Universidad Santiago de Cali

Programación Orientada a la Web, 2025B
Desarrollo de Backend: Modelo de Datos

Nombre del proyecto:
LearnUp

Descripción:
LearnUp es una plataforma web educativa enfocada en mejorar la comunicación y el aprendizaje colaborativo entre estudiantes y monitores. Surge como una solución a la falta de espacios digitales donde los estudiantes puedan resolver dudas, compartir conocimientos y mantenerse conectados con su entorno académico de forma práctica y accesible. Su propósito principal es facilitar el intercambio de apoyo entre quienes enseñan y quienes aprenden, fortaleciendo la experiencia educativa dentro y fuera del aula.

Nombres de los integrantes:
Esteban Marta Rojas

Erik Camilo Dussan Velasco

Johan Esneider Lucumi Palacios

ESTRUCTURA DEL DOCUMENTO:

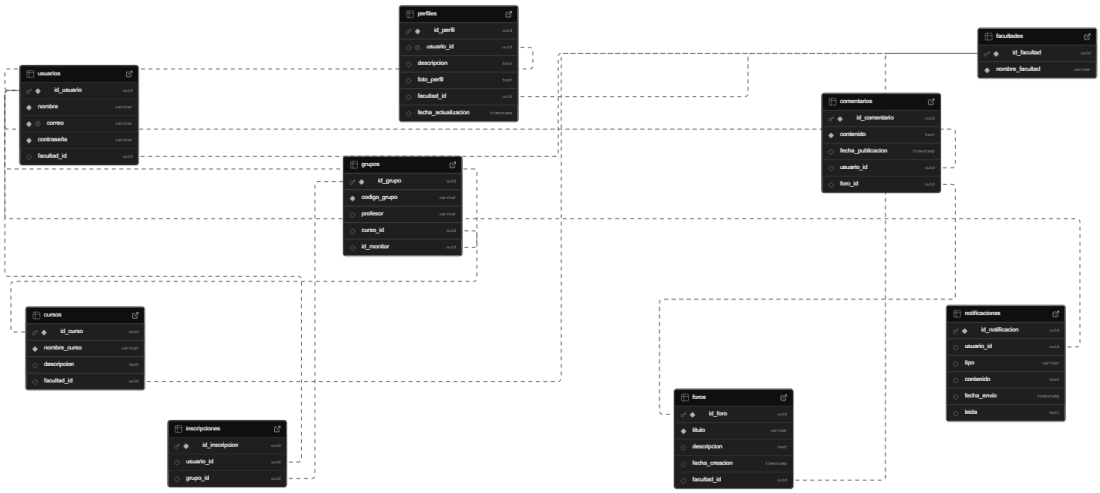
1. Objetivos del documento
El objetivo de este documento es describir el diseño del modelo de datos implementado en el proyecto LearnUp, una plataforma web desarrollada con NestJS que integra una base de datos relacional (Supabase/PostgreSQL) y una documental (MongoDB). El modelo busca garantizar una estructura coherente, eficiente y adaptable para las principales funcionalidades del sistema: gestión de usuarios, foros académicos y comunicación en tiempo real mediante chat.

2. Alcance
El documento cubre la descripción técnica y conceptual de ambos modelos de datos, explicando su propósito, estructura, relaciones y cómo se integran en la arquitectura general del proyecto. Cada modelo cubre distintas necesidades:

- **Supabase:** manejo de datos estructurados, autenticación, relaciones entre usuarios, cursos y foros.
- **MongoDB:** manejo de datos dinámicos y no estructurados, como mensajes de chat y notificaciones en tiempo real.

3. Modelo de datos relacional:

❖ **Diagrama Entidad-Relación (ERD):**



❖ **Descripción de las Entidades:**

- **FACULTADES:** Contiene los nombres de las facultades de la Universidad Santiago de Cali. Es el punto de entrada para navegar por cursos, grupos y foros.
- **USUARIOS:** Registra los estudiantes que acceden a la plataforma. Incluye datos de login y la facultad a la que pertenecen.
- **PERFILES:** Almacena la información visible del usuario: foto, descripción, facultad y última actualización. Tiene relación 1:1 con USUARIOS.
- **CURSOS:** Representa las asignaturas disponibles dentro de cada facultad. Cada curso pertenece a una facultad.
- **GRUPOS:** Define los grupos de clase asociados a un curso. Contiene el código del grupo, el profesor responsable y un monitor (que también es un usuario).
- **INSCRIPCIONES:** Relación muchos a muchos entre USUARIOS y GRUPOS. Indica qué estudiantes pertenecen a qué grupo.
- **FOROS:** Son espacios de discusión asociados a cada facultad.
- **COMENTARIOS:** Almacenan los mensajes o respuestas de los usuarios dentro de los foros.
- **NOTIFICACIONES:** Guarda alertas enviadas a los usuarios (por ejemplo, nuevas respuestas o anuncios).

