Aplicação com Banco de Dados

Nesta fase do curso, aprofundaremos nosso conhecimento sobre as bases de dados não relacionais, suas características distintas e aplicações práticas. Um dos nossos objetivos é determinar como o código JavaScript pode ser integrado com o banco de dados para testes. Serão abordados quatro pontos-chave: a apresentação do NoSQL, a configuração de um banco de dados local, a criação de uma aplicação web utilizando Express, e, por fim, a integração do banco de dados com nossa aplicação, destacando como Node.js se integra a esse contexto.

Apresentação de NoSQL

NoSQL representa os bancos de dados não relacionais e oferece uma estrutura de armazenamento flexível, ideal para dados não estruturados ou semiestruturados. Diferentemente dos bancos relacionais, os dados em NoSQL são organizados em coleções, e não tabelas, consistindo em documentos e campos, que formam uma estrutura de documentos JSON em chave-valor. Esta abordagem permite um modelo de dados mais dinâmico e escalável, adequado para as demandas de aplicações modernas que requerem alta velocidade e volumes variáveis de dados.

Durante as aulas, o MongoDB será nosso foco prático, pela sua popularidade e versatilidade. Também exploraremos outros bancos NoSQL, como Redis, Cassandra e Amazon Dynamo, para compreender a diversidade do NoSQL. O uso do MongoDB em nossa aplicação servirá como teste prático para integrar Node.js com o banco de dados, permitindo uma experiência educacional completa em desenvolvimento de aplicações.

Na sequência do curso, dedicaremos tempo à instalação do MongoDB, familiarizando-nos com a plataforma e explorando outras estruturas de dados além das tradicionais. Este conhecimento será crucial para o desenvolvimento da nossa aplicação web, onde aplicaremos os conceitos aprendidos para gerenciar coleções de dados de forma eficiente, iniciando com a coleção de médicos como nosso primeiro exemplo prático.

Exemplo de conexão com MongoDB usando Mongoose:

```
const mongoose = require('mongoose');

mongoose.connect('mongodb://localhost:27017/medDB',
{useNewUrlParser: true, useUnifiedTopology: true});
```

Ao mergulhar no universo NoSQL, abrimos novas possibilidades para o tratamento e análise de dados, preparando-nos para enfrentar os desafios do desenvolvimento de software com uma base sólida e atualizada. Este módulo não apenas enriquecerá nosso entendimento teórico, mas também nos equipará com as habilidades práticas necessárias para a implementação de soluções de banco de dados inovadoras e eficazes.

Preparação do Banco de Dados Local

Na continuação, abordaremos um aspecto crucial no desenvolvimento de aplicações modernas: a "Preparação do Banco de Dados Local". Como mencionado anteriormente, embora existam opções de bancos de dados em nuvem, como o MongoDB Atlas, nosso foco inicial será na configuração e uso de uma instância local do banco de dados, utilizando o MongoCompass. Esta ferramenta facilitará o gerenciamento de nossos projetos e permitirá uma manipulação mais direta dos dados.

Para iniciar, é importante destacar a importância de selecionar a versão correta do MongoDB para instalação. Recomenda-se a versão Community do MongoDB Compass, que, além de gratuita, satisfaz plenamente nossas

necessidades educacionais e experimentais. Durante a instalação, um processo simples e intuitivo, escolheremos as configurações padrão para facilitar o processo. Este passo é fundamental para garantir uma base sólida para nossos experimentos e aplicações futuras.

Uma vez instalado o MongoDB Compass, exploraremos sua interface, notadamente mais limpa e direta em comparação com as ferramentas de gerenciamento de bancos de dados relacionais. A simplicidade na visualização de coleções e a facilidade de conexão, geralmente através do localhost, facilitam o início rápido em projetos. Essa abordagem focada em coleções ao invés de tabelas tradicionais reflete a natureza flexível e escalável dos bancos de dados NoSQL.

O próximo passo será a criação de nossa base de dados e as coleções necessárias para o projeto. No MongoDB, uma coleção agrupa documentos, que são os registros de dados representados em formato JSON, permitindo uma estrutura de dados dinâmica e rica. Exemplificaremos essa prática com a criação de uma coleção de médicos, onde cada documento representará um médico com atributos específicos, como nome, telefone e especialidade, seguindo a estrutura chave-valor do JSON.

Finalmente, destacaremos as funcionalidades adicionais do Compass, como a importação de dados via documentos JSON, a possibilidade de exportação de coleções para uso externo ou análise, e as ferramentas de visualização e validação de esquemas. Essas características tornam o MongoCompass uma ferramenta poderosa e acessível para desenvolvedores, facilitando a integração e manipulação de dados em nossas aplicações.

