Trackademic - Sistema de Gestión de Notas Académicas

Integrantes

- Juan David Acevedo
- Jose Manuel Cardona
- Ricardo Andres Chamorro - A00399846
- Oscar Stiven Muñoz - A00399922
- Juan Camilo Muñoz

Contexto del problema

Trackademic es una solución digital que nace de la necesidad académica de cientos de estudiantes universitarios que enfrentan dificultades para organizar y calcular sus notas a lo largo del semestre. La aplicación busca resolver este problema mediante una plataforma intuitiva y colaborativa que permite gestionar planes de evaluación, registrar calificaciones y generar informes de progreso académico con datos confiables.

El sistema fue desarrollado como parte del curso "Sistemas Intensivos en Datos II", utilizando Spring Boot como framework principal, con una arquitectura que combina bases de datos relacionales y NoSQL, en línea con los requerimientos funcionales y no funcionales definidos para el sistema. La elección de Spring Boot se debe a su integración nativa con herramientas modernas de persistencia de datos y su soporte para el desarrollo de aplicaciones web completas, incluyendo controladores, servicios, repositorios y vistas usando Thymeleaf.

Trackademic no solo se enfoca en permitir a los estudiantes realizar seguimiento de sus notas, sino también en fomentar la colaboración entre compañeros mediante comentarios sobre los planes creados, y en proveer herramientas visuales que les permitan comprender su desempeño académico en tiempo real y tomar decisiones fundamentadas.

Requerimientos Funcionales:

RF1. Registro y autenticación de estudiantes
El sistema deberá proporcionar un mecanismo de registro seguro mediante el cual cualquier estudiante pueda crear una cuenta personal, aportando datos básicos (nombre, correo institucional, código estudiantil y contraseña). Una vez registrado, el estudiante deberá autenticarse con correo institucional y contraseña para acceder a todas las funcionalidades

de la aplicación, garantizando la privacidad de su información y la trazabilidad de sus acciones dentro del sistema.

RF2. Consulta de planes de evaluación existentes
Antes de ingresar cualquier nota, el sistema deberá permitir al estudiante consultar planes de evaluación previamente definidos para los grupos en los que está matriculado, mostrando un catálogo navegable con sedes, semestres y grupos disponibles. Cada grupo incluirá información relevante como nombre del curso, código y profesor, obtenida desde la base de datos institucional. Al acceder a la vista de un grupo, se indicará si existe o no un plan de evaluación ya registrado; en caso afirmativo, se mostrará su estructura (actividades y porcentaje de peso) para consulta previa al registro de notas.

RF3. Creación de un plan de evaluación
Si un grupo no tiene aún un plan de evaluación definido, o el estudiante prefiere elaborar uno propio, el sistema le permitirá crear un nuevo plan. Para ello deberá ingresar cada actividad (nombre de la evaluación y porcentaje de peso) de forma iterativa, hasta completar la definición total del 100% del curso, quedando listo para la carga de notas. Luego, de su creación, este se publica para que cualquier estudiante lo vea.

RF4. Validación automática de porcentajes
Cada vez que el estudiante agregue, modifique o elimine una actividad de un plan de evaluación, el sistema realizará de forma inmediata el cómputo de la suma de todos los porcentajes y solamente permitirá guardar cambios si ésta resulta exactamente igual al 100 %. En caso contrario, desplegará un mensaje de error aclarando el exceso o déficit acumulado, impidiendo la persistencia de un plan inválido.

RF5. Modificación de planes de evaluación propios
El estudiante podrá editar actividades de cualquier plan de evaluación que él mismo haya creado. No podrá modificar el de los demás estudiantes. Las modificaciones incluirán cambio de nombre de la actividad o ajuste del porcentaje. Tras cada cambio el sistema validará nuevamente el total de porcentajes y requerirá corrección antes de permitir la actualización definitiva en la base de datos.

