Sistema de Previaturas de Ingeniería en Computación modelado sobre una base de datos basada en grafos

Santiago Calvo *Instituto de Computación*
Facultad de Ingeniería, Universidad de la República
Montevideo, Uruguay
santiago.calvo.vello@fing.edu.uy
Emiliano de Sejas *Instituto de Computación*
Facultad de Ingeniería, Universidad de la República
Montevideo, Uruguay
emiliano.de.sejas@fing.edu.uy

Resumen

Este informe estudia el uso de bases de datos no relacionales basadas en grafos para la representación, gestión y consumo de sistemas de previaturas para la planificación y desarrollo de la carrera universitaria de un estudiante. A través del uso de este tipo de bases de datos, se logra una visión más intuitiva y dinámica del progreso de los estudiantes durante la carrera. Ofrece a los estudiantes una visualización más clara de la trayectoria académica restante y facilita la toma de decisiones informadas sobre la misma, lo que permite una gestión adecuada del progreso.

I. Introducción

En el paisaje académico actual, el seguimiento y la gestión del progreso estudiantil representan aspectos esenciales en la planificación educativa. Tradicionalmente, estos procesos se han llevado a cabo mediante sistemas basados en bases de datos relacionales, donde la información de los estudiantes y las materias cursadas se almacenan en tablas interrelacionadas. Aunque este enfoque ha demostrado ser útil, presenta limitaciones al intentar modelar y visualizar de manera efectiva la red de interdependencias entre las diversas materias en términos de previaturas, es decir, las asignaturas que deben ser cursadas antes de poder inscribirse en otras.

Este informe presenta una nueva aproximación a este desafío, explorando el uso de una base de datos no relacional basada en grafos para modelar el grafo de previaturas en un entorno académico. Estas bases de datos, con su capacidad para representar y manipular relaciones complejas entre entidades, ofrecen un marco potencialmente superior para modelar estas interdependencias de manera más intuitiva y dinámica.

Aquí, comenzaremos con una discusión detallada de la estructura y características de las bases de datos basadas en grafos, enfocándonos en cómo pueden ser utilizadas para capturar de forma eficiente las relaciones entre las diversas asignaturas y sus requisitos previos. Posteriormente, describiremos la metodología y el proceso que utilizamos para modelar el grafo de previaturas, ilustrando cómo este enfoque permite una representación más precisa y manejable de la trayectoria de cada estudiante.

Con este estudio, buscamos no solo proporcionar una herramienta más robusta y flexible para la gestión académica, sino también mejorar la capacidad de los estudiantes para entender y planificar su trayectoria educativa. Además, pretendemos abrir la puerta a futuras investigaciones y desarrollos en la utilización de bases de datos basadas en grafos en el ámbito educativo y de productos comerciales.

II. Trabajos relacionados

Existen dos precedentes fundamentales para el presente estudio.

El primero es el sitio web de Bedelías de la Universidad de la República (UdelaR). Este portal web constituye la fuente primaria de todos los datos relacionados con las asignaturas y sus respectivas previaturas. Sin embargo, esta plataforma, implementada con una base de datos relacional, presenta una interfaz de usuario poco intuitiva que complica la comprensión de la trayectoria académica. Asimismo, la visualización de las previaturas es limitada, no permitiendo observar la totalidad de los niveles de interdependencia entre las asignaturas.

El segundo precedente es el sitio web MiCarrera.uy. Esta es un plataforma de código abierto que extrae sus datos del sitio web de Bedelías. Al igual que el portal de Bedelías, MiCarrera.uy utiliza una base de datos relacional y efectúa gran cantidad de evaluaciones a nivel de aplicación, no a nivel de la base de datos. Esto puede dar lugar a problemas de rendimiento con un número elevado de materias, a pesar de que en una etapa inicial el sistema parece tener un buen desempeño. Esta aplicación

permite a los usuarios seleccionar las asignaturas cursadas y ver cuáles son las asignaturas que pueden cursar a continuación, así como visualizar las materias que son requisito para cada asignatura.

