

Trabalho Prático | DGT2815 - Introdução à Integração de Páginas Web com Banco de Dados

Material de orientações Trabalho Prático da disciplina
DGT2815 - Integração de Páginas Web com Banco de Dados

DGT2818 - Integração de Páginas Web com Banco de Dados**DGT2815 - Integração de Páginas Web com Banco de Dados**

Objetivos da prática

A partir dos objetivos listados abaixo, no final do projeto, você terá criado um servidor baseado em Express e Mongoose, com acesso ao banco de dados MongoDB, além de utilizar diversas tecnologias para implementar o front-end, tornando-se capacitado para lidar com plataformas de desenvolvimento variadas, e satisfazendo às necessidades de um mercado cada vez mais heterogêneo:

- Utilizar o banco de dados MongoDB para a criação de repositório NoSQL
- Implementar servidor baseado em Express e Mongoose
- Implementar biblioteca cliente de acesso em Java Script
- Utilizar front-ends baseados em React JS, Next JS e Angular

Materiais necessários para a prática

- Computador com acesso à internet;
- Editor de código Visual Studio Code;
- Banco de dados MongoDB e ferramenta Compass;

- Ambiente de desenvolvimento NodeJS;
- Navegador de internet instalado no computador.

Desenvolvimento da prática

Codifique as entidades contendo ao menos três elementos no formato JSON, na classe ControleEditoraService, que já estará configurada como serviço, devido à anotação `Injectable`

Adicionar os métodos `getNomeEditora` e `getEditoras`

Implementar o método `getEditoras`, com o retorno do vetor editoras

Implementar o método `getNomeEditora`, recebendo `codEditora`, do tipo numérico, e retornando o nome da editora, através de uma operação `filter` sobre o vetor editoras

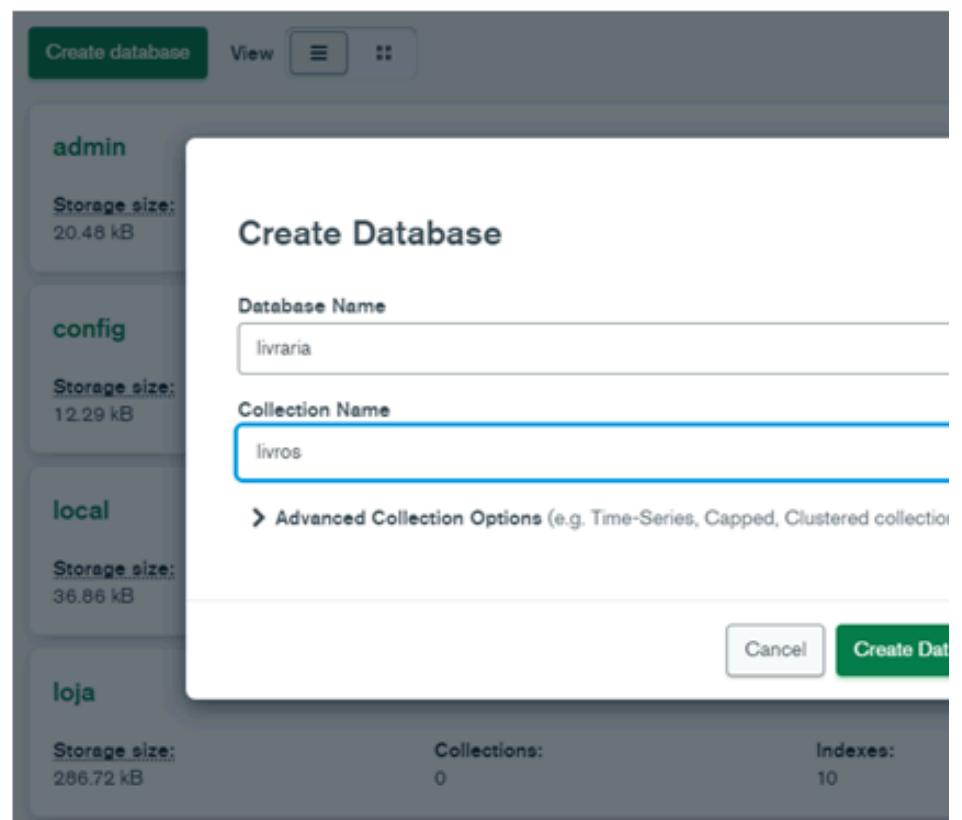
Obs.: Pode ser adaptado o código de `ControleEditora`, implementado na prática do nível 3 – "Meu Primeiro Framework".

