**Escuela de Ingenierías**

**Industrial, Informática y Aeroespacial**

**GRADO EN INGENIERÍA INFORMÁTICA**

Sistemas de Información de Gestión y Bussiness Intelligence

Memoria de la aplicación web: FutbolFichajes

Nombre del Alumno:

Aitor Del Río Ferreras

Resumen

Debido a la situación actual vivida por la pandemia, el mundo del futbol, al igual que en otros campos, sufre los estragos del virus en el tema económico. Muchos equipos se ven en la situación de tener que mirar mucho su situación económica para realizar nuevas incorporaciones en sus plantillas y para mantener la plantilla que tiene por lo que ofrecemos un sistema que ayudará a poder buscar jugadores de varias formas para poder utilizar mejor sus presupuestos.

Índice

[Glosario de símbolos o Terminología 4](#_Toc57812336)

[Introducción 5](#_Toc57812337)

[Desarrollo 6](#_Toc57812338)

[*1.* *DESCRIPCIÓN DEL PROBLEMA* 6](#_Toc57812339)

[*2.* *HERRAMIENTAS UTILIZADAS* 7](#_Toc57812340)

[*3.* *COMPOSICIÓN DE LA APLICACIÓN* 9](#_Toc57812341)

[1. Base de datos 10](#_Toc57812342)

[4. Backend 13](#_Toc57812343)

[5. Frontend 13](#_Toc57812344)

[a. Mostrar jugadores aleatorios 13](#_Toc57812345)

[b. Mostrar jugadores en base a filtros 13](#_Toc57812346)

[c. Mostrar los mejores jugadores 13](#_Toc57812347)

[d. Mostrar los jugadores promesas 13](#_Toc57812348)

[*4.* *ANÁLISIS DE RESULTADOS* 13](#_Toc57812349)

[*5.* *DAFO* 13](#_Toc57812350)

[*6.* *LÍNEAS DE FUTURO* 13](#_Toc57812351)

[*7.* *LECCIONES APRENDIDAS* 13](#_Toc57812352)

[Conclusión 15](#_Toc57812353)

[Bibliografía y referencias 16](#_Toc57812354)

# Glosario de símbolos o Terminología

Frontend: Parte exterior de una aplicación.

Backend: Parte interna de una aplicación que trabaja con los datos.

# Introducción

En este documento se expondrá el trabajo realizado para lograr terminar la aplicación web de recomendación de fichajes.

Primero se realizará una exposición detallada del problema que intentaremos resolver con nuestra aplicación y posteriormente daremos una explicación de su composición y funcionamiento.

La explicación se realizará desde abajo a arriba, es decir, empezaremos explicando la parte de persistencia de datos, donde analizaremos como hemos utilizado la tecnología neo4j.

Seguidamente explicaremos como hemos utilizado la tecnología node js para realizar el backend, que será la capa que hará de intermediario recibiendo peticiones del frontend y trasladándolas a la base de datos y después, cuando reciba los datos. Se los devolverá al frontend.

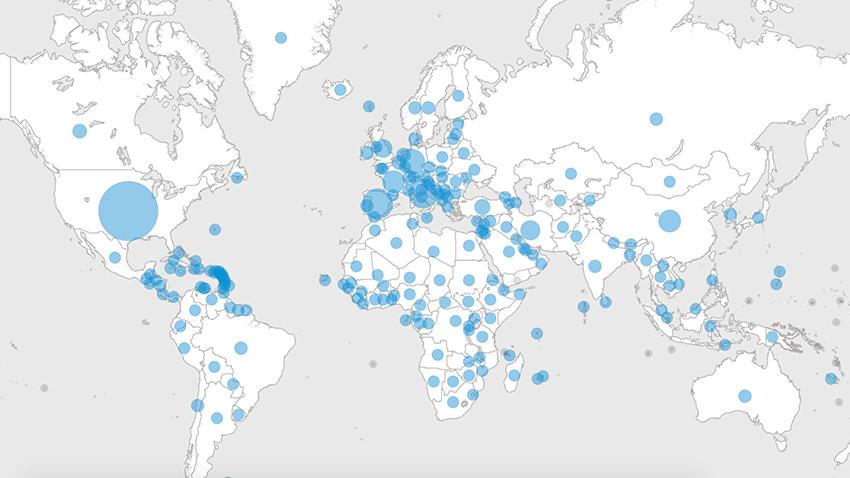
Acto seguido explicaremos como hemos utilizado la tecnología Vue js y en particular Vuetify para implementar una interfaz de usuario.

