Frameworks WEB

Acessando uma REST API através do Angular

Sistemas de Informação - UNISANTA

Índice

Acessando uma REST API....3
Introdução....3
Testando o acesso à API através do Browser....4
Testando o acesso à API através do Postman - GET....5
Criando novas tarefas através do Postman - POST....6
Eliminando uma tarefa com o Postman - DELETE....6
Alterando dados de uma tarefa existente - PATCH....7
Acessando a REST API através de um app Angular....8
Criando o app Angular....8
Preparar o APP para se comunicar com o servidor....8
Executando o método GET....9
Exercício....11
Resultado desejado....11
Método para realização da requisição POST....11
Método para a realização da requisição DELETE....11
Método para a realização da requisição PATCH....11
Histórico de revisões....12

Acessando uma REST API

Introdução

Links de referência para esta introdução:

https://pt.stackoverflow.com/questions/45783/o-que-%c3%a9-rest-e-restful

https://www.devmedia.com.br/angular-http-como-realizar-requisicoes-em-suas-aplicacoes/40642

Para a exploração deste conteúdo, vamos utilizar uma API REST que será disponibilizada através de um servidor configurado por meio de uma VM Linux disponível no repositório da matéria. O endereço de acesso à API no servidor será:

```
127.0.0.1:3000
```

Essa API permitirá a realização de CRUD em um Banco de Dados para armazenamento das tarefas de nosso sistema TODO. Segue tabela com a especificação dos endpoints disponíveis em nossa API:

Endpoint	Método HTTP	Objetivo
/api/getAll	GET	Obter todas as tarefas (itens)
/api/getOne/id	GET	Obtém uma tarefa identificada pelo id
/api/post	POST	Grava uma nova tarefa
/api/update/id	PATCH	Atualiza a tarefa identificada pelo id
/api/delete/id	DELETE	Elimina a tarefa identificada pelo id

Testando o acesso à API através do Browser

Considerando que a API está disponível a partir da VM, podendo ser acessada através do endereço 127.0.0.1:3000, e que, os browsers utilizam o método GET para carregar uma página WEB, vamos então utilizar a seguinte URL a partir de um browser para acessar o primeiro endpoint de nossa API:

`http://127.0.0.1:3000/api/getAll`

Deve aparecer o seguinte resultado:

De forma análoga ao **getAll**, podemo solicitar uma única tarefa a partir do seu id. Ex:

`http://127.0.0.1:3000/api/getOne/6228fe621ee63c02950811ba`

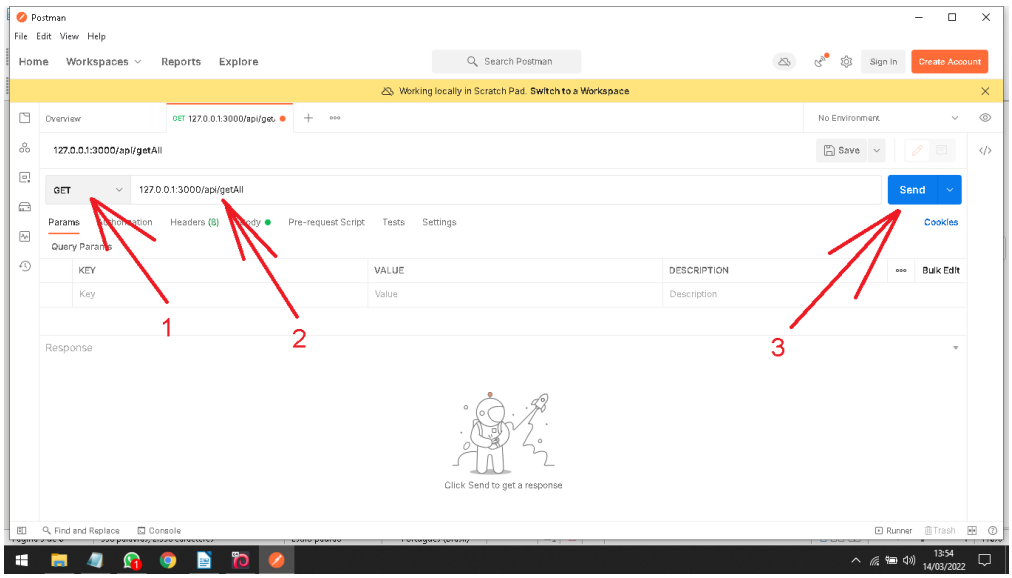
Aparecendo o seguinte resultado:

Testando o acesso à API através do Postman - GET

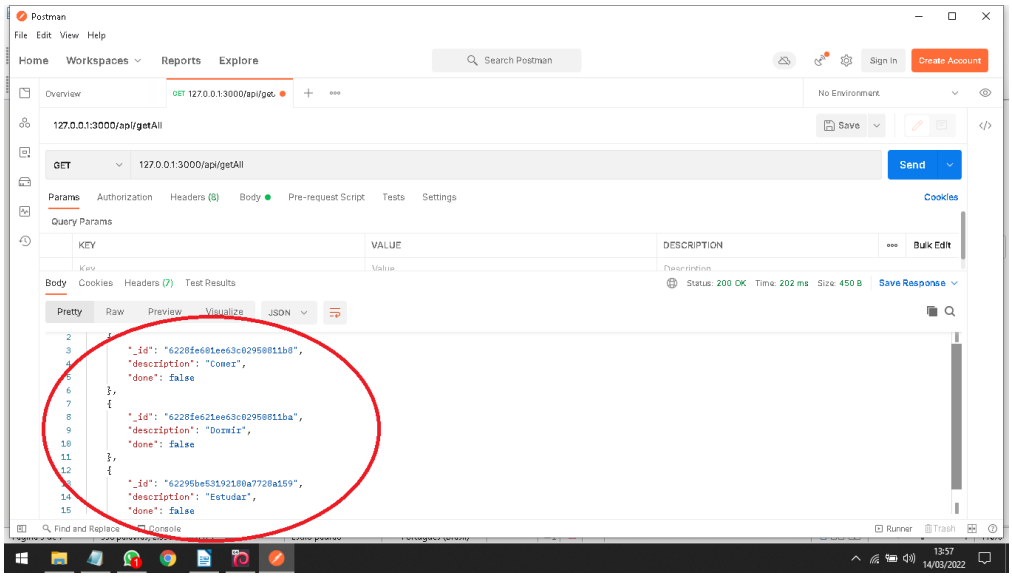
Para testarmos os outros métodos como POST, PATCH e DELETE, vamos fazer uso de uma ferramente chamada de Postman, que pode ser baixada a partir de:

`https://www.postman.com/downloads/`

Dentro do Postman, selecione o método "GET", insira a URL para ter todas as tarefas e clique em SEND:

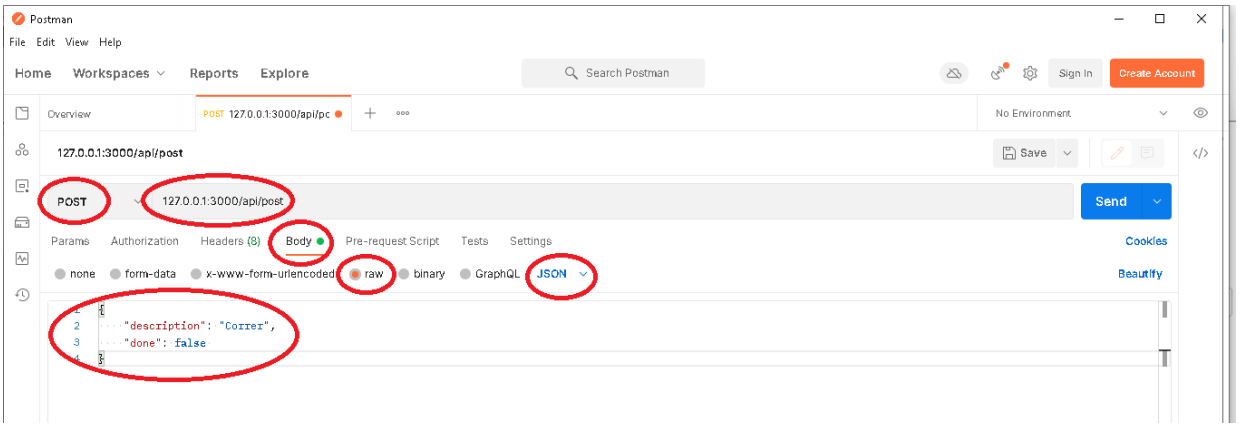


Você deve obter as seguintes respostas:



Criando novas tarefas através do Postman - POST

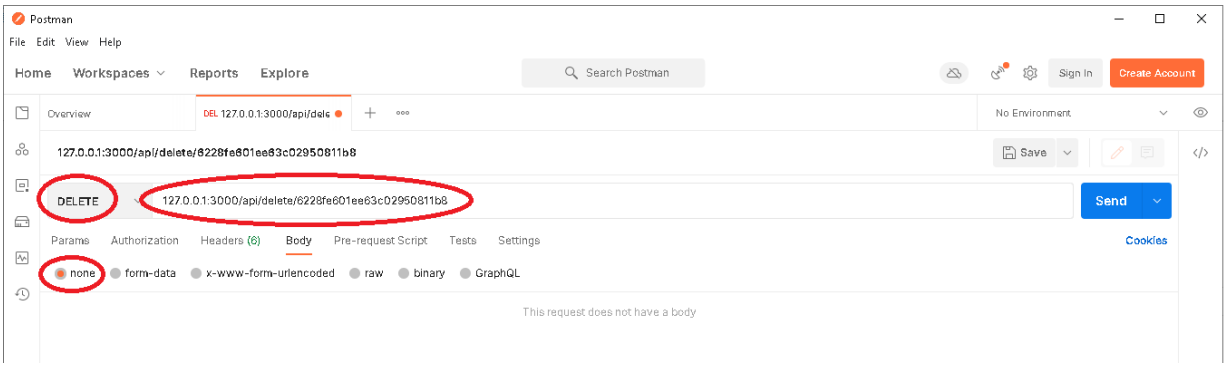
Considerando que o Postman facilita a utilização de outros métodos, vamos agora gravar uma nova tarefa no banco de dados, utilizando o método "POST" com o endpoint "/api/post" conforme exemplo abaixo (configurar corretamente os parâmetros em destaque):



Realize novamente uma leitura de todas as tarefas para verificar se a nova tarefa foi criada com sucesso.