Estos dos precedentes proporcionan un contexto crucial para el presente trabajo, que busca superar las limitaciones identificadas a través de la implementación de una base de datos no relacional basada en grafos para modelar el grafo de previaturas.

III. Beneficios de las bases de datos basadas en grafos

Éstas bases de datos ofrecen varios beneficios significativos que las hacen útiles para una variedad de aplicaciones. A continuación, se describen algunos de los beneficios más notables:

Capacidad de expresar relaciones complejas: Las bases de datos basadas en grafos son particularmente útiles para modelar y consultar relaciones complejas entre entidades. Esta capacidad es especialmente relevante en escenarios donde las relaciones entre los datos son tan importantes como los datos en sí mismos.

Eficiencia de consultas: Las bases de datos de grafos son generalmente más eficientes que las bases de datos relacionales para realizar consultas que implican relaciones múltiples, gracias a la forma en que almacenan y recuperan los datos. Neo4J, en particular, nos permite recorrer la red formada por el grafo similar a como lo haría Facebook para encontrar relaciones entre amigos [3].

Flexibilidad de esquema: A diferencia de las bases de datos relacionales, las bases de datos de grafos son inherentemente flexibles en términos de esquema. Esto significa que pueden adaptarse fácilmente a cambios en los datos o en los requerimientos del negocio.

Escalar horizontalmente: Las bases de datos de grafos pueden escalarse horizontalmente para manejar grandes volúmenes de datos, distribuyendo los datos a través de múltiples servidores para mejorar el rendimiento y la capacidad de almacenamiento. [2]

Agilidad en desarrollo: Las bases de datos de grafos facilitan la iteración y evolución rápida del modelo de datos, lo que las hace útiles en entornos de desarrollo ágil y en proyectos donde los requerimientos pueden cambiar con el tiempo.

Intuitividad: Los grafos son intuitivos y fáciles de entender, lo que facilita la exploración y comprensión de los datos por parte de los usuarios y en el desarrollo.

En conjunto, estos beneficios hacen de las bases de datos basadas en grafos una opción atractiva para una variedad de aplicaciones, desde redes sociales y motores de recomendación hasta detección de fraudes.

IV. Aplicación de base de datos de grafos sobre el problema

El grafo de previaturas de la Facultad de Ingeniería constituye un escenario idóneo para la implementación de bases de datos de grafos. Como su nombre sugiere, las relaciones complejas inherentes entre las materias, particularmente en lo que concierne a los requisitos de previatura, originan desafíos significativos durante el desarrollo, tanto en términos de verificación de integridad como en la propia gestión de los datos.

Los grafos se erigen como una estructura de datos excepcionalmente apropiada para este tipo de problemáticas, debido a que, en esencia, lo que buscamos examinar son los trayectos existentes entre las asignaturas. Esto no sólo facilita la verificación de requisitos, sino que también simplifica la generación de recomendaciones de materias para los estudiantes, un factor crítico en la toma de decisiones respecto a la continuación de su trayectoria académica. En este sentido, la utilización de bases de datos de grafos genera la capacidad de transformar el modo en el que se gestiona y se planifica el progreso académico en el entorno universitario.

V. Implementación y Ejecución

Para la implementación del sistema fue necesario realizar diversos pasos de los cuales algunos se describen a continuación.

V-A. Modelo de datos

El modelado de los datos sobre el grafo se representa en la figura 1. Los componentes más críticos de este modelo son los nodos *Subject*, que representan las asignaturas, y *Prerequisite*, que indican los requisitos de las asignaturas.

Como se ilustra en la figura, una asignatura no solo tiene requisitos, sino que también es necesaria para cumplir otros requisitos. A su vez, un requisito puede ser requisito para otros. De este modo, es posible modelar estructuras complejas, similares a las que se encuentran en la página de bedelías, como se puede apreciar en la figura 2.

