Softbinator
TECHNOLOGIES

Programarea aplicatiilor in NodeJS

Curs 7

Introducere in Baze de Date

Ce este o baza de date?

O colectie de date structurata si stocata electronic.

Tipuri baze de date:

1. SQL: PostgreSQL, MySQL, SQLite
2. NoSQL: MongoDB, Cassandra

SQL vs NoSQL

SQL	NoSQL
Structured Data	Unstructured/Semi-structured Data
ACID compliant	BASE compliant
Relationships	Document databases (Mongo DB) Key-value stores (Redis) Graph databases
Vertical Scalability	Horizontal Scalability

SQL vs NoSQL

1. SQL:
   a. Query-uri mai complexe
   b. Aplicatii tranzactionale
2. NoSQL:
   a. Aplicatii scalabile
   b. Volum mare de date

ORM

ORM (Object Relational Mapping) este o tehnica prin care putem face queries sau mutation asupra unei baza de date folosind paradigma orientata pe obiecte.

De cele mai multe ori, prin ORM ne referim la o librarie scrisa in limbajul de programare pe care il folosim pe server (JavaScript, Java, .NET, Rust, C, C++, etc.) care usureaza modul in care scriem queries.

Printr-o librarie ORM reducem boilerplate code, codul devine mai usor de inteles si mai usor de facut debug si abstractizeaza layer-ul de access catre baza de date, reducand erorile de SQL care ar fi putut aparea in cazul in care am fi scris query-uri “raw”.

Exemple de ORM-uri pt JS

- Sequelize (Postgres, MySQL, MariaDB, SQLite)
- Mongoose (MongoDB)
- Typegoose (Mongoose cu typescript)
- TypeORM (MySQL, MariaDB, Postgres, CockroachDB, SQLite, etc.)
- **Prisma (PostgreSQL, MySQL, SQLite, SQL Server)**
- Drizzle (Ce are Prisma + altele)

Prisma

- La laborator vom folosi Prisma, dar daca vi se pare altceva mai interesant, va incurajam sa incercati, intrucat fiecare ORM are pros and cons.
- Prisma are 3 componente principale:
  - Prisma client: O librarie care ofera “type-safe” access catre baza de date
  - Prisma migrate: un tool de migrare care genereaza automat fisiere de migrare in momentul in care se produc modificari asupra schemei
  - Prisma studio: un UI pentru vizualizarea datelor din baza de date

Prisma

Cum functioneaza?
Dupa ce instalam toate dependintele necesare pentru a rula Prisma, trebuie sa cream un fisier numit “schema.prisma”. Pe baza acestui fisier, tool-ul de migrari oferit de Prisma va genera automat fisiere de migrare pe care mai apoi le putem aplica peste baza de date.

In acelasi timp, daca folosim TypeScript, ne putem folosi de tipurile de date generate de Prisma care sunt foarte folositoare.

prisma/schema.prisma

```
model User {
  id    Int     @id @default(autoincrement())
  email String  @unique
  name  String?
  posts Post[]
}

model Post {
  id        Int     @id @default(autoincrement())
  title     String
  content   String?
  published Boolean @default(false)
  author    User    @relation(fields: [authorId], references: [id])
  authorId  Int
}
```

Prisma Query Engine

In spate, Prisma foloseste un Query Engine, care transforma instructiunile noastre de tipul “prisma.users.findMany()” in instructiuni de tipul “SELECT * FROM users” care pot fi rulate pe baza de date.

Acest Query Engine este o sabie cu doua taisuri. In 99% din cazuri este o binecuvantare, deoarece usureaza foarte mult modul in care scriem queries, suntem mult mai rapizi, e mai usor de inteles, etc. Dar in 1% din cazuri, in cazurile in care avem query-uri foarte FOARTE complicate, query engine devine incet, si poate sa scrie query-uri ineficiente, caz in care e recomandat sa scriem “raw queries”.

Raw queries

Cam fiecare ORM matur ofera posibilitatea de a scrie si executa raw queries.

Pros:

1. Avem control absolut asupra query-ului
2. E puternic si precis
3. Il putem optimiza mult mai mult

Cons:

1. Mult mai deschis la erori, e recomandat sa il testati foarte bine
2. Risc de SQL injection attacks