Portanto, essa etapa da preparação do banco de dados local não só nos familiariza com as operações básicas de um banco NoSQL, mas também

nos prepara para as próximas fases do projeto, onde integraremos o banco de dados à nossa aplicação com Express.

A importância de estabelecer um ambiente de teste local com MongoDB é evidente quando consideramos a necessidade de integrar o Node.js com o banco de dados dentro da disciplina. Testar localmente nos permite uma maior agilidade e controle no desenvolvimento, possibilitando que os alunos experimentem diretamente as nuances da integração entre o código JavaScript no Node.js e as operações de banco de dados. Além disso, essa abordagem incentiva a experimentação e a aprendizagem ativa, facilitando a compreensão dos conceitos de NoSQL e a aplicação prática de técnicas de desenvolvimento web modernas. Desta forma, a configuração de um banco de dados local torna-se não apenas um exercício técnico, mas também um componente estratégico no processo educacional, permitindo que os estudantes testem e validem suas soluções em um ambiente controlado e eficiente, preparando-os para desafios mais complexos no mundo real.

Criação da aplicação com Express

Esta fase é crucial para a construção do projeto da disciplina e proporciona uma compreensão prática do uso do Express, um framework essencial para o desenvolvimento de aplicações web utilizando Node.js.

Para contextualizar, o Express facilita a configuração e a gestão das rotas da aplicação, oferecendo um conjunto de recursos que tornam o desenvolvimento mais ágil e eficiente. Similar ao Spring Boot do Java em sua proposta de simplificar a criação de aplicações, o Express se destaca no ecossistema Node por sua popularidade e eficácia.

Na prática, começaremos estruturando nossa aplicação denominada “Med App”, uma referência ao contexto de um consultório médico que estamos simulando. A escolha do nome reflete a natureza web da aplicação, não se

limitando a um aplicativo móvel, mas abrangendo uma solução mais ampla e acessível via navegador.

O primeiro passo envolve a inicialização do projeto com `npm init` dentro da pasta destinada ao projeto. Este processo configura o `package.json`, que é fundamental para gerenciar as dependências e configurações do projeto. A estrutura inicial do projeto inclui a criação de diretórios para Models, Rotas, Repositórios e Serviços, que organizam o código de maneira lógica e funcional.

A instalação do Express é feita via NPM, adicionando-o como uma dependência do projeto. Este momento é propício para verificar quaisquer alertas de versão ou compatibilidade durante a instalação, assegurando que a base do projeto esteja sólida e atualizada. A integração do Express com o MongoDB em nossa aplicação de teste exemplificará o processo de conexão e manipulação de dados em um ambiente de desenvolvimento real, usando Node.js.

Este é um exemplo de inicialização da aplicação e configuração básica do Express:

```
const express = require('express');

const app = express();

app.get('/', (req, res) => {

  res.send('Bem-vindo à MedApp!');

});

app.listen(3000, () => {

  console.log('Aplicação escutando na porta 3000');
```

```
});
```

Com o Express instalado, a criação do arquivo `index.js` serve como ponto de entrada da aplicação. Este arquivo incorporará as configurações iniciais do Express, incluindo a definição de rotas básicas e a configuração do servidor para escutar na porta 3000. Esse passo é vital para garantir que a aplicação esteja operacional e pronta para ser expandida com funcionalidades específicas.

Os próximos passos incluirão a detalhamento das rotas no arquivo `router.js`, que direcionará as requisições HTTP para os controladores apropriados, refletindo a lógica de negócio da nossa aplicação Med App. A estrutura de pastas e arquivos criada anteriormente servirá de base para a implementação de entidades, serviços e repositórios, que serão detalhados e desenvolvidos nas aulas subsequentes.

Ao final desta fase, teremos uma aplicação Express configurada e pronta para evolução, com um ambiente organizado que facilitará a adição de novas funcionalidades e a integração com o banco de dados. Esta etapa consolida o alicerce sobre o qual construiremos nossa aplicação, demonstrando a eficácia do Express no desenvolvimento de soluções web com Node.js.

A seguir, integraremos as entidades ao projeto e realizaremos a conexão com o banco de dados, marcando um avanço significativo no desenvolvimento do nosso projeto prático.

Adição de bancos de dados na aplicação

Este segmento conclui nossa exploração prática sobre a integração de bancos de dados à aplicação, um passo crucial para o desenvolvimento robusto e flexível de software.