RF6. Ingreso, modificación y eliminación de notas individuales
Una vez seleccionado o creado su plan de evaluación, el estudiante podrá utilizarlo como una plantilla para poder ingresar las calificaciones obtenidas en cada actividad. Adicionalmente, tendrá la libertad de modificar o eliminar cualquiera de las notas previamente registradas por él, de modo que refleje siempre su estado real de progreso académico. Además, cuando el estudiante elimine una nota cambiará a estado “nota eliminada” para que le de cuenta al mismo que fue eliminada y pueda cambiarlo a su disposición.

RF7. Visualización del consolidado de notas por semestre
El sistema generará un informe en pantalla donde se consolide, actividad por actividad y curso por curso, la totalidad de las calificaciones del estudiante en un semestre académico.

Este reporte detallado incluirá (a) nombre del grupo, (b) profesor, (c) actividades y porcentaje de cada una, y (d) nota obtenida y contribución al promedio final. De esta forma el usuario podrá entender su desempeño global en ese periodo.

RF8. Informe de progreso por asignatura
Complementariamente, se ofrecerá un segundo informe tipo “progreso” para cada curso: mostrará la nota acumulada hasta el momento versus la nota necesaria para alcanzar el umbral de aprobación (p. ej. 3.0/5.0 o 70 %). Este informe calculará automáticamente, según el porcentaje restante, el puntaje mínimo requerido en las actividades pendientes, apoyando la toma de decisiones del estudiante.

RF9. Comentarios colaborativos sobre planes de evaluación
El sistema deberá habilitar un espacio de comentarios asociado a cada plan de evaluación, permitiendo que los estudiantes hagan aportes, sugerencias o preguntas de forma colaborativa. Cada comentario incluirá autor y fecha, y no afectará la estructura del plan hasta que el propietario decida incorporar las recomendaciones.

Requerimientos No Funcionales:

RNF1. El sistema debe estar implementado usando Spring Boot y Spring MVC, con vistas Thymeleaf.

RNF2. La base de datos relacional debe estar implementado en PostgresSQL y contener las entidades proporcionadas por la universidad

RNF3. El sistema debe utilizar MongoDB para manejar planes de evaluación, notas, de estudiantes y comentarios, por su flexibilidad estructurada.

RNF4. El sistema debe seguir el principio de separación de capas: controlador, servicio, repositorio y entidad.

RNF5. La validación de que un plan tenga el 100% del total en pesos debe realizarse tanto a nivel de backend como de frontend

RNF6. La aplicación debe ofrecer una interfaz clara, intuitiva y responsive para el uso por parte de estudiantes.

RNF7. La solución debe poder desplegarse en servicios en la nube como Railway (PostgresSQL) y Railway (MongoDB).

Sustentación de la selección de la base de datos

Características de los datos y decisión arquitectónica

El sistema maneja dos tipos de datos con características muy diferentes:

Datos estructurados y transaccionales, tales como información de empleados, facultades, grupos y cursos, que poseen relaciones bien definidas y requieren integridad referencial estricta. Estos datos son generalmente estáticos o administrados por la universidad, y su estructura se mantiene relativamente estable en el tiempo.

Datos semi-estructurados y flexibles, como los planes de evaluación, las notas académicas ingresadas por estudiantes y los comentarios colaborativos. Estos datos cambian constantemente, pueden contener estructuras anidadas (listas de actividades, observaciones, estados), y su esquema puede evolucionar según las necesidades de los usuarios.

Justificación de la solución políglota

Para satisfacer estos requerimientos contrastantes, se optó por una arquitectura **políglota**, que permite utilizar diferentes tecnologías de almacenamiento según el tipo y comportamiento de los datos.

PostgreSQL fue elegido para manejar los datos estructurados. Este sistema gestor de bases de datos relacional ofrece un sólido soporte para integridad referencial, transacciones ACID, y un modelo tabular que se alinea perfectamente con los esquemas de datos institucionales. Además, su compatibilidad con Spring Data JPA facilita la integración con el backend desarrollado en Spring Boot, permitiendo realizar operaciones complejas mediante repositorios declarativos y consultas SQL personalizadas.