Codifique o controlador de livros, no arquivo `controle-livros.service.ts`:

Vamos colocar a **mão na massa**?! Siga as **instruções** abaixo para desenvolvimento desta missão.

👉 1º Procedimento | Criando o Servidor com Express e Mongoose

- Crie o banco de dados **livraria**, e a coleção **livros**, no **MongoDB Compass**, através da opção Create Database, como na imagem seguinte:



- Crie o aplicativo `livros-servidor`, baseado em Node JS, com Express e Mongoose:
 - Instalar o gerador de aplicativos do **Express**, de forma **global**, utilizando o comando `npm install -g express-generator`

- b. Gerar o servidor, através do comando **express -e livro-servidor**
 - c. Entrar no diretório gerado, utilizando o comando **cd livro-servidor**
 - d. Instalar as dependências do Node JS, através do comando **npm install**
 - e. Instalar o Mongoose, utilizando o comando **npm install mongoose**
 - f. Instalar o gerenciador de CORS, utilizando o comando **npm install cors**
- Através do ambiente do Visual Studio Code, defina a conexão com o banco de dados MongoDB e a classe de modelo para o Mongoose:
- a. Adicionar uma pasta com o nome **modelo**
 - b. Na pasta **modelo**, criar os arquivos "**conexao.js**" e "**livro-schema.js**"
 - c. Em "**conexao.js**", definir uma variável **banco**, referenciando a biblioteca **mongoose**, e uma variável **options**, no formato JSON, com os atributos **useUnifiedTopology** e **useNewUrlParser**, ambos com valor **true**
 - d. Efetuar a conexão com o MongoDB, a partir da variável **banco**, e exportar a variável no padrão de módulo do Java Script
 - e. Em "**livro-schema.js**", importar a conexão efetuada na variável **banco** e definir a estrutura do banco, na variável **LivroSchema**, instanciando um objeto do tipo **banco.Schema**, e passando a estrutura no formato **JSON**
- f. Associar **LivroSchema** e a coleção **livros**, definida na criação do banco, ao modelo de dados **Livro**, através de uma chamada para **banco.model**
- g. Exportar o modelo **Livro** no padrão de módulo do Java Script
- Implemente o gerenciamento de dados para Livro:
- a. Criar o arquivo "**livro-dao.js**", na pasta **modelo**
 - b. Em "**livro-dao.js**", definir uma variável **Livro**, referenciando o modelo exportado por **livro-schema**
 - c. Adicionar a função **obterLivros**, no estilo Arrow Function, assíncrona, sem parâmetros, e que deve retornar o conjunto de livros obtidos na chamada para o método **find**, sem parâmetros, de **Livro**
 - d. Adicionar a função **incluir**, no estilo Arrow Function, assíncrona, com um parâmetro **livro**, no formato JSON, invocando o método **create**, de **Livro**, com a passagem do valor fornecido