❖ **Relaciones y Cardinalidades:**

Relación	Tipo	Cardinalidad	Descripción
Facultad → Usuarios	Relacional	1:N	Una facultad puede tener muchos usuarios.
Usuario → Perfil	Relacional	1:1	Cada usuario tiene un único perfil.
Facultad → Cursos	Relacional	1:N	Una facultad puede ofrecer varios cursos.
Curso → Grupos	Relacional	1:N	Un curso puede tener varios grupos.
Usuario → Grupos (como monitor)	Relacional	1:N	Un usuario puede ser monitor de varios grupos.
Grupo → Inscripciones	Relacional	1:N	Un grupo puede tener múltiples estudiantes inscritos.
Usuario → Inscripciones	Relacional	1:N	Un usuario puede estar en varios grupos.
Facultad → Foros	Relacional	1:N	Cada facultad tiene un foro general.
Foro → Comentarios	Relacional	1:N	Cada foro puede tener múltiples comentarios.
Usuario → Comentarios	Relacional	1:N	Un usuario puede escribir varios comentarios.
Usuario → Notificaciones	Relacional	1:N	Un usuario puede recibir múltiples notificaciones.

❖ **Justificación de Normalización y Motor de Base de Datos**:

- Se aplicaron las tres primeras formas normales (1FN, 2FN, 3FN):
    - Cada atributo contiene solo un valor atómico (1FN).
    - Todas las columnas dependen de la clave primaria (2FN).
    - No existen dependencias transitivas (3FN).
- Este diseño asegura consistencia, integridad referencial y menor redundancia.

Elección del motor: Supabase (PostgreSQL)

- PostgreSQL es ideal para relaciones estructuradas y manejo de múltiples entidades interconectadas.
- Supabase facilita autenticación, relaciones y APIs automáticas.
- Soporta fácilmente integraciones con NestJS y Node.js (backend actual del proyecto).

❖ **Requerimientos Funcionales:**

- **Registro e inicio de sesión**
  - El sistema permitirá que los estudiantes se registren proporcionando su nombre, correo y contraseña.
  - Los usuarios podrán iniciar sesión para acceder a las funcionalidades de la plataforma.
  - Esta información se almacena en la tabla USUARIOS.

- **Gestión y visualización de perfil**
  - Cada usuario tendrá un perfil personal con datos como nombre, descripción, facultad y foto de perfil.
  - Podrá editar su información básica y visualizar el perfil de otros usuarios.
  - Se utiliza la tabla PERFILES, relacionada con USUARIOS y FACULTADES.

- **Navegación por facultades**
  - En la página principal se mostrarán todas las facultades disponibles.
  - Cada usuario solo podrá acceder a los cursos de la facultad a la que pertenece.
  - Involucra las tablas FACULTADES y USUARIOS.

- **Listado y visualización de cursos**
  - El sistema mostrará los cursos disponibles dentro de cada facultad, junto con su nombre y descripción.
  - Los usuarios podrán explorar los cursos antes de unirse a un grupo.
  - Se utiliza la tabla CURSOS, relacionada con FACULTADES.

- **Gestión de grupos y monitores**
  - Cada curso puede tener varios grupos, los cuales están dirigidos por diferentes profesores o monitores.
  - Los usuarios podrán consultar el grupo correspondiente a su código y ver la información del profesor o monitor.
  - Las tablas involucradas son GRUPOS, USUARIOS y CURSOS.

- **Inscripción a grupos**
  - Los estudiantes podrán inscribirse al grupo que corresponda a su curso (por ejemplo, *BWM-04*).
  - Cada estudiante solo podrá estar inscrito en un grupo por curso.
  - La información se guarda en la tabla INSCRIPCIONES, relacionada con USUARIOS y GRUPOS.

- **Participación en foros**
  - Cada facultad contará con un foro general donde los usuarios podrán compartir ideas, recursos y enlaces.
  - Los estudiantes podrán crear publicaciones o temas de discusión.

- Utiliza las tablas FOROS, FACULTADES y USUARIOS.

- **Comentarios en foros**
  - Los usuarios podrán responder a publicaciones dentro del foro, comentar, editar o eliminar sus mensajes.
  - Esta información se maneja con las tablas COMENTARIOS, FOROS y USUARIOS.

- **Notificaciones del sistema**
  - Los usuarios recibirán notificaciones cuando alguien responda a sus comentarios o cuando haya nuevos mensajes o anuncios.
  - Se gestiona con la tabla NOTIFICACIONES, relacionada con USUARIOS.

