Adicionando CRUD na aplicação

Nesta aula, vamos mergulhar na mecânica fundamental que sustenta as aplicações modernas: as operações de CRUD - Create, Read, Update, Delete. Este conjunto de operações básicas forma a espinha dorsal da interação entre os usuários e os dados armazenados, permitindo a criação, leitura, atualização e exclusão de registros dentro de nossa aplicação. Com foco prático, delinearemos a estrutura necessária para implementar essas ações essenciais, começando com a revisão do nosso Diagrama de Entidade-Relacionamento, uma ferramenta crucial para compreender as interconexões entre as entidades que compõem nosso projeto. Ao longo desta aula, construiremos uma base sólida que não apenas fortalecerá seu entendimento teórico, mas também preparará o terreno para a aplicação direta desses conceitos essenciais ao desenvolvimento de software.

Revisão de Entidade-Relacionamento do Mini Projeto

Neste módulo, mergulharemos na implementação do CRUD (Create, Read, Update, Delete) em nossa aplicação. Este processo é dividido em quatro etapas cruciais, iniciando com a revisão do Diagrama de Entidade-Relacionamento (ER) do nosso mini projeto. Este diagrama é fundamental, por esboçar a estrutura sobre a qual nossa aplicação é construída, detalhando entidades, atributos e os relacionamentos entre eles, de forma a espelhar situações do mundo real dentro do contexto do software.

O primeiro passo é relembrar a estrutura do nosso Diagrama ER, verificando se todas as entidades necessárias já estão incorporadas ao projeto. Caso faltem itens no seu projeto, este é o momento ideal para adicionar e ajustar conforme necessário. O Diagrama ER atua como um mapa, guiando o

desenvolvimento do sistema ao definir claramente as entidades e seus relacionamentos, facilitando assim, a implementação de operações CRUD de maneira eficaz.

Posteriormente, avançaremos para o mapeamento das rotas dos controladores, estabelecendo uma conexão entre o modelo, a visualização e o controlador. Seguindo com a criação dos modelos com base no Diagrama ER, validaremos os modelos criados e compararemos com o modelo relacional previamente estudado. Finalmente, dedicaremos atenção à implementação das operações CRUD, compreendendo as ações necessárias para adicionar, ler, atualizar e deletar registros no contexto do nosso mini projeto.

Ao revisarmos nosso Diagrama ER, nos concentramos nas entidades essenciais do nosso sistema de consultório médico, como consultas (appointments), médicos, pacientes e prescrições, e seus inter-relacionamentos. Essa revisão nos permite uma abstração de alto nível, onde relacionamos as entidades da aplicação às situações reais, facilitando assim a transição da teoria para a prática de codificação.

Nesse contexto, exploramos o exemplo da entidade consulta, que se relaciona com médicos, pacientes e prescrições, mostrando como estas relações simulam o funcionamento real de um consultório médico dentro da nossa aplicação. A representação tradicional de entidades e relacionamentos em um diagrama ER, utilizando retângulos para entidades e losangos para relacionamentos, é discutida, bem como a importância de entender a unicidade e a repetição de atributos, destacando a chave primária de cada entidade.

Esta introdução ao CRUD no âmbito do nosso mini projeto de consultório médico prepara o terreno para as próximas etapas, onde nos aprofundaremos na implementação efetiva das operações CRUD, mapeamento de rotas e validação de modelos. Estes passos são essenciais para o desenvolvimento de uma aplicação robusta, flexível e de fácil manutenção, alinhados com as práticas recomendadas de engenharia de software. A seguir, focaremos no mapeamento das rotas dos controladores,

dando um passo adicional em direção à preparação da nossa aplicação para as operações CRUD.

Mapeamento de Rotas dos Controladores

Prosseguindo, a segunda etapa do nosso processo envolve o mapeamento das rotas dos controladores, uma fase crucial para a integração e o funcionamento eficaz da nossa aplicação. Nesta parte, destacaremos como organizar e direcionar as rotas utilizando o Express, um framework do Node.js que já configuramos previamente na nossa aplicação.