Estos requisitos introducen diversas complejidades, tales como la necesidad de cumplir solo algunos (”tener solo alguna”), la restricción de no cumplir ciertos requisitos (”no debe tener”) y, por supuesto, la necesidad de cumplir todos los requisitos (”debe tener todas”). Estas complejidades se pueden modelar en un grafo de manera bastante equivalente, pero con la ventaja de la gestión de nodos. Así, podemos tener requisitos anidados dentro de otros requisitos y relaciones de éstos con las asignaturas.

Un ejemplo de esta modelización en nuestra base de datos basada en grafos se puede ver en la figura 3, aunque se han excluido las relaciones del tipo ”no debe tener”para simplificar la visualización.

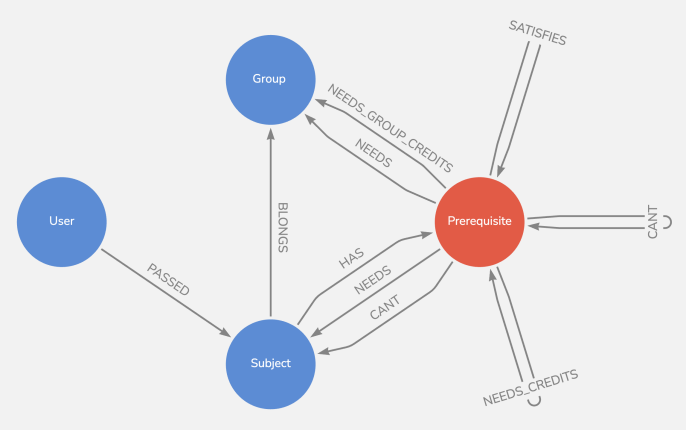

Figura 1: Esquema de grafo.

Figura 2: Ejemplo bedelias.

V-B. Carga de datos

Para el desarrollo de este estudio, fue imperativo realizar una carga de datos significativa. Desafortunadamente, no existía ninguna fuente de datos disponible que pudiera ser importada de manera sencilla a la base de datos. Por consiguiente, se vio la necesidad de concebir un proceso capaz de leer los datos requeridos y generar las consultas correspondientes. Dado el volumen de los datos y su carácter dinámico, no era factible realizar una conversión manual de los mismos.

Aprovechando las capacidades de Typescript, se procesaron los datos a partir de un archivo YML, generando consultas dinámicas para crear los datos necesarios. Estos datos abarcaban elementos tales como las asignaturas, los grupos y los prerrequisitos de las asignaturas. Las consultas se formularon utilizando el lenguaje Cypher y se ejecutaron en una base de datos Neo4j, proporcionando un ambiente propicio para la experimentación y la subsiguiente creación de la aplicación.

En lo que respecta a la subida de datos, el enfoque se centró en lograr una funcionalidad óptima, dejando en segundo plano los aspectos relacionados con el rendimiento. Este enfoque resultó en la generación de una serie de consultas numerosas y de eficiencia limitada, pero que lograron cumplir con el objetivo principal del estudio. Este aspecto, sin embargo, representa un área potencial de mejora en futuros trabajos de investigación y optimización.

V-C. Consultas relevantes

Se implementaron diversas consultas de interés, presentando cada una de estas un nivel de complejidad progresivamente superior. Nuestra base de datos basada en grafos tiene como objetivo principal facilitar del proceso de adquisición e interpretación de información pertinente acerca de las asignaturas que un estudiante puede cursar a lo largo de su recorrido académico.

La primera consulta, aunque aparentemente evidente, aborda la obtención de todas las asignaturas disponibles, acompañadas de la información relativa a las que el usuario ha cursado. Aunque los detalles específicos de las consultas están consignados en el anexo VIII, la primera consulta puede resumirse como: ”Dado un usuario, obtener todas las asignaturas y determinar, para