- e. Adicionar a função **excluir**, no estilo Arrow Function, assíncrona, com um parâmetro **codigo**, invocando o método **deleteOne**, de **Livro**, onde será utilizado um objeto JSON, com o atributo **_id** recebendo o valor de **codigo**
 - f. Exportar as três funções definidas, com base no padrão de módulo
- Implemente as rotas que serão utilizadas para o gerenciamento dos livros:
 - a. Criar o arquivo "**livros.js**", na pasta **routes**, onde estão definidas as rotas criadas por padrão pelo Express Generator (index e users)
 - b. No arquivo "**livros.js**", com a inclusão das funções **obterLivros**, **incluir** e **excluir**, a partir de **livro-dao**, criação da variável **express**, referenciando a biblioteca de mesmo nome, e criação do objeto **router**, obtido a partir da chamada para o método **Router**, de **express**
 - c. Adicionar a resposta para a rota **raiz**, no modo **GET**, a partir do método de **router** com o mesmo nome, com envio dos **livros**, no formato JSON, obtidos através da chamada para **obterLivros**
 - d. Adicionar a resposta para a rota **raiz**, no modo **POST**, a partir do método de **router** com o mesmo nome, com a recepção de um **livro**, no **corpo** da requisição, passagem do livro para a função **incluir**, e envio de mensagem indicando sucesso ou falha na operação, no formato JSON
 - e. Adicionar a resposta para a rota **raiz**, concatenada a um segmento com o **código** do livro (**_id**), no modo **DELETE**, a partir do método de **router** com o mesmo nome, com a extração do **segmento** com o código, a partir do campo **params** da requisição, passagem do código para a função **excluir**, e envio de mensagem indicando sucesso ou falha na operação
 - f. Exportar **router**, com base no padrão de módulo do Java Script
- Configure o servidor express, modificando a porta, gerenciando o sistema de CORS (Cross-Origin Resource Sharing), e incluindo as rotas:
 - a. Definir a porta utilizada como **3030**, no arquivo **www**, da pasta **bin**, com a mudança de **var port = normalizePort(process.env.PORT || '3000')** para **var port = normalizePort(process.env.PORT || '3030')**
 - b. Em "**app.js**", na pasta **raiz** do projeto, definir uma variável **livroRouter**, referenciando **routes/livros**
 - c. Adicionar a rota no objeto **app**, referenciando a url de base "**/livros**" e o objeto de roteamento **livroRouter**, com a chamada para o método **use**
 - d. Para as duas ações anteriores, posicionar as linhas de código na sequência das utilizadas para a rota **raiz** e a rota **users**, adicionadas por padrão
 - e. Configurar o sistema de CORS, definindo uma variável **cors**, que receberá a biblioteca de mesmo nome, antes de instanciar **app**, e utilizá-la pelo app logo após instanciar, através do método **use**, mantendo a configuração padrão, onde temos acesso a partir de qualquer porta ou endereço

- Execute o servidor com **npm start**, e teste as chamadas através do **Postman** ou do **Boomerang**, lembrando que apenas o método GET pode ser verificado através do navegador;
- Após testar o servidor, utilize também o MongoDB Compass para verificar os dados que foram inseridos

👉 2º Procedimento | Alteração dos Projetos Clientes

- Crie um diretório denominado **clientes**, e copiar para dentro dele os projetos **livros-react**, **livros-next** e **livros-angular**;
- Abra a cópia do projeto **livros-react** no Visual Studio Code;
- Altere a classe **Livro**, no arquivo **src/model/Livro.ts**, mudando o tipo do atributo **codigo** para **String**, com o objetivo de compatibilizar com os identificadores do banco de dados MongoDB
- Altere a classe **ControleLivros**, no arquivo **src/controle/ControleLivros.ts**, de acordo os passos seguintes:
 - a. Definir a constante global **baseURL**, recebendo o **endereço de base** do servidor Express, em "**http://localhost:3030/livros**"

b. Definir a interface **LivroMongo**, para compatibilizar o tipo de **Livro** às chamadas para o servidor, contendo os atributos apresentados a seguir:

