API REST avec Fastify + Prisma

Sommaire

Partie 1

Objectif :

- Installer le projet
- Configurer Fastify
- Configurer Prisma
- Créer un modèle
- Exposer des routes CRUD simples

---

1. Prérequis

- Node.js >= 18
- PostgreSQL installé

---

2. Initialisation du projet

```
mkdir fastify-prisma-api
cd fastify-prisma-api
npm init -y
```

Installation dépendances :

```
npm install fastify @prisma/client
npm install -D prisma typescript ts-node-dev @types/node
```

Initialiser Prisma :

```
npx prisma init
```

---

3. Configuration TypeScript

Créer `tsconfig.json` :

```
{
  "compilerOptions": {
    "target": "ES2020",
    "module": "CommonJS",
    "rootDir": "./src",
    "outDir": "./dist",
    "strict": true,
    "esModuleInterop": true
  }
}
```

Ajouter dans `package.json` :

```
"scripts": {
  "dev": "ts-node-dev --respawn src/server.ts"
}
```

---

2. Configuration Prisma avec SQLite

Modifier `prisma/schema.prisma` :

```
datasource db {
  provider = "sqlite"
  url      = env("DATABASE_URL")
}

generator client {
  provider = "prisma-client-js"
}
```

---

3. Configuration du fichier .env

```
DATABASE_URL="file:./dev.db"
```

Explication :

- Prisma va créer automatiquement un fichier `dev.db`
- Le fichier sera situé à la racine du projet

4. Lancer la migration

```
npx prisma migrate dev --name init
```

Résultat :

- Fichier dev.db créé
- Tables générées
- Client Prisma généré

Aucune base externe nécessaire.

4.a Structure projet avec SQLite

```
fastify-prisma-api/
 ├── prisma/
 ├── src/
 ├── dev.db
 ├── .env
 ├── package.json
```

4.b Configuration Prisma avec postgres

```
npm install @prisma/adapter-pg pg
npm install -D @types/pg
```

Fichier `.env` :

```
DATABASE_URL="postgresql://user:password@localhost:5432/fastifydb"
```

Modifier `prisma/schema.prisma` :

```
datasource db {
  provider = "postgresql"
  url      = env("DATABASE_URL")
}

generator client {
  provider = "prisma-client-js"
```

```
}

model Post {
  id        Int      @id @default(autoincrement())
  title     String
  content   String?
  createdAt DateTime @default(now())
}
```

Créer la base :

```
npx prisma migrate dev --name init
```

---

5. Structure simple du projet

```
src/
 ├── server.ts
 └── prisma.ts
```

---

6. Prisma Client

`src/prisma.ts`

```
import "dotenv/config";
import { PrismaPg } from "@prisma/adapter-pg";
import { PrismaClient } from "../generated/prisma/client";

const connectionString = `${process.env.DATABASE_URL}`;
const adapter = new PrismaPg({ connectionString });
const prisma = new PrismaClient({ adapter });
export { prisma };
```

7. Serveur + Routes simples

`src/server.ts`

```
import Fastify from "fastify"
import { prisma } from "./prisma"

const app = Fastify({ logger: true })

app.get("/posts", async () => {
```

```
  return prisma.post.findMany()
})

app.post("/posts", async (request, reply) => {
  const { title, content } = request.body as any

  const post = await prisma.post.create({
    data: { title, content }
  })

  reply.code(201)
  return post
})

app.listen({ port: 3000 }, () => {
  console.log("Server running on http://localhost:3000")
})
```

---

8. Lancement

```
npm run dev
```

Tester :

- GET http://localhost:3000/posts
- POST http://localhost:3000/posts

Suite >>