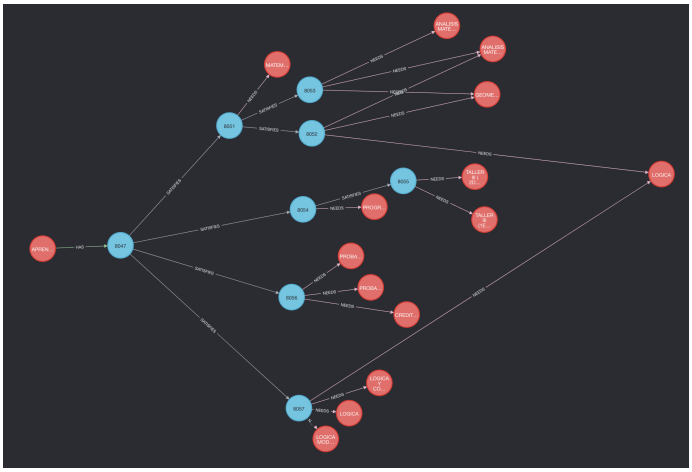

Figura 3: Ejemplo Grafo Neo4j.

cada una, si existe una relación del tipo *PASSED* entre la asignatura y el usuario". Posteriormente, el volumen de resultados fue limitado para evitar sobrecargar al usuario con una excesiva cantidad de asignaturas inmanejables en una sola vista, optimizando simultáneamente el rendimiento del sistema. 1

La subsiguiente consulta en orden de importancia se relaciona con las asignaturas recomendadas. Dichas asignaturas no necesariamente pueden ser cursadas inmediatamente por el usuario, pero sí representan las materias hacia las que existen potenciales rutas a seguir. Estas se clasifican en base a la cantidad de caminos disponibles. Dichos caminos se identifican a partir de las asignaturas que mantienen una relación de *PASSED* con el usuario, y de ellas, explorando las relaciones de hacia los prerequisitos. Se ordenan en base a la cantidad de prerequisitos satisfechos. 2

Finalmente, se implementaron tres consultas adicionales que se utilizaron de manera conjunta para identificar las asignaturas que el estudiante tiene la capacidad de cursar. Obtener este resultado implicó la ejecución de una serie de evaluaciones en la capa de aplicación. Esta necesidad surgió debido a que *Neo4J* presenta limitaciones al explorar un grafo con condiciones dinámicas, una situación que se asemeja a la navegación por un árbol de decisiones. Teóricamente, este proceso podría haberse ejecutado a nivel de base de datos mediante la creación de extensiones con el lenguaje Java, pero las limitaciones temporales nos llevaron a buscar alternativas. 3

En lugar de extender la funcionalidad de la base de datos, se optó por realizar algunas evaluaciones en la aplicación web, lo que proporcionó una mayor flexibilidad a costa de una disminución en el rendimiento. A partir de las recomendaciones, que incluyen todas las asignaturas con alguna conexión con el estudiante, se obtuvo el grafo completo de requerimientos para cada asignatura en forma de caminos. Este grafo se reconstruyó en la aplicación para que, tras la obtención de las asignaturas cursadas y la suma de créditos en cada grupo de asignaturas, se pudiera evaluar de forma recursiva. El resultado de este proceso de evaluación son las asignaturas que el estudiante tiene la capacidad de cursar.

Este enfoque conlleva un coste significativo en términos de rendimiento, dado que implica la realización de tres consultas independientes y la reconstrucción del grafo en la aplicación. Esto genera un impacto considerable en el tiempo requerido para el procesamiento y la memoria necesaria.

V-D. Interfaz de usuario

A fin de corroborar la eficacia y la facilidad de uso de la base de datos implementada, se desarrolló una aplicación que interactúa con la misma. La intención tras este esfuerzo fue la de facilitar la ejecución de consultas de manera intuitiva, permitiendo una interacción amigable y eficiente con los datos almacenados. Para lograr la aplicación, se hizo uso de tecnologías contemporáneas, incluyendo NextJS y React, junto con el conector proporcionado por Neo4J, que facilita la comunicación entre la aplicación y la base de datos.

Pese a la aparente complejidad de la tarea, el proceso de desarrollo de la aplicación no presentó obstáculos significativos. Se logró implementar un sistema de manejo de usuarios robusto, utilizando un servicio externo para la autenticación y gestión de los perfiles. En conjunto con la base de datos, se conserva un registro detallado de las asignaturas que cada usuario ha cursado, permitiendo un seguimiento individualizado de la trayectoria académica.