- c. **Eliminar** o vetor **livros**, que funcionava como fonte de dados
- d. Alterar o método **obterLivros** para comportamento **assíncrono**, com uso de **fetch** no endereço de base, modo **GET**, recuperação da resposta como vetor **JSON**, e retorno com o **mapeamento** de cada elemento para um novo objeto do tipo **Livro**
- e. Alterar o método **excluir** para o comportamento **assíncrono**, recebendo **codigo**, do tipo **String**, chamando **baseURL/codigo** via **fetch**, no modo **DELETE**, e retornando o campo **ok** da resposta, indicando sucesso ou falha
- f. Alterar o método **incluir** para o comportamento **assíncrono**, adotando o parâmetro **livro**, do tipo **Livro**, com a conversão dele para uma interface **LivroMongo**, chamada para **baseURL** via **fetch**, no modo **POST**, incluindo o tipo de conteúdo como informação do header e a interface, que deve ser convertida para texto através de **JSON.stringify**, no corpo. O retorno da função será o campo **ok** da resposta, indicando sucesso ou falha
- Altere o código-fonte de **LivroLista**, no arquivo **src/LivroLista.js**, de acordo com os passos seguintes:
 - a. Alterar a implementação de **useEffect**, adotando o modelo **assíncrono** na chamada para **obterTodos** do controlador, seguida do operador **then**, e consequente invocação de **setLivros** com uso do resultado
 - b. Alterar a implementação do método **excluir**, efetuando a chamada para **setCarregado**, com valor **false**, apenas ao **final** da execução do método **excluir** do controlador, o que é viabilizado pelo operador **then**
 - c. No mapeamento para **LinhaLivro**, adicionar o **index** no lambda, utilizando o valor no atributo **key** do componente que é repetido, pois deve ser um valor numérico, e agora o código do livro é textual
 - Altere o código-fonte de **LivroDados**, no arquivo **src/LivroDados.js**, de acordo com os passos seguintes:
 - a. Ao nível do método **incluir**, alterar o **construtor** do livro, utilizando um **texto vazio** para o código
 - b. No mesmo método, efetuar a chamada para **navigate**, direcionando para a raiz,

apenas ao **final** da execução do método incluir do controlador, o que é viabilizado pelo operador **then**

- Com o projeto livro-servidor **em execução**, iniciar a execução da nova versão de **livros-react** com **npm start**, e testar as funcionalidades através de um navegador, acessando o endereço **http://localhost:3000**

- **Encerre** a execução da nova versão de **livros-react** e abra a cópia do projeto **livros-next** no Visual Studio Code
- Substitua os arquivos **Livro.ts** e **ControleLivros.ts** pelas versões modificadas na cópia do projeto **livros-react**, adequando o sistema baseado em Next JS ao novo modelo de chamadas para o servidor Express
- Altere o arquivo **pages/LivroLista.tsx**, de acordo com as instruções seguintes:
 - a. **Remover** a constante baseURL, com o endereço do servidor interno, e as funções assíncronas, de acesso à API interna
 - b. Adicionar uma instância de **ControleLivros**, com o nome **controleLivros**
 - c. Alterar a implementação de **useEffect**, adotando o modelo **assíncrono** na chamada para **obterTodos** de **controleLivros**, seguida do operador **then**, e consequente invocação de **setLivros** com uso do resultado
 - d. Alterar a **assinatura** do método **excluir**, adotando código do tipo **String**
 - e. Alterar a implementação do método **excluir**, efetuando a chamada para **setCarregado**, com valor **false**, apenas ao **final** da execução do método excluir de **controleLivros**, o que é viabilizado pelo operador **then**
 - f. No mapeamento para **LinhaLivro**, adicionar o **index** no lambda, utilizando o valor no atributo **key** do componente que é repetido, pois deve ser um valor numérico, e agora o código do livro é textual

Observação

É importante analisar a grande similaridade com o código utilizado na classe LivroLista, na nova versão de livros-react, diferenciando-se apenas pelo uso de tipos, devido à adoção do Type Script.

- Altere o arquivo **pages/LivroDados.tsx**, de acordo com as instruções seguintes:

- a. **Remover** a constante baseURL, com o endereço do servidor interno, e a função assíncrona, de acesso à API interna
- b. Adicionar uma instância de **ControleLivros**, com o nome **controleLivros**
- c. Ao nível do método **incluir**, alterar o **construtor** do livro, utilizando um **texto vazio** para o código
- d. No mesmo método, efetuar a chamada para o método **push** de **Router**, direcionando para **LivroLista**, apenas ao **final** da execução do método incluir de **controleLivros**, o que é viabilizado pelo operador **then**

Observação

É importante analisar a grande similaridade com o código utilizado na classe LivroDados, na nova versão de livros-react, diferenciando-se apenas pelo uso de tipos, devido à adoção do Type Script

- Com o projeto livro-servidor **em execução**, iniciar a execução da nova versão de **livros-next** via comando **npm run dev**, e testar as funcionalidades através de um navegador, acessando o endereço **http://localhost:3000**