No coração deste módulo, o mapeamento de rotas representa a definição de como as solicitações dos usuários são encaminhadas para os controladores adequados, garantindo que a aplicação responda de maneira coerente. Essa organização não apenas simplifica a gestão das rotas, mas também potencializa a clareza e a manutenção do código.

Para entrar na prática, assuma que você está em seu ambiente de desenvolvimento com o projeto aberto. No diretório de rotas, verifique se seu arquivo router.js já está preparado. Caso contrário, é neste momento que estabeleceremos as conexões necessárias. Inicialmente, importaremos as dependências dos controladores, como o `appointmentController`, `doctorController`, `patientController`, e `prescriptionController` de seus respectivos arquivos, organizando nosso projeto para que cada aspecto da aplicação possa ser acessado através de rotas específicas.

Após a importação, utilizaremos a função `router.use` do Express para mapear a raiz de cada um desses controladores. Esse processo assegura que, a partir da entrada principal da aplicação, as rotas corretas sejam acessíveis, estabelecendo um caminho claro para as operações de CRUD que implementaremos. É importante que cada controlador seja mapeado adequadamente, garantindo a cobertura completa das funcionalidades planejadas para o projeto.

Entretanto, apenas mapear as rotas no roteador não é suficiente. Cada controlador precisa ser preparado para ser reconhecido como um módulo e

responder às requisições adequadamente. Para isso, cada arquivo de controlador deve importar o Express e utilizar o método `router` para definir as rotas específicas daquela parte da aplicação. Finalizamos cada controlador exportando-o como um módulo, permitindo sua integração efetiva com o sistema de roteamento do Express.

Essa abordagem modular e organizada facilita a manutenção do código e a adição de novas funcionalidades, além de preparar o terreno para a implementação prática das operações de CRUD, que será o foco das próximas etapas de nosso mini projeto. A clareza na definição das rotas e a estruturação adequada dos controladores são fundamentais para o sucesso do desenvolvimento, permitindo que a aplicação cresça de forma organizada e sustentável.

Em seguida, mergulharemos nas especificidades de cada ação do CRUD, aplicando os conceitos teóricos à prática conforme as necessidades do nosso projeto. Além disso, revisaremos os modelos para assegurar que estejam alinhados com o mapeamento realizado e prontos para a implementação das funcionalidades. Portanto, atualize seu projeto, afie sua curiosidade e prepare-se para avançar ainda mais na construção da nossa aplicação.

Criação dos Modelos com Entidade-Relacionamento

Aqui, aprofundaremos no processo de criação dos modelos com base no Diagrama de Entidade-Relacionamento (ER), crucial para estruturar a forma como as entidades se interligam na nossa aplicação. Com o enfoque nos bancos de dados não relacionais, exploraremos como os relacionamentos se manifestam e se diferenciam do contexto relacional tradicional.

Embora já tenhamos discutido e definido modelos anteriormente, a revisão se faz necessária para assegurar a correta implementação dos relacionamentos conforme previsto no nosso Diagrama ER. Essa etapa é vital, especialmente considerando que nossa aplicação utiliza um banco de dados não relacional, onde os relacionamentos precisam ser adaptados

para essa realidade.

Para ilustrar a diferença entre os bancos relacionais e não relacionais, recorremos a um exemplo de um projeto anterior que utilizava o Sequelize com um banco de dados relacional. No Sequelize, os tipos de dados e as relações são definidos de maneira que espelham diretamente a estrutura do banco de dados relacional, incluindo aspectos como autoincremento e campos obrigatórios. Esse paralelo ajuda a entender como os modelos e relacionamentos são construídos em uma arquitetura relacional.

Em contrapartida, ao revisarmos nossos modelos no contexto de um banco de dados não relacional, observamos que os relacionamentos são estabelecidos de maneira menos explícita. Eles são inferidos através das operações que interligam dados de diferentes coleções (ou 'documentos'), requerendo uma implementação cuidadosa para assegurar a integridade e o funcionamento adequado das relações entre entidades.

Por exemplo, a entidade Prescription em nosso projeto possui atributos obrigatórios e opcionais, incluindo relações implícitas com outras entidades, como Appointments. Essa relação é sugerida pela presença de um campo obrigatório que vincula a Prescription a um determinado Appointment, mas sem a explicitação comum encontrada nos bancos de dados relacionais.

