiente:



Este gráfico se encuentra también subido a GitHub con mayor calidad para que se pueda observar con detalle que es cada nodo.

Para poder ver la base de datos se utiliza la siguiente Query:

MATCH (n) DETACH DELETE n

Para eliminar todo el contenido de la base de datos se utiliza la siguiente Query:

MATCH (n) RETURN n

3. FRONTEND

El frontend está creado con VUE y utiliza axios para hacer las solicitudes a la base de datos. Esta formado por dos páginas y un diálogo de error.

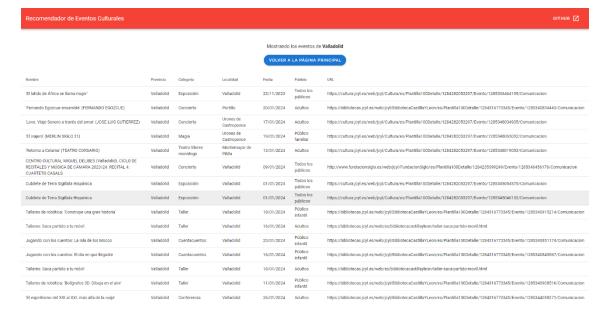
A. PÁGINA DE INICIO

En esta página se le presenta al usuario un formulario para que marque sus intereses. Una vez seleccionados, el usuario tiene que dar al botón de buscar eventos para que empiece a funcionar la búsqueda y recomendación de los eventos.



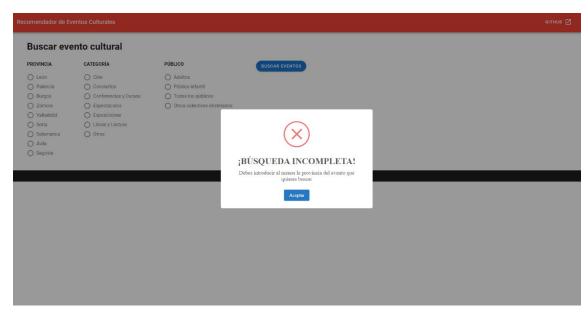
B. PÁGINA DE RESULTADOS

Una vez pulsado el botón de buscar, el usuario es redireccionado a la página de resultados. En esta página, se muestran los eventos que se le han recomendado para los datos introducidos por el usuario.



C. ALERTA DE ERROR

En caso de que el usuario no maque la provincia en la que desea encontrar los eventos, se le mostrar un mensaje de error indicando que es necesario seleccionar una provincia



4. BACKEND

Para el backend, utilizaremos NodeJS con drivers de Express y neo4j. El backend es la parte lógica de nuestra aplicación, que se encargará de enviar los datos al frontend cuando los solicite a través de la API.

Método	Descripción
eventosProvincia(provincia)	Devuelve todos los eventos que tienen lugar
	en la provincia que se le pasa por parámetro
eventosLocalidad(localidad)	Devuelve todos los eventos que tienen lugar
	en la localidad que se le pasa por parámetro
eventosFecha(fecha)	Devuelve todos los eventos que tienen lugar
	en la fecha que se le pasa por parámetro
eventosCategoria(categoría)	Devuelve todos los eventos de la categoría
	que se le pasa por parámetro
eventosInfantil()	Devuelve los eventos que son aptos para
	niños y público familiar
eventosAdultos()	Devuelve los eventos que son para adultos

5. EJECUCIÓN DE LA APLICACIÓN

Para ejecutar la aplicación por primera vez, tendremos que abrir Neo4J e importar la base de datos con los comandos que se indican en el apartado "Base de Datos".

A continuación, iremos a la carpeta "backend" y abriremos un terminal. El el pondremos el siguiente comando para ejecutar la aplicación de NodeJS

node app.js

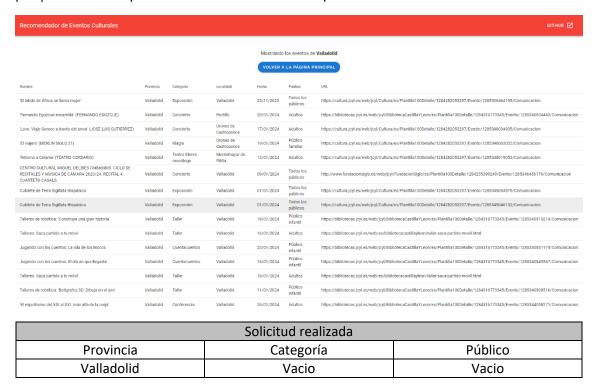
Una vez ejecutado el backend, solo queda ejecutar el frontend. Para ellos nos dirigimos a la carpeta fronted y ejecutaremos el comando npm run serve.