Gracias a las consultas dinámicas previamente detalladas y ejecutadas sobre la base de datos, la aplicación es capaz de generar recomendaciones personalizadas para cada estudiante. Asimismo, permite consultar en todo momento las asignaturas acutalmente disponibles para ser cursadas por el usuario. Esta última es sumamente útil, puesto que de manera personalizada le ofrece a cada usuario una perspectiva del avance en su carrera.

Este tipo de funcionalidad es un claro testimonio del potencial que albergan las bases de datos basadas en grafos en el ámbito académico. No sólo posibilitan una planificación más eficiente del itinerario de estudio, sino que también propician

un enfoque personalizado que responde a las necesidades y expectativas de cada estudiante. Esta conjunción de eficacia y adaptabilidad subraya la validez de nuestro enfoque y sugiere la existencia posibilidades interesantes para futuras investigaciones y aplicaciones.

La interfaz de usuario se puede encontrar en https://proyecto-bdnr-bwg7.vercel.app/. El código fuente de esto se encuentra en https://gitlab.fing.edu.uy/emiliano.de.sejas/proyecto-bdnr-grupo5/-/tree/main/web.

VI. Experimentación

VI-A. Datos utilizados

Inicialmente, se intentó recopilar la información del sistema de previaturas del sistio de bedelías. Esta tarea presentó varias dificultades y debido a las restricciones de tiempo se decidió utilizar los datos del respositorio abierto del proyecto Mi Carrera[1]. Estos han supuesto una buena base que nos permitió disminuir el tiempo invertido en la obtención de datos, y poder enfocarnos en la implementación. [1]

Estos datos se obtuvieron de la misma forma que habíamos planificado inicialmente, utilizando un *web scraper* que puede verse también en el repositorio de github [2] del proyecto mencionado.

VI-B. Resultados

El producto final de este proceso es una aplicación que ejecuta las consultas mencionadas anteriormente. Como ya mencionamos, algunas de estas consultas llevan a cabo evaluaciones en la capa de aplicación, por lo que sus resultados deben analizarse teniendo en cuenta este factor.

Estas consultas son utilizadas para ofrecer a un alumno, la capacidad de agregar las materias que ya aprobó, para luego obtener recomendaciones de posibles materias a seguir y conocimiento de las materias que actualmente tiene disponibles para cursar.

La tabla I presenta los promedios de los tiempos de ejecución de 100 consultas para cada ruta del servidor. Dado que todos estos tiempos promedio son inferiores a un segundo, inferimos que un usuario promedio puede tener una experiencia positiva utilizando esta aplicación.

Es pertinente destacar que la aplicación MiCarrera.uy presenta tiempos de respuesta entre $500ms$ y $800ms$. Aunque una comparación exhaustiva entre ambas aplicaciones no se encuentra dentro del alcance de este proyecto, debido a diferencias en ciertas funcionalidades y en la presentación de la información, esto proporciona una referencia comparativa en términos de la experiencia de usuario entre ambos sistemas.

El proyecto mencionado emplea una base de datos relacional, a través de la cual se realizan múltiples consultas para reconstruir el grafo de previas y evaluarlo en capa de aplicación. Esto representa un escenario en el cual, si se implementaran optimizaciones adicionales, mencionadas en las conclusiones VII, se podría lograr una experiencia de usuario aún superior a la que proporciona el producto existente.

La comparación con el sistema de bedelías de la UdelaR resulta aún más compleja. Este sistema divide el acceso a la información en múltiples pantallas, probablemente debido a limitaciones técnicas. Por lo tanto, resulta inviable cuantificar las mejoras en términos de velocidad y facilidad de uso para el estudiante. Además, la página oficial de bedelías contiene información adicional acerca de las asignaturas que no fue requerida para este proyecto. Sin embargo, observamos cómo una base de datos basada en grafos puede ser ventajosa al permitir una implementación de acceso a la información de forma más similar a la utilizada en este proyecto, facilitando a los estudiantes el acceso intuitivo a los datos necesarios para planificar su carrera académica.