- **Chat en tiempo real por grupo**
  - Cada grupo contará con un chat en tiempo real para comunicación directa entre estudiantes, monitores y profesores.
  - Los mensajes se almacenarán en la base de datos MongoDB, dentro de la colección chats.

4. Modelo de datos documental:

❖ **Estructura de los documentos y colecciones (JSON):**

```
{
  "_id": ObjectId("671123abc..."),
  "grupo_id": "uuid-del-grupo",
  "nombre_grupo": "BWM-04",
  "mensajes": [
    {
      "mensaje_id": "uuid-mensaje",
      "usuario_id": "uuid-del-usuario",
      "nombre_usuario": "Johan Lucumi",
      "contenido": "Hola, alguien tiene el taller 3?",
      "fecha_envio": "2025-10-11T14:23:00Z",
      "tipo": "texto",
      "visto": false
    }
  ]
}
```

❖ **Descripción de las relaciones entre documentos:**

- Cada documento en la colección chats está asociado con un grupo específico de la base de datos relacional (GRUPOS en Supabase).

- El campo grupo_id funciona como una referencia externa (foreign key lógica) hacia el grupo correspondiente en la base relacional.
- Dentro del arreglo mensajes, cada mensaje contiene el usuario_id, que referencia a un usuario en la tabla USUARIOS del sistema relacional.
- Aunque no existe una relación estricta entre colecciones (como en SQL), MongoDB permite esta vinculación lógica y flexible, ideal para manejar estructuras jerárquicas y en tiempo real.

❖ **Justificación de la elección del motor de base de datos documental:**

Se seleccionó MongoDB como base de datos documental por las siguientes razones:

- **Flexibilidad estructural:** los documentos en formato JSON permiten almacenar mensajes con diferentes atributos sin necesidad de modificar esquemas.
- **Escalabilidad:** MongoDB maneja grandes volúmenes de datos y conexiones simultáneas, ideal para chats en tiempo real con múltiples usuarios.
- **Rendimiento en lectura y escritura:** su diseño basado en documentos optimiza las consultas rápidas y la inserción continua de nuevos mensajes.
- **Compatibilidad con NestJS:** el framework utilizado en el backend se integra fácilmente mediante Mongoose, simplificando la gestión de modelos y esquemas.
- **Naturaleza temporal de los datos:** los mensajes y notificaciones son datos efímeros que cambian constantemente, por lo que no requieren la rigidez de una base relacional.

❖ **Requerimientos funcionales:**

El modelo documental con MongoDB cubre los siguientes requerimientos del sistema:

- **Chat en tiempo real por grupo:** permite enviar y recibir mensajes instantáneamente entre usuarios, profesores y monitores.
- **Historial de conversaciones:** guarda los mensajes dentro de cada grupo de forma estructurada y accesible.
- **Gestión de estados del mensaje:** seguimiento de propiedades como *visto*, *fecha de envío* y *tipo de mensaje* (texto, archivo, imagen, etc.).
- **Escalabilidad horizontal:** permite que múltiples grupos funcionen de forma independiente sin afectar el rendimiento global del sistema.

5. Conclusiones:

La combinación de una base de datos relacional (Supabase/PostgreSQL) y una no relacional (MongoDB) ofrece un enfoque equilibrado y eficiente para el manejo de la información en LearnUp. PostgreSQL brinda la estructura, seguridad e integridad necesarias para gestionar datos críticos como usuarios, cursos, grupos y foros, garantizando la consistencia de la información. Por su parte, MongoDB permite manejar datos dinámicos, como los mensajes del chat en tiempo real, de forma flexible y escalable, sin afectar el rendimiento del sistema principal.

Entre las **ventajas** más importantes se destacan:

- La capacidad de aprovechar las fortalezas de cada tecnología para diferentes tipos de datos.
- La escalabilidad horizontal del chat gracias a MongoDB.
- La seguridad y consistencia en los registros académicos proporcionadas por Supabase.

Sin embargo, existen desafíos, como la necesidad de mantener sincronización entre ambas bases de datos y gestionar múltiples conexiones desde el backend. A futuro, el proyecto puede ampliarse integrando notificaciones en tiempo real, almacenamiento de archivos multimedia, y un sistema de borrado lógico de datos, además de optimizar la arquitectura hacia un entorno de microservicios, donde cada componente (chat, usuarios, cursos) se gestione de forma independiente.

6. Bibliografía:

- NestJS Documentation: https://docs.nestjs.com/
- Supabase Documentation: https://supabase.com/docs/
- PostgreSQL Documentation: https://www.postgresql.org/docs/
- MongoDB Documentation: https://www.mongodb.com/docs/
- Mongoose ODM Guide: https://mongoosejs.com/docs/
- QuickDBD Database Diagram Tool: https://www.quickdatabasediagrams.com/