🧠 Objetivo da Aula

Ensinar os alunos a criar um servidor básico com Node.js e Express, configurar o banco de dados MongoDB, utilizar o Prisma para interagir com o banco e testar rotas com Thunder Client.

🧰 Pré-requisitos

Antes de começar, verifique se todos os alunos têm instalado:

- Node.js (versão LTS)
- VS Code
- Extensão Thunder Client no VS Code
- MongoDB Atlas (ou local)
- Conta no MongoDB (https://www.mongodb.com/)
- Terminal funcionando (CMD, Bash ou Terminal do VS Code)

📦 Etapa 1: Iniciando o Projeto

```
mkdir aula-node
cd aula-node
npm init -y
npm install express prisma @prisma/client mongoose cors dotenv
```

✍️ Etapa 2: Criando o servidor com Express

Crie o arquivo index.js:

```
const express = require('express');
const cors = require('cors');
require('dotenv').config();

const app = express();
app.use(cors());
app.use(express.json());

app.get('/', (req, res) => {
  res.send('API funcionando!');
});

const PORT = process.env.PORT || 3000;
app.listen(PORT, () => console.log(`Servidor rodando na porta ${PORT}`));
```

Crie um arquivo .env:

```
PORT=3000
DATABASE_URL=mongodb+srv://<usuario>:<senha>@cluster.mongodb.net/meubanco?retryWrites=true&w=majority
```

🧩 Etapa 3: Configurando o Prisma com MongoDB

```
npx prisma init
```

No schema.prisma (em prisma/schema.prisma):

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

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

model Aluno {
  id    String @id @default(auto()) @map("_id") @test.ObjectId
  nome  String
  idade Int
}
```

Depois, rode:

```
npx prisma generate
```

🌐 Etapa 4: Conectando ao Banco e Criando Rotas

Crie o arquivo alunoController.js:

```
const { PrismaClient } = require('@prisma/client');
const prisma = new PrismaClient();

async function listarAlunos(req, res) {
  const alunos = await prisma.aluno.findMany();
  res.json(alunos);
}

async function criarAluno(req, res) {
  const { nome, idade } = req.body;
  const aluno = await prisma.aluno.create({
    data: { nome, idade: Number(idade) },
  });
  res.status(201).json(aluno);
}

module.exports = { listarAlunos, criarAluno };
```

E um routes.js:

```
const express = require('express');
const { listarAlunos, criarAluno } = require('./alunoController');

const router = express.Router();

router.get('/alunos', listarAlunos);
router.post('/alunos', criarAluno);

module.exports = router;
```

Atualize seu index.js:

```
const express = require('express');
const cors = require('cors');
require('dotenv').config();

const routes = require('./routes');

const app = express();
app.use(cors());
app.use(express.json());
app.use(routes);

const PORT = process.env.PORT || 3000;
app.listen(PORT, () => console.log(`Servidor rodando na porta ${PORT}`));
```

🧪 Etapa 5: Testando com Thunder Client

Abra o **Thunder Client** no VS Code:

✔️ Testar rota GET:

- Método: GET
- URL: http://localhost:3000/alunos

✔️ Testar rota POST:

- Método: POST
- URL: http://localhost:3000/alunos
- Body (JSON):

```
{
  "nome": "Maria",
  "idade": 17
}
```