VII. Conclusión

El proyecto logra resolver nuestra duda inicial de si es posible construir un sistema de recomendaciones de materias utilizando en una base de datos no relacional basada en grafos. Con resultados muy prometedores, creemos que logra cumplir con el objetivo planteado y abre las puertas para la construcción de un proyecto más complejo, que podría incluso formar parte del

[1]Mi Carrera https://micarrera.uy

[2]repositorio de github https://github.com/cedarcode/mi_carrera

Cuadro I: Duración de consultas a servidor

Consultas realizadas	***Tiempo de respuesta***
subjects.all	$422ms$
subjects.available	$798ms$
subjects.recommended	$388ms$

propio sistema de bedelías. Detectamos algunos puntos de mejora que creemos necesario para que el producto construído pueda alcanzar su máximo potencial:

- Mejora de rendimiento en las consultas
- Generar recomendaciones basadas en parámetros proporcionados por usuarios
- Mejor modelado y utilizacion de previas para cursos y exámenes
- Evaluación de otros motores de bases de datos basadas en grafos
- Filtrado de materias inválidas o no activas

A continuación detallamos cada uno de las oportunidades detectadas.

VII-A. Mejora de rendimiento en las consultas

Como fue mencionado en la sección de **Consultas Relevantes**, la consulta principal (materias habilitadas) hace uso de un algoritmo para la reconstrucción del sistema de previaturas fuera de la base de datos (en el propio servidor web) y otro que, recursivamente, evalúa la estructura con los datos del usuario para verificar si se encuentra habilitado.

Estos algoritmos no fueron implementados con CYPHER puesto que presentaron muchas dificultades. Evaluar recursivamente una estructura probó ser mas complejo de lo estimado inicialmente. Sin embargo, para este tipo de tareas, neo4j soporta el uso de funciones de usuario escritas en Java [4]. Estas funciones o procedimientos se ejecutan directamente en la misma JVM que corre el motor de la base de datos, por lo que se beneficia gratamente de la cercanía a los datos, ya que los tiempos de acceso son menores y se posee control total sobre el recorrido de cualquier estructura.

Si bien evaluamos llevar a cabo este enfoque, consideramos que no es lo suficientemente relevante al curso y por tanto elegimos la implementación en TypeScript.

VII-B. Generar recomendaciones basadas en parámetros proporcionados por usuarios

Actualmente, las recomendaciones generadas por el sistema consisten de evaluar cuantas previas de una materia el usuario ya tiene completadas. Para ello se buscan todas las materias que el usuario tenga al menos una previa, se ordenan por cantidad de previas completadas y se muestran como recomendaciones". Estas recomendaciones, no tienen en cuenta la preferencia del usuario, pues solo se .[ex]tiendeçiegamente a materias que podrían ser completadas con menos esfuerzo. Puesto que Ingeniería en Computación es una carrera en la que los alumnos tienen una gran libertad para elegir materias optativas, muchas veces es más relevante que materia un usuario es afín en lugar de cuál requiere menos esfuerzo para poder cursarla. Por ello, permitiendo al usuario calificar las materias que cursó, se puede obtener una calificación general para los grupos de materias. De esta forma podemos priorizar las recomendaciones en base a la medida que ya tenemos de esfuerzo necesario pero además por la puntuación del grupo a la que cada materia pertenzca.

VII-C. Mejor modelado y utilizacion de previas para cursos y exámenes

Las materias que el usuario puede marcar como completadas en el sistema solo admiten la aprobación de tipo examen. Esto fue una simplificación para reducir la cantidad de casos borde que hay entre los examenes y los cursos. Si bien ambas realidades se encuentran modeladas en la base y pueden ser consultadas independientemente, consideramos oportuno que ambos sistemas de previaturas se representen y sean tratados independientemente. Este problema pudo surgir de la acoplación a la representación que se obtuvo del sistema de previaturas, por lo que un mejor modelado que permita abstraer mejor las nociones de çursoz .[ex]amen"podría simplificar la tarea de consultarlos de manera precisa. Por otra parte, la dependencia sobre cantidad de créditos, ya sea de un grupo en particular o de cantidad absoluta, puede ser abstraída para no utilizar nodos del tipo :Prerequisite, puesto que esta simplicación presentó dificultades a la hora de procesar la estructura de previas de una materia.

