Integrando Frontend com Backend

UNISANTA

Índice

Modificações previstas

Agora vamos modificar nosso App Frontend para acessar os serviços disponíveis em nossa API desenvolvida para rodar no Backend.

Basicamente vamos alterar as operações de manipulação do array por requisições HTTP que acessam os endpoints da API. Dessa forma, nosso Frontend será capaz de utilizar dados armazenados no Banco De Dados armazenado na nuvem.

Preparando a comunicação

Agora vamos modificar nosso Frontend para conseguir emitir requisições HTTP para a API rodando no Backend.

Importando o módulo HttpClient

Modifique o arquivo "**/scr/app/app.module.ts**" inserindo o código destacado em verde abaixo:

```
import { NgModule } from '@angular/core';
import { BrowserModule } from '@angular/platform-browser';
import { HttpClientModule } from '@angular/common/http';

import { AppComponent } from './app.component';
import { ItemComponent } from './item/item.component';

@NgModule({
  declarations: [
    AppComponent,
    ItemComponent
  ],
  imports: [
    BrowserModule,
    HttpClientModule
  ],
  providers: [],
  bootstrap: [AppComponent]
})
export class AppModule { }
```

Obs: Cuidado para não esquecer de acrescentar a vírgula após a declaração **BrowserModule**. Perceba que ela também está destacada em verde.

Modifique o arquivo "**app.component.ts**" inserindo o código destacado em verde abaixo:

```
import { Component } from '@angular/core';
import { Tarefa } from "./tarefa";
import { HttpClient } from '@angular/common/http';

@Component({
  selector: 'app-root',
  templateUrl: './app.component.html',
  styleUrls: ['./app.component.css']
})
export class AppComponent {
  title = 'TODOapp';

  arrayDeTarefas: Tarefa[] = [];
  apiURL : string;

  constructor(private http: HttpClient) {
    this.apiURL = 'http://localhost:3000';
    this.READ_tarefas();
  }

  CREATE_tarefa(descricaoNovaTarefa: string) {
    var novaTarefa = new Tarefa(descricaoNovaTarefa, false);
    this.arrayDeTarefas.unshift(novaTarefa);
  }

  READ_tarefas() {
    this.arrayDeTarefas = [
      new Tarefa("Estudar Frameworks WEB", false),
      new Tarefa("Comer Pizza", false),
      new Tarefa("Ajudar meus pais", false)
    ];
  }

  DELETE_tarefa(tarefaAserRemovida: Tarefa) {
    this.arrayDeTarefas.splice(this.arrayDeTarefas.indexOf(tarefaAserRemovida), 1);
  }
}
```

Seu projeto agora está pronto para realizar requisições HTTP.

Executando o Backend

Acesse a pasta do Backend e execute o servidor utilizando o seguinte comando:

```
nodemon index "sua URL de conexão com o BD"
```

Executando o Frontend

Acesse a pasta do Frontend e execute o programa utilizando o seguinte comando:

```
ng serve
```

Lendo os dados a partir da API

Agora vamos fazer nosso App acessar a API para ler as tarefas armazenadas em nosso Banco de Dados. Para isso vamos enviar uma requisição HTTP com o método GET para o endpoint "**/api/getAll**"

Substitua o **corpo** do método **READ_tarefas** dentro de **app.component.ts** pelo seguinte conteúdo destacado em verde:

```
READ_tarefas() {
    this.http.get<Tarefa[]>(`${this.apiURL}/api/getAll`).subscribe(
      resultado => this.arrayDeTarefas=resultado);
}
```

Perceba que agora o método READ_tarefas alimenta o array com os dados obtidos a partir da API, ao contrário de popular o array com tarefas fixas.

Testando no Browser

Abra o browser utilizando o endereço:

http://localhost:4200/

Se tudo estiver correto, você verá que as tarefas obtidas são aquelas que estão armazenadas em seu Banco de Dados armazenado na nuvem.

Salvando dados a partir da API

Agora vamos modificar nosso App para acessar a API para gravar novas Tarefas em nosso Bando de Dados. Para isso vamos enviar uma requisição HTTP com o método POST para o endpoint "**/api/post**" enviando a nova tarefa como parâmetro.

Modifique o **corpo** do método **CREATE_tarefa** dentro de **app.component.ts,** conforme o novo conteúdo abaixo. Basicamente você vai substituir a linha **this.arrayDeTarefas.unshift(novaTarefa);** pelo código em verde abaixo:

```
CREATE_tarefa(descricaoNovaTarefa: string) {
    var novaTarefa = new Tarefa(descricaoNovaTarefa, false);
    this.http.post<Tarefa>(`${this.apiURL}/api/post`, novaTarefa).subscribe(
      resultado => { console.log(resultado); this.READ_tarefas(); });
}
```

Testando no Browser

Abra o browser utilizando o endereço:

http://localhost:4200/

Experimente adicionar uma nova tarefa à lista de tarefas existentes.

Acesse sua conta do MongoDB Atlas e constate que a nova tarefa foi gravada com sucesso.