Para finalizar, explicaremos el funcionamiento de varios casos de uso de la aplicación web, explicando el recorrido tanto fuera de la aplicación (es decir, las acciones que hacemos y los datos que se nos muestra) y también por dentro (haremos un breve recorrido desde por ejemplo la pulsación de un botón hasta la obtención de los datos).

# Desarrollo

1. *DESCRIPCIÓN DEL PROBLEMA*

Debido a la pandemia que esta asolando a cualquier rincón del planeta, tanto en el tema sanitario, por la perdida de tantas vidas humanas, como en lo económico, con las terribles consecuencias que esta trayendo como puede ser perdidas de trabajo, salarios mas bajos o economías mas bajas a nivel global. En nuestro caso nos vamos a centrar en los estragos que esta sufriendo el mundo del deporte, mas concretamente el mundo del futbol.



En el mundo del futbol, solo la ausencia de publico en las gradas de los estadios ha producido que los equipos vean que su nivel económico ha caído drásticamente hasta el punto de que numerosos equipos han acabado sus años con balances económicos en negativo con números negativamente altos. Debido a esto, los equipos necesitan rentabilizar mas el dinero que tienen, ya que es poco y, por otra parte, necesitan cada año hacer proyectos nuevos para poder reinar tanto en su país, como en su continente y a nivel mundial y para ello, una parte muy importante de estos proyectos son la salida y entrada de nuevos jugadores.

Gracias a nuestra aplicación, permitiremos a los clubes poder consultar datos sobre jugadores los cuales ellos mismos podrán introducir las características que quien que ellos tengan para que puedan ficharlo pudiendo observar tanto sus características personales como las económicas.

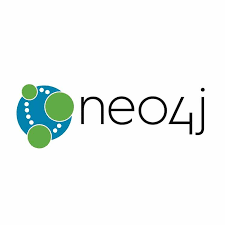
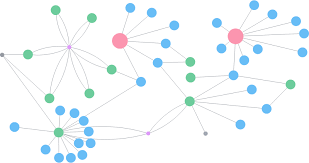
1. *HERRAMIENTAS UTILIZADAS*

Para realizar la aplicación, se nos otorgaba una gran libertad a la hora de elegir lenguaje y tipo de programación (ya sea una aplicación de escritorio o una aplicación web) pero había una sola condición, se debía utilizar el sistema de gestión de bases de datos de grafos NEO4J.

NEO4J permite crear y gestionar bases de datos en forma de grafos, unas bases de datos muy utilizadas en el ámbito empresarial y cada vez mas utilizadas a nivel global.

Estas bases de datos nos permiten crear nodos de un tipo de datos, en mi caso nodos de tipo jugador y equipo, y relacionar los nodos por medio de relaciones, aristas que unen dos nodos dándole un significado a la relación. Los nodos pueden tener propiedades, datos que otorgan datos a los nodos con los que luego poder diferenciarlos o para trabajar con los nodos.

Las bases de datos por grafos, a diferencia de las que son relacionales que utilizan tablas para relacionarse con otras tablas, los nodos no tienen que estar relacionados con otros estrictamente solo por el tipo de nodo que sea, algo que ocurre con las bases de datos relacionales.

Para usar correctamente NEO4J en nuestro programa, se realizó una búsqueda exhaustiva en una pagina web denominada Kaggle. Kaggle es una pagina que posee una gran cantidad de datos para proporcionar a los concursantes de las competiciones que propone que suelen basarse en machine learning o en el campo de la ciencia de datos en su mayor medida. Obtuve una base de datos de un popular videojuego denominado “FIFA 21”. Es un archivo con extensión .csv donde los jugadores tenían los datos que necesitaba sin tener campos vacíos o nulos.

Tabla

Descripción generada automáticamente

Aquí tenemos un pequeño ejemplo de los datos que utilizamos para trabajar en la aplicación. El archivo csv viene con datos separados por el separador “;”.

Para poder “jugar” con los datos, esto es, manipularlos y trabajar adecuadamente con ellos, teníamos que introducir los datos del archivo en la base de datos, para ello utilizamos el lenguaje que entiende NEO4J, CYPHER. Este lenguaje nos permite trabajar con las bases de datos de NEO4J de una forma muy cómoda debido a que nos ahorra mucho código respecto a, por ejemplo, sql y además nos proporciona un lenguaje muy similar al lenguaje natural, lo que hace que sea un lenguaje mas sencillo de aprender.