- Abra a cópia do projeto **livros-angular** no Visual Studio Code, sem a necessidade de parar a execução de livros-next, já que usa uma porta diferente;
- Altere a classe **Livro**, no arquivo **src/app/livro.ts**, modificando o tipo do atributo **codigo** para **String**, e usando texto vazio para inicialização no construtor;
- Altere a classe **ControleLivrosService**, em **src/app/controle-livros.service.ts**, de acordo os passos seguintes:
 - a. Definir a constante global **baseURL**, recebendo o **endereço de base** do servidor Express, em "**http://localhost:3030/livros**"
 - b. Definir a **interface LivroMongo**, para compatibilizar o tipo de **Livro** às chamadas para o servidor, contendo os atributos apresentados a seguir:

- c. **Eliminar** o vetor **livros**, que funcionava como fonte de dados
- d. Alterar o método **obterLivros** para comportamento **assíncrono**, com uso de **fetch** no endereço de base, modo **GET**, recuperação da resposta como vetor **JSON**, e retorno com o **mapeamento** de cada elemento para um novo objeto do tipo **Livro**
- e. Alterar o método **excluir** para o comportamento **assíncrono**, recebendo **codigo**, do tipo **String**, chamando **baseURL/codigo** via **fetch**, no modo **DELETE**, e retornando o campo **ok** da resposta, indicando sucesso ou falha
- f. Alterar o método **incluir** para o comportamento **assíncrono**, adotando o parâmetro **livro**, do tipo **Livro**, com a conversão dele para uma interface **LivroMongo**, chamada para **baseURL** via **fetch**, no modo **POST**, incluindo o tipo de conteúdo como informação do header e a interface, que deve ser convertida para texto através de **JSON.stringify**, no corpo. O retorno da função será o campo **ok** da resposta, indicando sucesso ou falha

Observação

Os três métodos podem ser simplesmente **copiados** da nova versão de **ControleLivros**, na versão alterada de **livros-react**, pois não há nenhuma diferença em termos de código-fonte.

- Altere o código-fonte de **LivroListaComponent**, que foi definido no arquivo **src/app/livro-lista/livro-lista.component.ts**, de acordo com as instruções que são apresentadas a seguir:
 - a. Alterar a implementação do método **ngOnInit**, para utilizar o modelo **assíncrono** na chamada para **obterTodos** do controlador, e consequente atribuição do resultado ao vetor **livros**, sequenciada pelo operador **then**
 - b. Alterar a assinatura do método **excluir**, usando **codigo** como **String**
 - c. Alterar a implementação do método **excluir**, efetuando a nova chamada **assíncrona** para **obterTodos** apenas ao **final** da execução de excluir do **controlador**, o que é viabilizado pelo operador **then**
- Altere o código-fonte de **LivroDadosComponent**, que foi definido no arquivo **src/app/livro-dados/livro-dados.component.ts**, de acordo com as instruções que são apresentadas a seguir:
 - a. Ao nível do método **incluir**, efetuar a chamada para **navigateByUrl**, do **router**, direcionando para "**/lista**", apenas ao **término** da execução do método **incluir** do **controlador**, o que é viabilizado pelo operador **then**
- Com o projeto **livro-servidor** **em execução**, inicie a execução da nova versão de **livros-angular** via comando **ng serve**, e teste as funcionalidades através de um navegador, acessando o endereço **http://localhost:4200**

Resultados esperados da prática

- É importante que o código seja organizado;
- Explore as funcionalidades do Visual Studio Code para facilitar o desenvolvimento da prática;
- Nessa missão, é esperado que você demonstre habilidades básicas para:
 - a. construção de servidores com base em Express e Mongoose
 - b. integrar diferentes tipos de front-end ao back-end baseado em protocolo HTTP

Entrega da prática

Siga as etapas abaixo para **realizar a entrega desta missão**:

- Armazene o seu projeto em um repositório no **GITHUB**;
- Compartilhe o link com o seu tutor para correção da prática, por meio da **Sala de Aula Virtual**, na aba "**Trabalhos**" do respectivo nível de conhecimento.

👉 **Ei, não se esqueça de entregar este trabalho na data estipulada no calendário acadêmico!**

Feito com o Microsoft Sway

Crie e compartilhe apresentações, histórias e relatórios interativos e muito mais.

[Introdução](#)