A criação desses modelos no contexto não relacional demanda uma atenção particular às operações que efetivamente estabelecem esses vínculos, como inserções e consultas que referenciam IDs de outras coleções. Esse método garante que, mesmo em um banco de dados não relacional, possamos representar e manipular relações complexas entre as entidades de nossa aplicação.

Ao final desta etapa, é fundamental que você valide se os modelos estão adequadamente configurados e alinhados com o Diagrama ER, contemplando todos os atributos necessários e respeitando os relacionamentos definidos. Essa conformidade é essencial para avançarmos para a implementação das operações de CRUD, onde aplicaremos na prática o que foi planejado e modelado.

Com os modelos definidos e os relacionamentos claramente estabelecidos,

mesmo que de forma adaptada ao contexto não relacional, estamos preparados para o próximo passo: a implementação das operações de CRUD em nossa aplicação. Essa etapa nos permitirá explorar a fundo o significado e a aplicação de cada uma dessas operações no desenvolvimento do nosso projeto. Prepare-se para colocar em prática os conceitos aprendidos e avançar na construção da nossa aplicação.

Criação das Ações de CRUD

Para finalizar, abordaremos a implementação das ações de CRUD (Create, Read, Update, Delete) na nossa aplicação. As ações de CRUD são fundamentais em qualquer desenvolvimento de software por comporem as operações básicas que permitem manipular os dados armazenados em nossa aplicação.

CRUD é um acrônimo para Create (Criar), Read (Ler), Update (Atualizar) e Delete (Deletar). Essas operações são essenciais para interagir com a base de dados, permitindo-nos criar novos registros, ler e exibir informações, atualizar dados existentes e deletar registros indesejados. Essa abordagem é universal, aplicável em diversas linguagens de programação e tipos de base de dados, seja ela relacional ou não relacional.

Para a criação das ações de CRUD em nossa aplicação, utilizaremos o Express, um framework do Node.js, que simplifica o mapeamento e a manipulação de rotas. Inicialmente, é preciso garantir que todas as dependências necessárias estejam corretamente importadas em nossos controladores, incluindo os modelos de dados com os quais trabalharemos.

A implementação começa no nível mais baixo com a criação de um repositório (repository) para cada entidade, onde definimos as funções específicas para as operações de CRUD. Por exemplo, para adicionar uma nova consulta (appointment), criamos uma função assíncrona `saveAppointment` no arquivo `appointmentRepository.js`, que receberá os dados necessários para a criação do registro no banco de dados. Esse processo se repete para as operações de leitura, atualização e deleção,

respeitando a lógica específica de cada ação.

Após definir as funções no repositório, movemos para a camada de serviço (service), onde chamamos essas funções do repositório, aplicando regras de negócio adicionais se necessário. É nessa camada que podemos implementar validações ou tratamentos específicos antes de executar a operação no banco de dados.

Finalmente, no controlador (controller), mapeamos as rotas HTTP para as funções correspondentes no serviço, definindo a lógica de como as requisições serão tratadas e respondidas. Para cada ação de CRUD, associamos uma rota específica, por exemplo, uma rota POST para criar um novo registro, GET para leitura, PUT para atualização e DELETE para remoção de um registro.

Essa metodologia de implementação, seguindo a sequência repositório > serviço > controlador, garante uma organização clara e uma separação de responsabilidades na aplicação, facilitando a manutenção e a escalabilidade do projeto.

Encerramos esta aula com a compreensão das ações de CRUD e sua implementação prática. Esse conhecimento é fundamental para o desenvolvimento de aplicações dinâmicas e interativas, permitindo que você avance no mundo da programação com confiança. Agora, cabe a você aplicar esses conceitos nas demais entidades da aplicação, solidificando sua compreensão e habilidades em CRUD. Até a próxima aula!

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 nossa jornada e reforçar a compreensão e aplicação das operações de CRUD (Create, Read, Update, Delete) em aplicações Node.js, recomendo o vídeo “Node.js API testável #5 - Operações de CRUD (Create, Read, Update, Delete)”. Este conteúdo é apresentado por Waldemar Neto no canal “Waldemar Neto - Dev Lab” disponível no YouTube.

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