Después de haber hablado de lo referente a la base de datos y las herramientas que conlleva, hablaremos de las herramientas utilizadas en el backend de la aplicación. Para realzar el backend de la aplicación, además de utilizar el lenguaje JavaScript, hemos utilizado NodeJs, que es un entorno de programación en tiempo real utilizado para crear backend en aplicaciones web. Dentro de node js hemos utilizado las siguientes herramientas:

1. Express Js: entorno de programación que nos permite realizar todas las funciones propias de HTTP (get,put,post y delete).
2. Neo4j driver: driver que permite la conexión del backend desde nodejs con la base de datos neo4j y realizar peticiones a datos de la base de datos.

El backend de la aplicación se compone de todo el código necesario para recibir consultas por parte del cliente y se las traslada a la base de datos, realiza la petición y le devuelve los datos la base de datos al backend, que posteriormente se le devolverá al cliente. Para realizar esta aplicación, se utiliza la técnica de programación MVC que después explicaremos mas profundamente.

El frontend este compuesto del código que proporciona una vista con la que el usuario se comunica y le realiza peticiones que posteriormente irán al backend para ser procesadas. Para realizar el frontend se utilizan las siguientes herramientas:

1. VUE js: Framework que permite la creación, de forma sencilla, de aplicaciones web proporcionando gran facilidad de introducción de diferentes librerías para trabajar juntamente con Vue js.
2. Vuetify: Framework que combina vuejs con la estética de material design para poder crear la interfaz de usuario.
3. Axios: Librería de JavaScript que nos permite realizar peticiones HTTP desde el cliente. Lo utilizaremos desde el frontend para realizar las peticiones al backend.
4. *COMPOSICIÓN DE LA APLICACIÓN*

En esta aplicación, hemos implantado un sistema de programación denominado MVC(modelo vista controlador). En este tipo de sistema, los datos se mueven desde una base de datos o un sistema de persistencia de datos teniendo una capa entre medias, el modelo. Explicaremos detenidamente cada parte de este sistema:

* Modelo: Es la capa que trabaja con los datos. Esta capa esta conectada con la base de datos y es la que trabaja con los datos, es decir, esta capa es quien realiza peticiones a la base de datos para obtener datos, según unos filtros o para realizar cambios en datos dentro de la base de datos.
* Vista: Es la capa que contiene el código para visualizar la interfaz de usuario con la que el usuario interactuara para obtener algún tipo de salida. En la vista el usuario introducirá unos datos y a través del controlador, se le pasara al modelo y es quien pedirá los datos oportunos.
* Controlador: Su función es recibir las ordenes que el usuario introduce por medio de la interfaz de usuario, en su defecto consola de comandos, y trasladárselas al modelo para que realice las funciones pertinentes con los datos y después se las devuelva al controlador y este a la vista.

Interfaz de usuario gráfica, Diagrama, Aplicación

Descripción generada automáticamente

1. Base de datos

Para la base de datos, hemos utilizado el sistema gestor de bases de datos de grafos Neo4j, que nos permite generar bases de datos en forma de grafos a partir de un archivo con formato csv. La base de datos esta compuesta por 19.422 nodos los cuales son de dos tipos:

* Jugador: representan un jugador de futbol de un equipo. Todos los nodos de este tipo tienen las siguientes propiedades:
  + Edad: Representa la edad de un jugador.
  + Equipo: Representa el equipo donde juega.
  + Id: Es un numero identificador del jugador.
  + Nacionalidad: Representa al país al que pertenece el jugador.
  + Nombre: nombre común o a partir del cual se le conoce al jugador.
  + NombreCompleto: nombre completo del jugador.
  + Posición: Posición en la que el jugador juega en el campo (la abreviatura esta en ingles, en el frontend se traduce al castellano).
  + Potencial: Es la puntuación máxima que podrá alcanzar un jugador en un futuro.
  + Precio: precio, en euros, que cuesta un jugador.
  + Puntuación: es la puntuación actual de un jugador la cual puede mejorar o empeorar en un futuro.
  + Salario: Dinero que cobra un jugador, en euros, por semana.
* Equipo: representa el equipo donde una serie de jugadores juega. Las propiedades que tiene son las siguientes:
  + Id: Identificador el equipo.
  + Nombre: nombre completo del equipo.