MongoDB, por otro lado, se seleccionó para gestionar los datos con estructura dinámica. Su modelo basado en documentos permite representar de forma natural los planes de evaluación como objetos que incluyen actividades con pesos variables, notas registradas, y comentarios anidados. Esta flexibilidad elimina la necesidad de definir previamente una estructura rígida en la base de datos y permite iterar rápidamente durante el desarrollo. Asimismo, MongoDB ofrece buen rendimiento en operaciones de escritura frecuentes, algo crucial en un entorno donde los estudiantes modifican planes y calificaciones constantemente.

A diferencia de otras bases de datos NoSQL, como las orientadas a columnas (entre las cuales se encuentra Apache Cassandra), MongoDB no obliga a diseñar el modelo de datos a partir de patrones de consulta rígidos. En soluciones como Cassandra, cada tipo de consulta requiere una tabla diseñada específicamente para esa lectura, lo que conduce a la duplicación de datos y a una complejidad innecesaria. Si bien Cassandra ofrece grandes ventajas en escenarios de alta escalabilidad y rendimiento distribuido, como sistemas de monitoreo en tiempo real o registro de eventos a gran escala, dichas características no son prioritarias en el presente proyecto, cuyo foco está más centrado en la estructuración y análisis dinámico de datos académicos personales.

Ventajas de esta arquitectura

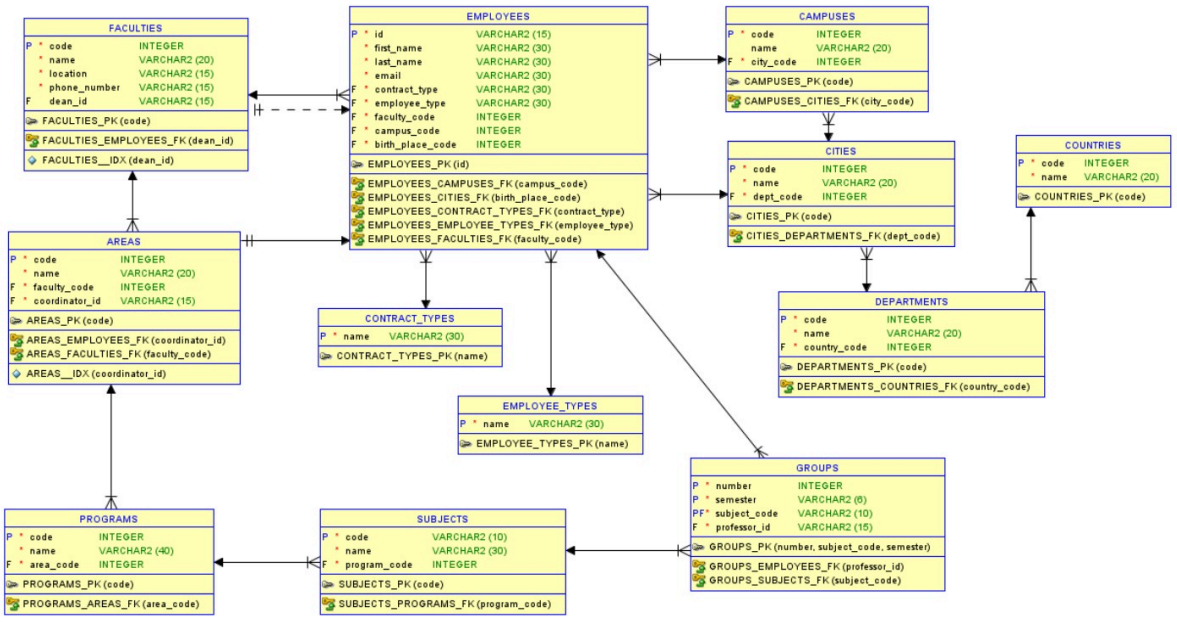
- Permite aprovechar lo mejor de ambos mundos: integridad y robustez de datos con PostgreSQL, y flexibilidad y escalabilidad con MongoDB.
- Facilita el mantenimiento y escalamiento independiente de cada subsistema.
- La separación lógica mejora la modularidad y la claridad del diseño.
- Se ajusta a los requerimientos tanto funcionales como no funcionales definidos por el cliente.

