Aula 05

1. Instalar e configurar o PostgreSQL
2. Criar um novo projeto Node.js
3. Instalar as dependências
4. Configurar o TypeScript
5. Configurar o Sequelize
6. Criar a estrutura de diretórios
7. Criar o modelo de Partida
8. Criar a conexão com o banco de dados
9. Criar o arquivo principal da API
10. Rodar a API
11. Testar a API

01. Instalar e configurar o PostgreSQL

Primeiro, certifique-se de que o PostgreSQL esteja instalado na sua máquina. Se ainda não estiver, você pode baixá-lo e instalá-lo em postgresql.org.

02. Criar um novo projeto Node.js

No terminal, crie uma pasta para o seu projeto e inicialize o npm:

```
mkdir minha-api
cd minha-api
npm init -y
```

Isso criará o arquivo package.json na pasta.

03. Instalar as dependências

Instale o Express, Sequelize e outras ferramentas necessárias:

```
npm install express sequelize pg pg-hstore
npm install typescript ts-node @types/node @types/express --save-dev
npm install sequelize-cli --save-dev
npm install @types/sequelize --save-dev
```

- express: Framework para facilitar o desenvolvimento da API.
- sequelize: ORM que facilita a interação com o PostgreSQL.
- pg e pg-hstore: Pacote para suportar PostgreSQL no Sequelize.
- typescript, ts-node: Para adicionar suporte ao TypeScript.
- @types/node e @types/express: Tipos do Node.js e do Express.
- sequelize-cli: Ferramentas de linha de comando para configurar o Sequelize.

04. Configurar o TypeScript

Crie um arquivo de configuração do TypeScript (tsconfig.json) na raiz do projeto:

```
{
  "compilerOptions": {
    "target": "ES2020",
    "module": "commonjs",
    "strict": true,
    "esModuleInterop": true,
    "skipLibCheck": true,
    "forceConsistentCasingInFileNames": true,
    "outDir": "./dist"
  },
  "include": ["src/**/*.ts"],
  "exclude": ["node_modules"]
}
```

05. Configurar o Sequelize

Execute o comando a seguir para inicializar o Sequelize e gerar a estrutura de diretórios:

```
npx sequelize-cli init
```

Isso criará a seguinte estrutura de pastas:

- config/: Arquivo de configuração do banco de dados.
- models/: Onde os modelos do Sequelize serão definidos.
- migrations/: Para armazenar arquivos de migração.
- seeders/: Para criar dados fictícios iniciais, se necessário.

Agora, configure o arquivo config/config.json para conectar ao banco de dados PostgreSQL:

```
{
  "development": {
    "username": "postgres",
    "password": "postgres",
    "database": "minha_api",
    "host": "127.0.0.1",
    "dialect": "postgres"
  }
}
```

06. Criar a estrutura de diretórios

Agora, crie a pasta src onde colocaremos o código da nossa API:

```
mkdir src
```

07. Criar o modelo de Partida

Agora, dentro da pasta models, crie um arquivo chamado partida.ts para definir o modelo de partida:

```
// models/partida.ts
import { DataTypes, Model } from 'sequelize';
import { sequelize } from '../database';

export class Partida extends Model {
  public id!: number;
  public jogador_brancas!: number;
  public jogador_pretas!: number;
  public data!: Date;
  public resultado!: string;
}

Partida.init(
  {
    id: {
      type: DataTypes.INTEGER,
      autoIncrement: true,
      primaryKey: true,
    },
    jogador_brancas: {
      type: DataTypes.INTEGER,
      allowNull: false,
    },
    jogador_pretas: {
      type: DataTypes.INTEGER,
      allowNull: false,
    },
    data: {
      type: DataTypes.DATE,
      allowNull: false,
      defaultValue: DataTypes.NOW,
    },
    resultado: {
      type: DataTypes.STRING,
      allowNull: false,
      validate: {
        isIn: [['brancas', 'pretas', 'empate']],
      },
    },
```

```
  },
  {
    sequelize,
    modelName: 'Partida',
    tableName: 'partidas',
  }
);
```

08. Criar a conexão com o banco de dados

Crie um arquivo src/database.ts para configurar a conexão com o banco de dados:

```
// src/database.ts
import { Sequelize } from 'sequelize';

export const sequelize = new Sequelize('postgres', 'postgres', 'postgres', {
  host: 'localhost',
  dialect: 'postgres',
});
```

09. Criar o arquivo principal da API

Dentro da pasta src, crie um arquivo chamado index.ts que será o ponto de entrada da nossa aplicação.

```
// src/index.ts
import express, { Request, Response } from 'express';
import { sequelize } from './database';
import { Partida } from './models/partida';

const app = express();
const port = 3000;

app.use(express.json());

// Conectar ao banco de dados
sequelize
  .authenticate()
  .then(() => {
    console.log('Conexão com o banco de dados estabelecida com sucesso.');
  })
  .catch((err) => {
    console.error('Erro ao conectar ao banco de dados:', err);
  });

// Sincronizar modelos
```

```
sequelize.sync({ force: true }).then(() => {
  console.log('Tabelas sincronizadas.');
});

// Rota principal da API
app.get('/', (req: Request, res: Response) => {
  res.send('Hello, API com TypeScript e PostgreSQL - Partidas de Xadrez!');
});

// Rota para listar todas as partidas
app.get('/partidas', async (req: Request, res: Response) => {
  const partidas = await Partida.findAll();
  res.json(partidas);
});

// Rota para criar uma nova partida
app.post('/partidas', async (req: Request, res: Response) => {
  const { jogador_brancas, jogador_pretas, data, resultado } = req.body;

  if (!jogador_brancas || !jogador_pretas || !resultado) {
    return res.status(400).json({ error: 'jogador_brancas, jogador_pretas e Resultado são ob
  }

  if !data {
    const novaPartida = await Partida.create({
      jogador_brancas,
      jogador_pretas,
      resultado,
    });
  }
  else {
    const novaPartida = await Partida.create({
      jogador_brancas,
      jogador_pretas,
      data,
      resultado,
    });
  }

  res.status(201).json(novaPartida);
});

// Inicia o servidor
app.listen(port, () => {
  console.log(`Servidor rodando na porta ${port}`);
});
```

10. Rodar a API

Adicione o script para rodar a API diretamente no package.json. Abra o arquivo package.json e adicione o seguinte script:

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

Agora você pode rodar a API com o comando `npm run dev`. Isso vai rodar o servidor e ele estará escutando na porta 3000.

11. Testar a API

Você pode testar as rotas da API com ferramentas como o Postman.