Para poder obtener estos datos, se obtienen a partir del archivo, con extension csv que se encuentra en la carpeta etc “baseDatosFifaCompleto.csv”.

Para cargar este fichero en la base de datos se haria introduciendo este comando:

Imagen que contiene Tabla

Descripción generada automáticamente

Este comando realiza las siguientes funciones:

1. Carga el fichero csv con las cabeceras con el comando “load csv headers from…” y le dice a la base de datos que los campos están separados por “;” con la sentencia “FIELDTERMINATOR ‘;’’” y guardamos cada fila en la palabra row.
2. Crea un nodo de tipo Jugador rellenando cada propiedad con los datos que hay en cada columna de la fila row.
3. Busca si hay un nodo de tipo Equipo con el campo que hay en la columna equipo del jugador.
   1. Si el equipo no existe, crea un nodo de tipo Equipo con el nombre del equipo y crea una relación JUEGA\_EN desde el nodo Jugador al nodo Equipo.
   2. Si el equipo existe, toma la referencia del nodo Equipo que contiene el nombre del equipo y realiza la relación JUEGA\_EN desde el nodo Jugador al nodo Equipo.

La base de datos cuenta con 18.741 relaciones que unen los nodos de tipo Jugador a los nodos de tipo Equipo con una relación de tipo JUEGA\_EN.

1. Backend

Para implementar el backend, como antes hemos mencionado, hemos utilizado la herramienta Node js y hemos programado con JavaScript utilizando los drivers de express js y neo4j.

Todo el backend se encuentra en codigo en el fichero que se encuentra en la carpeta Backend del proyecto app.js mientras que en el resto de las carpetas de dicho directorio se encuentran las dependencias que necesita el fichero para ejecutarse.

A continuacion, vamos a explicar, funcion por funcion las tareas que realiza la base de datos:

* dameJugadoresPromesas:

Texto

Descripción generada automáticamente

En esta funcion se realizan los siguientes pasos para alcanzar nuestro objetivo: poder mandar al frontend una lista con los diez jugadores cuyo potencial sea mayor y que su edad este entre 15 y 22 años:

Una vez que tenemos la consulta, se la mandamos a la base de datos con el metodo run de la sesion que teniamos abierta. Este metodo nos devuelve 3 situaciones:

* + onNext: nos devuelve fila a fila la tabla con todos los nodos desde la base de datos. Lo que hago es guardar cada fila,cogiendo sus propiedades, en la lista que voy a devolver al frontend.
  + onCompleted: lo devuelve cuando ya termino de devolver todos los datos referentes a la consulta por lo que dentro de esta funcion podremos devolver al frontend la lista con los datos devueltos de la base de datos.
  + onError: es devuelto cuando se produce un error con la base de datos.
* dameLosMejores:

Texto

Descripción generada automáticamente

La finalidad de esta funcion es la de devolver los mejores jugadores de la base de datos. Para devolver los mejores, lo que hacemos es decirle que devuelv los jugadores que mejor puntuacion tienen devolviendo los 10 mejores jugadores. Para ello, se realizan las mismas funciones que la funcion anterior ya que guarda en la lista las filas que va devolviendo la base de datos y cuando acaba, la funcion devuelve la lista al frontend.

* dameJugadoresAleatorios:

Texto

Descripción generada automáticamente

La finalidad de esta funcion es la de devolver jugadores sin una dinamica fija mas que elegir un numero aleatorio, que representara el potencial y en base a ese numero aleatorio, se le pedira a la base de datos que devuelva los jugadores, como en los casos anteriores, en una lista.

* dameNacionalidades:

Texto

Descripción generada automáticamente

Esta funcion se utiliza para poder tener en un desplegable en el frontend todas las nacionalidades que hay en la base de datos. Al igual que en las funciones anteriores, se envia una consulta a la base donde le decimos que queremos las nacionalidades distintas para que no haya repeticiones y cuando esas nacionalidades estan guardadas en la lista, podemos enviarsela al frontend.

* dameEquipos:

Texto

Descripción generada automáticamente

Al igual que la funcion dameNacionalidades, el objetivo de esta funcion es el de poder mostrar, en un desplegable del frontend, los distintos equipos que hay en la base de datos por lo que le pedimos en la consulta que devuelva equipos para que no se produzca repeticiones y que despues envie la lista con los equipos.