Tecnologías Utilizadas

1. Backend: Spring Boot 3.4.5
2. Bases de Datos: PostgreSQL y MongoDB (hosting Railway)
3. Herramientas: Lombok, Spring Data JPA, Spring Data MongoDB
4. Frontend: Thymeleaf para vistas MVC, garantizando una interfaz web intuitiva y responsive.
5. Despliegue: Bases de datos alojadas en Railway (PostgreSQL y MongoDB).

Modelado de Datos

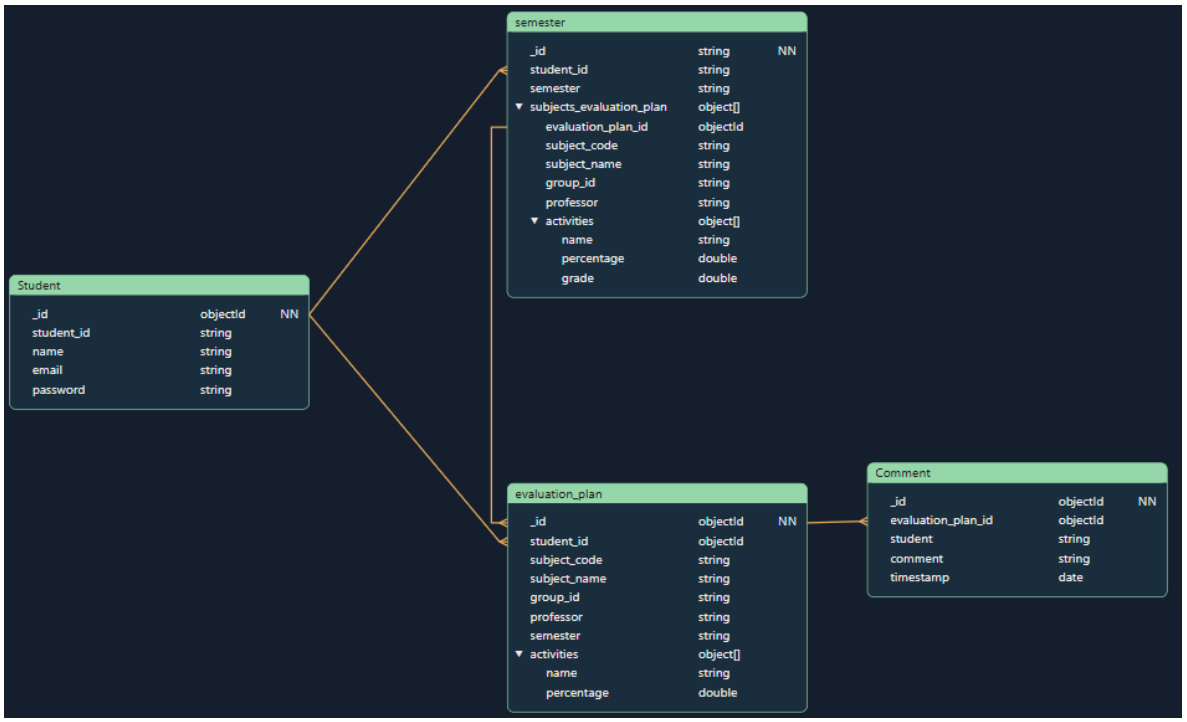
Modelo Relacional (PostgreSQL)



El modelo relacional dado inicialmente, la universidad ya tiene una estructura de datos concreta y estable en PostgreSQL ya ofrece una estructura sólida y coherente con su organización académica. Este esquema refleja adecuadamente la distribución institucional de sedes, facultades, programas, materias y grupos de clase, con relaciones bien definidas que garantizan la consistencia de los datos. La arquitectura propuesta por Trackademic puede acoplarse de forma directa a este modelo sin necesidad de realizar modificaciones estructurales, dado que las entidades cubren completamente la lógica académica actual. Esto asegura una integración estable y eficiente, salvo en el caso de transformaciones organizacionales significativas a futuro.

Este modelo está altamente normalizado, lo que permite garantizar integridad referencial y evitar redundancias. Está optimizado para realizar consultas relacionales complejas y es ideal para integrar la información académica estructurada del sistema institucional. Lo cual para el resto de entidades a usar se usará en MongoDB sin una modificación previa del modelo relacional.

