Leaftime

MODELADO DE DATOS.

MODELADO DE DATOS

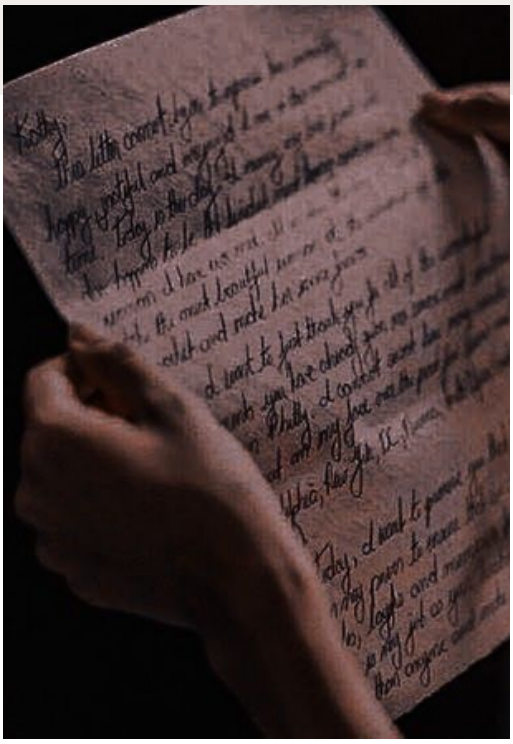
Y ARQUITECTURA DE LA BASE DE DATOS (MONGODB)

Leaftime

A diferencia de las bases de datos SQL (como MySQL o PostgreSQL) que usan tablas y filas, MongoDB usa Colecciones y Documentos (en formato JSON/BSON). Esto tiene implicaciones directas en cómo estructuramos y relacionamos nuestros datos. En esta actividad, definirán los "Schemas" (la estructura) de sus datos y analizarán las implicaciones de usar un modelo NoSQL.

Parte A: DEFINICION DE COLECCIONES Y SCHEMAS

POST **LIBROS:**



```
{
  "_id": "69347ee0d795187af8e08b46",
  "title": "String",
  "author": "String",
  "pages": 0,
  "published": "1867-02-13",
  "status": {
  type: String,
  enum: ["Leyendo", "Terminado", "Abandonado",
"Empezado", "Pendiente"],
  default: "Pendiente"
},
  "genre": {
  type: String,
  enum: ["Fantasía", "Terror", "Romance",
"Aventura", "Suspenso", "Ciencia ficción"]
},

  "createdAt": "2025-12-06T19:07:12.089Z",
  "updatedAt": "2025-12-06T19:07:12.089Z"
}
```

PARTE B: Discusión de Implicaciones NoSQL

La flexibilidad de MongoDB es una ventaja para nuestro proyecto porque nos permite agregar o cambiar campos del libro fácilmente y desarrollar más rápido sin preocuparnos por modificar un esquema rígido.

Sin embargo, también significa que debemos usar Mongoose para mantener consistencia y evitar errores, ya que sin restricciones podríamos terminar con datos incompletos o inconsistentes. MongoDB es ideal para una aplicación que crecerá y cambiará en el tiempo, pero requiere disciplina para mantener el orden.

Relaciones (Embedding vs. Referencing)

Tipo de relación	Se usa para...	Ejemplo en nuestra app	Razón
Referencias	Datos que se usan en varias colecciones y pueden cambiar	Autor del libro (authorId)	Si el autor cambia su nombre, no queremos actualizar
Embedding	Datos pequeños y propios del documento	Notas o tags dentro del libro	Siempre se leen junto con el libro y no necesitan su

Denormalización

(Duplicación de Datos)

En bases NoSQL como MongoDB, a veces se duplican datos a propósito para evitar hacer consultas adicionales y así acelerar las lecturas, aunque eso implique más trabajo cuando los datos cambian.
En nuestra aplicación MERN de gestión de libros, sí hay lugares donde la denormalización sería útil.

PUNTOS CLAVE.

En nuestra aplicación, la denormalización es útil principalmente en:

1. *Guardar el nombre del autor dentro del libro, además del authorId, para cargar las listas más rápido.*
2. *Duplicar el nombre del género si existe una colección aparte, evitando consultas adicionales.*
3. *Guardar el progreso actual del libro, además del historial de progreso, para leer datos más rápido.*

Aunque duplicar información requiere más trabajo al actualizar, mejora considerablemente el rendimiento al mostrar listas de libros y evitar consultas extra a otras colecciones.

INTEGRANTES:

ITZAMARA SOTERO MARTINEZ

SANJUANA GARCÍA HERNÁNDEZ

YHAIRI XIMENA RIVERA REYES