Database Integration

Terms

Databases
Database Engines
Migrations
Models
Object-relational Mapper (ORM)
Prisma

Summary

- We use *databases* to permanently store data. There are many *database engines* available. Some of the popular ones are MySQL, PostgreSQL, MongoDB, etc.

- To connect our applications to a database, we often use an *Object-relational Mapper* (ORM). An ORM is a tool that sits between a database and an application. It's responsible for mapping database records to objects in an application. Prisma is the most widely-used ORM for Next.js (or Node.js) applications.

- To use Prisma, first we have to define our data *models*. These are entities that represent our application domain, such as User, Order, Customer, etc. Each model has one or more fields (or properties).

- Once we create a model, we use Prisma CLI to create a *migration* file. A migration file contains instructions to generate or update database tables to match our models. These instructions are in SQL language, which is the language database engines understand.

- To connect with a database, we create an instance of PrismaClient. This client object gets automatically generated whenever we create a new migration. It exposes properties that represent our models (eg user).

Key Commands

```
# Setting up Prisma
npx prisma init

# Formatting Prisma schema file
npx prisma format

# Creating and running a migration
npx prisma migrate dev
```

Working with Prisma Client

```
await prisma.user.findMany();
await prisma.user.findUnique({ where: { email: 'a' } });
await prisma.user.create({ data: { name: 'a', email: 'a' } });
await prisma.user.update({ where: { email: 'a' }, data: { email: 'b' } });
```