Modelo de Documentos (MongoDB)



El diseño del modelo NoSQL para esta aplicación se fundamenta en la flexibilidad y escalabilidad que ofrece MongoDB, permitiendo un crecimiento y adaptación sencilla a futuros cambios en los requerimientos sin las limitaciones de un esquema rígido tradicional. Se optó por una separación clara de responsabilidades mediante documentos independientes que representan entidades específicas: el documento Student contiene la información básica e inmutable de cada estudiante, como su identificación, nombre y correo electrónico, evitando la duplicación o mezcla con datos académicos variables.

Por otro lado, el documento evaluation_plan representa las plantillas de planes de evaluación que los estudiantes pueden crear y compartir. Este diseño mantiene organizadas las actividades, asignaturas y porcentajes como una estructura reutilizable y consistente. El documento semester se encarga de almacenar el uso personalizado de una plantilla de evaluación en un semestre concreto, incluyendo las notas obtenidas por actividades, permitiendo que cada estudiante pueda tener un registro independiente sin afectar la plantilla original. Los comentarios se manejan mediante un documento separado, Comment, que se vincula a las plantillas, facilitando la gestión y visualización de retroalimentación sin interferir con los datos de notas o semestres específicos.

La ausencia de atributos como carrera o semestre fijo dentro del documento Student responde a la necesidad de que los usuarios puedan acceder, consultar y utilizar cualquier plan de evaluación sin restricciones, atendiendo escenarios como doble carrera, materias complementarias o repetición de semestres. Esto hace que el sistema sea mucho más dinámico y adaptable a distintos perfiles académicos. Asimismo, la referencia explícita desde los documentos de semestre hacia las plantillas originales evita la duplicación de planes y garantiza la integridad de la información base, mientras que el embebido de las actividades y calificaciones dentro de semestre y evaluación mejora la eficiencia en las consultas, reduciendo la necesidad de múltiples accesos a la base de datos.

Finalmente, la decisión de mantener los comentarios desacoplados y vinculados solo a las plantillas originales asegura una modularidad que facilita su administración independiente y la disponibilidad general de retroalimentación para todas las instancias que utilicen esa plantilla. Este enfoque integral permite que el modelo evolucione fácilmente con futuras mejoras o ampliaciones sin requerir migraciones complejas o reestructuraciones profundas, aprovechando así al máximo las fortalezas del modelo document-based de MongoDB para ofrecer un sistema robusto, flexible y mantenible.

Conexión Externa a las Bases de Datos

1. **MongoDB (usando mongosh):**

```
mongosh
'mongodb://mongo:IwJvnHxCPXRMXumoDbkEAeRhBjyyUStF@mainline.proxy.r
lwy.net:52476/trackademic?authSource=admin'
use trackademic
```

2. **PostgreSQL (usando psql):**

```
psql -h caboose.proxy.rlwy.net -p 45135 -U postgres -d railway
```

Cuando solicite la contraseña, ingrese:

```
cQNaFJywkdHPzUFmlfauUSQRSXSiAphl
```

Ejecución de la Aplicación

1. **Clonar el repositorio:**

git clone https://github.com/tu-usuario/Trackademic.git

2. **Entrar a la carpeta del proyecto**

```
cd Trackademic
```

3. **Instalar dependencias:**

- Opción 1: Usando el wrapper de Maven

```
./mvnw clean install
```

- Opción 2: Con Maven instalado globalmente

```
mvn clean install
```

Asegúrate de tener Java 17 o superior instalado.

4. **Ejecutar con Maven:**

- Opción 1: Usando el wrapper de Maven

```
./mvnw spring-boot:run
```

- Opción 2: Con Maven instalado globalmente

```
mvn spring-boot:run
```

5. **Acceder a la aplicación: Abre tu navegador y dirígete a la siguiente URL:**

http://localhost:8080/Trackademic/

Puedes registrarte o usar este usuario:
email: alice.s@trackademic.edu.co
contraseña: pass123

Repositorio público del proyecto

Toda la documentación técnica, el código fuente y las instrucciones de despliegue están disponibles públicamente en el siguiente repositorio:

https://github.com/JuanCamiloMunozB/Trackademic