VII-D. Evaluación de otros motores de bases de datos basadas en grafos

En un punto anterior, mencionamos que algunas funcionalidades no pudieron ser logradas utilizando el lenguaje de consulta de neo4j (CYPHER). Si bien existe la posibilidad de utilizar Java para lograr una solución eficiente, es posible que otros motores de bases de datos tengan lenguajes de consulta con un conjunto distinto de funcionalidades que permitan realizar mas eficientemente las consultas relevantes a este proyecto. Entre los motores más populares (además de neo4j) tenemos a Amazon Neptune y ArangoDB.

VII-E. Filtrado de materias inválidas o no activas

Un gran problema que observamos al obtener recomendaciones, es que algunas de las materias no se están dictando acutalmente. Por ejemplo, Cálculo 2 es insistentemente recomendada hasta que se completa una materia que la anula, en este caso Cálculo Diferencial e Integral en Varias Variables. Este problema es inherente al de comprobación de la integridad y calidad de los datos utilizados. Si bien ciertas materias no pueden ser omitidas, puesto que se generaría inconsitencia entre las materias que un usuario completó y las que el sistema permite almacenar, se puede afirmar que se necesita una capa más de refinido para lograr recomendaciones enteramnente relevantes.

VII-F. Observaciones finales

La experiencia de modelar y consultar la realidad de previaturas de una carrera probó tener más dificultad de la inicialmente estimada. Sin embargo, el uso de bases de datos basadas en grafos para el modelado de la realidad planteada, fue una elección adecuada. Si bien no se logró cubrir la totalidad de las funcionalidades planteadas, la base de datos en su estado final tiene logra representar la realidad de manera fiel y permite obtener datos relevantes de muchas maneras. El lenguaje de consultas CYPHER provee uan sintáxis que permite expresar relaciones entre datos de manera casi trivial, característica que disfrutamos mucho durante la realización de este proyecto y que esperamos poder utilizar en otros proyectos futuros. Por úlitmo, creemos que el producto generado es capaz que aportar valor a muchos estudiantes para el desarrollo de su carrera.

VIII. ANEXO

Listado 1 Todas las materias junto con estado de aprobado

```
MERGE (u:User { id: \$userId }) WITH u
   MATCH (subject: Subject)
   WHERE subject.name =~ '.*\$input.*'
   WITH subject, exists((u)-[:PASSED]->(subject)) AS passed
   RETURN subject, passed
   ORDER BY passed DESC
```

Listado 2 Materias recomendadas

```
MATCH
   (user :User { id: \$userId })-[:PASSED]->(originator :Subject)
      <-[:NEEDS]-(:Prerequisite)
      <-[:SATISFIES*]-(:Prerequisite)
      <-[:HAS]-(subject :Subject)
   WHERE NOT exists((user)-[:PASSED]->(subject))
   RETURN DISTINCT
      subject,
      count(DISTINCT originator) as priority,
      collect(DISTINCT originator) as originatorList
   ORDER BY priority DESC
```

Listado 3 Grafo de previaturas para evaluación

```
UNWIND \$codes as s
   MATCH
      p = (materia:Subject { code: s })-[:HAS]->
         (:Prerequisite)-[:SATISFIES|CANT*]->
         (:Prerequisite)-[:NEEDS]->(:Subject)
   RETURN materia.code as code, p
```

REFERENCIAS

[1] cedarcode. Mi Carrera., 2019.
[2] Julie Ferrario. Do Graph Databases Scale? Yes? No? Let's see!, 2020.
[3] Guy Harrison. Next Generation Databases. Chapter 5: Tables are Not Your Friends: Graph Databases, 2010.
[4] Neo4J. Neo4j Procedures., 2023.