Neste contexto, discutiremos duas ferramentas essenciais para a gestão de dados em aplicações Node.js: o Sequelize e o Mongoose. O Sequelize é um ORM (Object-Relational Mapping) destinado a bancos de dados relacionais como MySQL, PostgreSQL, entre outros. Ele oferece uma abordagem programática para mapear objetos de aplicativo a tabelas de banco de dados, facilitando operações de CRUD (Create, Read, Update, Delete) através de métodos intuitivos e abstratos.

Por outro lado, o Mongoose é dedicado a bancos de dados não relacionais, especificamente MongoDB. Ele proporciona uma modelagem de dados esquemática para MongoDB, permitindo a definição de tipos de dados e validações com facilidade, além de métodos para interação com o banco. Em nossa aplicação, utilizaremos o Mongoose para exemplificar a manipulação de dados em um contexto não relacional.

Para exemplificar, abordaremos a configuração inicial do Mongoose, começando pela instalação do pacote via npm. Seguiremos para a criação do arquivo de configuração do banco de dados, onde especificaremos a conexão ao MongoDB local, utilizando a porta padrão e definindo a coleção para nossa aplicação. Este arquivo gerenciará a conexão e monitorará o estado do banco, reportando erros ou confirmando a conexão bem-sucedida.

Este é um exemplo de definição de um esquema e modelo para médicos:

```
const mongoose = require('mongoose');

const Schema = mongoose.Schema;

const medicoSchema = new Schema({

  nome: String,

  especialidade: String,
```

```
  contato: String

});

const Medico = mongoose.model('Medico', medicoSchema);
```

Já este, utilizando Sequelize:

```
const Sequelize = require('sequelize');

const sequelize = new Sequelize('database', 'username',
'password', {

  host: 'localhost',

  dialect: 'mysql'

});
```

Em seguida, introduziremos os modelos de dados, essenciais para a interação com o banco. Criaremos esquemas para definir a estrutura de nossas coleções, especificando tipos de dados, validações e relações entre documentos. Este passo é fundamental para garantir a integridade dos dados e a eficácia das operações de banco.

Por fim, discutiremos a importância de exportar os modelos configurados, permitindo que sejam utilizados em outras partes da aplicação. Isso inclui a interação com a API, onde os modelos serão empregados para realizar consultas, inserções, atualizações e exclusões de documentos.

Esta etapa encerra nossa jornada pelo desenvolvimento de uma aplicação com banco de dados, desde a configuração do ambiente até a manipulação de dados. Os conhecimentos adquiridos aqui servem de base para projetos mais complexos, destacando a importância da escolha adequada de

ferramentas conforme o tipo de dados e as necessidades específicas de cada aplicação.

Aguardamos vocês nas próximas aulas, onde aplicaremos os conceitos discutidos em cenários práticos, solidificando o aprendizado e preparando-os para desafios mais avançados em Programação II. Até lá, continuem explorando e praticando com as ferramentas apresentadas.

GitHub da Disciplina

Você pode acessar o repositório da disciplina no GitHub a partir do seguinte link: https://github.com/FaculdadeDescomplica/ProgramacaoII. Esse repositório tem como principal objetivo guardar os códigos das aulas práticas da disciplina para aprimorar suas habilidades em vários tópicos, incluindo a criação e consumo de APIs com controle de autenticação utilizando Node.js e utilizando boas práticas de programação e mercado.

Conteúdo Bônus

Para complementar o aprendizado e aprofundar os conhecimentos adquiridos em aula, recomendo o vídeo "Criando uma API Completa com Node.js, Express e MongoDB - Passo a Passo do Zero!", disponível no YouTube no canal "Gabriel Rangel - Fala, Coders!" , esse material irá ajudá-lo a consolidar a teoria vista com a prática de desenvolvimento real.

Referências Bibliográficas

Bibliografia Básica:

ELMASRI, R.; NAVATHE, S. B. **Sistemas de banco de dados**. 7.ed. Pearson: 2018.

MEDEIROS, L. F. de. **Banco de dados**: princípios e prática. Intersaberes: 2013.

VICCI, C. (Org.). **Banco de dados**. Pearson: 2014.

Bibliografia Complementar:

CARDOSO, L. da C. **Design de aplicativos**. Intersaberes: 2022.

JOÃO, B. do N. **Usabilidade e interface homem-máquina**. Pearson: 2017.

LEAL, G. C. L. **Linguagem, programação e banco de dados**: guia prático de aprendizagem. Intersaberes: 2015.

PUGA, S.; FRANÇA, E.; GOYA, M. **Banco de dados**: implementação em SQL, PL/SQL e Oracle 11g. Pearson: 2013.

SETZER, V. W.; SILVA, F. S. C. **Bancos de dados**. Blucher: 2005.

Ir para exercício