6. ANÁLISIS DE RESULTADOS

Para probar la aplicación, se han realizado cuatro posibles casos:

A. SOLO PROVINCIA

En esta situación, la aplicación devuelve los eventos relacionados con la provincia que se introduce, bien sea porque la localidad en la que se desarrolla pertenece a esa provincia, o porque en la descripción del evento se nombra a la provincia.



B. PROVINCIA Y CATEGORÍA

En la base de datos, los eventos se categorizan temáticas. La aplicación agrupa todas ellas en siete grandes categorías para que sea más fácil al usuario poder seleccionar cual le interesa más con una mayor sencillez.



Solicitud realizada			
Provincia	Categoría	Público	
Valladolid	Cine	Vacio	

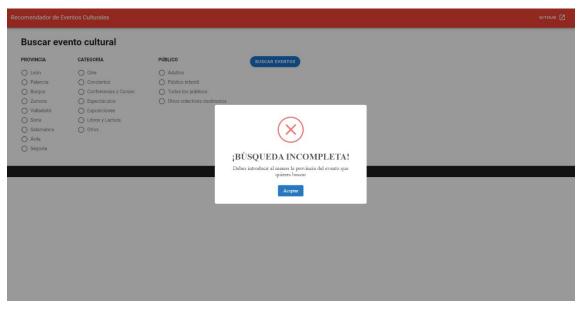
C. PROVINCIA, CATEGORÍA Y PÚBLICO



Solicitud realizada		
Provincia	Categoría	Público
Valladolid	Libros	Adultos

D. NINGUN DATO

En caso de que se pida a la aplicación una consulta sin indicar la provincia, devolverá un error



Solicitud realizada		
Provincia	Categoría	Público
Vacio	Vacio	Vacio

7. ANALISIS DAFO

Debilidades	Amenazas	
 La base de datos es muy pequeña La base de datos es solo de Castilla y León La mayoría de los eventos son de bibliotecas 	- La Junta de Castilla y León puede dejar de actualizar datos	
Fortalezas	Oportunidades	
 Interfaz sencilla Comprueba eventos duplicados 	 Agregar más fuentes de eventos para tener más datos Ampliar a otras comunidades Integración con la API 	

DEBILIDADES

La base de datos solo utiliza los datos abiertos que proporciona la Junta de Castilla y León. La Junta, prácticamente solo añade eventos de bibliotecas y eventos en los auditorios que ellos gestionan, sesgando mucho los datos.

AMENAZAS

La aplicación solo utiliza el portal de datos abiertos de la Junta de Castilla y León como fuente de los datos. Si la Junta decide dejar de actualizar dicho portal, la aplicación no serviría de nada.

FORTALEZAS

La interfaz de la aplicación es muy sencilla e intuitiva, lo que permite a los usuarios poder utilizarla sin necesidad de mirar la documentación. La aplicación, además, comprueba si existe algún evento duplicado. Esta funcionalidad es útil si se implementan varias fuentes de datos.

OPORTUNIDADES

La aplicación esta programada de una forma que permite la integración con otras fuentes de datos de una manera muy fácil, permitiendo aumentar la base de datos. Por otro lado, se puede hacer consultas a la API de datos abiertos para evitar tener que descargar el CSV cada vez que se quieran actualizar los datos.

8. PROXIMOS PASOS

El siguiente paso necesario para continuar con la aplicación es conseguir más fuentes de información para poder aumentar el tamaño de la base de datos.

Otra posible mejora es la creación de perfiles de usuario, de forma que podamos recopilar datos de los intereses de cada usuario y, aplicando técnicas basadas en Inteligencia Artificial poder ofrecer unas recomendaciones basadas en búsquedas anteriores. Para realizar esta idea es necesario una gran cantidad de usuarios, ya que es muy fácil que la información se sesgue.

Durante la implementación de la aplicación se ha intentado implementar LlamaIndex, pero no tenia la suficiente cantidad de datos ni de usuarios, ni de eventos como para poder crear unas

recomendaciones eficaces. Se probó a utilizar CHATGPT, pero los resultados no tenía ninguna relación con los datos de entrada.

Para poder implementar una IA, veo necesario implementar un sistema híbrido en el que, a través de la interacción de los usuarios, se vayan creando las recomendaciones, tal como hace Spotify o Netflix. Así, cuanto más interactúe el usuario, mejores serán estas recomendaciones.

9. REFERENCIAS Y HERRAMIENTAS UTILIZADAS

- https://graphacademy.neo4j.com/?ref=redirect
- https://analisis.datosabiertos.jcyl.es/explore/dataset/eventos-de-la-agenda-cultural-categorizados-y-geolocalizados/export/
- https://github.com/asant05/SIBI-Viajes/
- https://www.youtube.com/watch?v=YcOzzDgVFgk&ab_channel=Graphileon
- https://edit-csv.net/
- https://www.llama2.ai/
- https://arxiv.org/abs/2401.04057
- https://vuejs.org/guide/introduction.html
- https://nodejs.org/docs/latest/api/

_