Removendo dados a partir da API

Agora vamos modificar nosso App para acessar a API para remover Tarefas em nosso Bando de Dados. Para isso vamos enviar uma requisição HTTP com o método DELETE para o endpoint "**/api/delete/id**" onde o id especifica qual item desejamos remover do banco de dados. Considerando a necessidade de utilização do parâmetro "id" a partir deste momento, então vamos também adicioná-lo como atributo da classe Tarefa, modificando o arquivo "**/src/app/tarefa.ts**" conforme destacado em verde abaixo:

```
export class Tarefa {
    _id : string | undefined ;
    descricao: string;
    statusRealizada: boolean;

    constructor(_descricao: string, _statusRealizada: boolean) {
        this.descricao = _descricao;
        this.statusRealizada = _statusRealizada;
    }
}
```

Substitua o **corpo** do método **DELETE_tarefa** dentro de **app.component.ts** pelo seguinte conteúdo destacado em verde:

```
DELETE_tarefa(tarefaAserRemovida: Tarefa) {
    var indice = this.arrayDeTarefas.indexOf(tarefaAserRemovida);
    var id = this.arrayDeTarefas[indice]._id;
    this.http.delete<Tarefa>(`${this.apiURL}/api/delete/${id}`).subscribe(
      resultado => { console.log(resultado); this.READ_tarefas(); });
  }
```

Testando no Browser

Abra o browser utilizando o endereço:

http://localhost:4200/

Experimente remover uma tarefa existente

Alterando dados a partir da API

Agora vamos modificar nosso App para acessar a API para alterar Tarefas em nosso Bando de Dados. Para isso vamos enviar uma requisição HTTP com o método PATCH para o endpoint "**/api/update/id**" enviando a tarefa modificada como parâmetro. Lembrando que o parâmetro o "id" especificará qual item desejamos alterar no banco de dados.

Esta alteração será um pouco maior, pois o App original fazia as alterações diretamente a partir da referência dos objetos armazenados no array. Como agora os dados precisam ser atualizados no Bando de Dados, precisaremos adicionar um método de UPDATE em **AppComponent** e executá-lo a partir das ações realizadas em **ItemComponent**.

Para isso, adicione o método **UPDATE_tarefa** abaixo dentro de **app.component.ts**.

```
UPDATE_tarefa(tarefaAserModificada: Tarefa) {
    var indice = this.arrayDeTarefas.indexOf(tarefaAserModificada);
    var id = this.arrayDeTarefas[indice]._id;
    this.http.patch<Tarefa>(`${this.apiURL}/api/update/${id}`,
      tarefaAserModificada).subscribe(
        resultado => { console.log(resultado); this.READ_tarefas(); });
}
```

Agora vamos adicionar um atributo para armazenar um objeto com o nome **modificaTarefa** do tipo **EventEmitter** dentro de **item.component.ts,** conforme exemplo destacado em verde abaixo:

```
...
export class ItemComponent {
  emEdicao = false;
  @Input() tarefa: Tarefa = new Tarefa("", false);
  @Output() removeTarefa = new EventEmitter();
  @Output() modificaTarefa = new EventEmitter();
}
```

Agora, vamos emitir esse evento nas duas possíveis situações de alteração de dados da tarefa, ou seja:

- Quando o usuário altera o status da tarefa
- Quando o usuário altera a descrição da tarefa

Para isso, seguem destacadas em verde, as alterações que devem ser realizadas em **item.component.html**:

```
<div class="item">
    <input [id]="tarefa.descricao" type="checkbox" [checked]="tarefa.statusRealizada"
        (change)="tarefa.statusRealizada = !tarefa.statusRealizada; modificaTarefa.emit()" />
    <label [for]="tarefa.descricao">{{tarefa.descricao}}</label>

    <div *ngIf="!emEdicao" class="btn-wrapper">
        <button class="btn" (click)="emEdicao = true ">Editar</button>
        <button class="btn btn-warn" (click)="removeTarefa.emit()">Remover</button>
    </div>
    <div *ngIf="emEdicao">
        <input class="sm-text-input" #editedItem placeholder="escreva o novo nome da tarefa
aqui"
            [value]="tarefa.descricao">
        <div class="btn-wrapper">
            <button class="btn" (click)="emEdicao = false ">Cancelar</button>
            <button class="btn btn-save" (click)="tarefa.descricao=editedItem.value; emEdicao
= false; modificaTarefa.emit()">Salvar</button>
        </div>
    </div>
</div>
```

Por fim, temos que solicitar a execução do método **UPDATE_tarefa** assim que **AppComponent** detectar a ocorrência do evento "**modificaTarefa**".

Para isso, adicione a modificação destacada abaixo em verde, no código de **app.component.html**:

```
<div class="main">
    <h1>Mozilla Dev - My To Do List</h1>
    <input class="lg-text-input" #campoNovoItem />
    <button class="btn-primary" (click)="CREATE_tarefa(campoNovoItem.value)">
        Adicionar Tarefa</button>
    <ul>
        <li *ngFor="let tarefaDoLoop of arrayDeTarefas">
            <app-item [tarefa]="tarefaDoLoop" (removeTarefa)="DELETE_tarefa(tarefaDoLoop)"
(modificaTarefa)="UPDATE_tarefa(tarefaDoLoop)"></app-item>
        </li>
    </ul>
</div>
```

Testando no Browser

Abra o browser utilizando o endereço:

http://localhost:4200/

Experimente alterar o status e a descrição de uma tarefa existente.

Histórico de revisões

Revisão: 00
Data: 19/07/2022
Descrição das alterações:
Documento original