* dameJugadoresConFiltros:

Texto

Descripción generada automáticamenteTexto

Descripción generada automáticamenteTexto

Descripción generada automáticamente

Esta funcion es la que mas trabajo a nivel de desarrollo y a nivel computacional genera. En esta funcion, la finalidad es la de devolver una serie de jugadores según unos

1. Frontend
   1. Mostrar jugadores aleatorios
   2. Mostrar jugadores en base a filtros
   3. Mostrar los mejores jugadores
   4. Mostrar los jugadores promesas
2. *ANÁLISIS DE RESULTADOS*
3. *DAFO*
4. *LÍNEAS DE FUTURO*
5. *LECCIONES APRENDIDAS*

Una vez realizado este trabajo, puedo afirmar que soy capaz de aprender nuevas herramientas por mi cuenta, aunque debo dar las gracias a mi profesor de la asignatura por recomendarme el sistema de bases de datos neo4j.

He descubierto un mundo de bases de datos totalmente nuevo, las bases de datos por grafos y el lenguaje cypher que hace que las consultas sean muy sencillas.

Para aprender este lenguaje, tuve que realizar varios tutoriales de la pagina de neo4j además de haber visto varios videos, todos ellos de trabajadores de neo4j.

También estuve leyendo ciertos fragmentos de algún libro, recomendado por el profesor que me ayudo a reforzar los conocimientos que tenia sobre los grafos.

Me he dado cuenta de que soy capaz de aprender cosas nuevas por mi mismo ya que el profesor nos dio la idea y viendo los tutoriales pude aprender en varias semanas como funciona las bases de datos neo4j además de entender como funciona el lenguaje cypher.

Realmente el lenguaje cypher y el SQL tienen muchas cosas en común ya que ambos realizan consultas y peticiones a una base de datos.

Si que es cierto que neo4j y SQL son muy diferentes dado que uno crea bases de datos formadas por tablas que tienen relaciones entre ellas mientras que las bases de datos neo4j, es decir, las bases de datos de grafos unen nodos de cierto tipo de objeto, pero no todos los nodos de ese objeto tienen porque estar unidos a un nodo en especial.

También he aprendido a organizarme por mi cuenta a la hora de desarrollar una aplicación. He creado mi propia dinámica de desarrollo de software, al menos a la hora de hacer aplicaciones web que estén compuesta por frontend y backend. Mi nueva dinámica seria la siguientes:

1) crear la base de datos sin tener en cuenta que tipo sea, es decir, sin tener en cuenta que sea una SQL, no-SQL o de grafos.

2)crear el backend con sentencias sencillas para poder conectarlo con la base de datos mediante los drivers necesarios (en este caso, el driver necesario fue el de neo4j).

3)crear el frontend para poder conectarlo al backend, en este caso con la herramienta axios y así poder realizar peticiones al backend.

Una vez creado todas las conexiones entre los distintos elementos, cuando creaba las peticiones a la base de datos, implementaba el código necesario y posteriormente utilizaba la herramienta postman para comprobar que el código realizado funcionaba correctamente y una vez comprobado esto, podemos hacer las peticiones con el frontend directamente.

Al haber creado esta dinámica, posiblemente ya existente, me ayudaba a llevar cierta rapidez a la hora de realizar el código y comprobar que funcionaba correctamente.

# Conclusión

Gracias a este trabajo, he sido capaz de, por mi mismo, aprender un nuevo lenguaje como es cypher y un nuevo sistema de bases de datos y a programar y entender que es un sistema de recomendación, para que se utiliza, que tipos de sistemas de recomendación existen y que beneficio se obtienen de ellos en el mundo real.

# Bibliografía y referencias

Graph database: bases de datos para una interconexión eficiente: <https://www.ionos.es/digitalguide/hosting/cuestiones-tecnicas/graph-database/#:~:text=A%20diferencia%20de%20las%20de,con%20diferentes%20valores%20de%20atributo>

Plataforma para hacer competiciones de inteligencia artificial: <https://inlab.fib.upc.edu/es/blog/plataforma-para-hacer-competicions-de-inteligencia-artificial>

Qué es NodeJS y para qué sirve: <https://openwebinars.net/blog/que-es-nodejs/>

MVC (Model, View, Controller) Explicado.: <https://codigofacilito.com/articulos/mvc-model-view-controller-explicado>

Qué es MVC: <https://desarrolloweb.com/articulos/que-es-mvc.html>