Eliminando uma tarefa com o Postman - DELETE

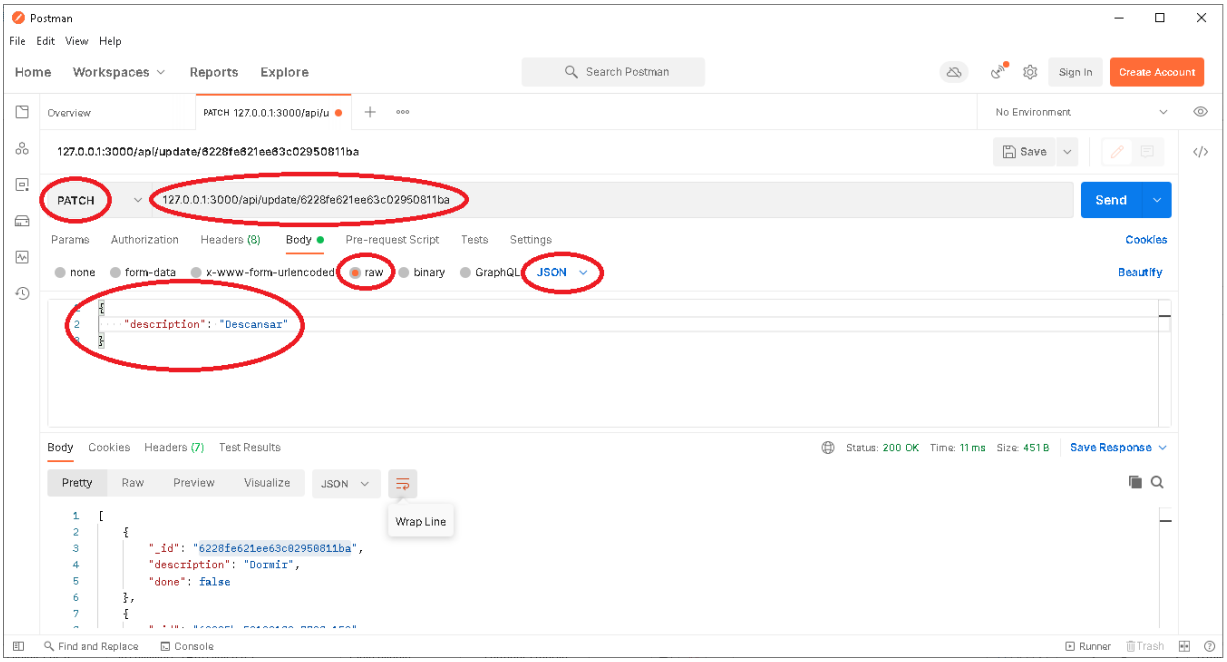
Agora vamos eliminar uma tarefa através do método "DELETE" utilizando o id da tarefa "Comer" (6228fe601ee63c02950811b8). Ex:



Realize novamente uma leitura de todas as tarefas para verificar se a tarefa foi eliminada com sucesso.

Alterando dados de uma tarefa existente - PATCH

Por fim, vamos agora alterar alguma informação de uma tarefa existente. Para isso usaremos o método "PATCH" informando o endpoint e parâmetros corretos. Ex:



Perceba que somente o atributo "description" foi alterado. Realize novamente uma leitura de todas as tarefas para verificar se a descrição da tarefa com o id informado (originalmente configurada com a descrição "Dormir") foi alterada com sucesso.

Acessando a REST API através de um app Angular

Considerando que conseguimos testar nossa REST API com sucesso, vamos agora desenvolver um app Angular capaz de realizar a comunicação com API.

Para isso, vamos fazer uso da classe **HttpClient**, disponível no módulo **HttpClientModule**, ou seja, um mecanismo nativo desse framework que oferece uma interface simplificada para realizar essa tarefa.

Criando o app Angular

```
ng new RestApiAngularHttpClient --routing=false --style=css
```

Preparar o APP para se comunicar com o servidor

Inserir a seguinte linha em **app.module.ts**:

```
import { HttpClientModule } from '@angular/common/http';
```

Acrescentar a seguinte linha dentro da sessão **imports** de **app.module.ts**:

```
HttpClientModule
```

A linha acima permitirá que o Angular consiga injetar uma dependência no momento que precisarmos utilizar um objeto do tipo **HttpClient**

Inserir o seguinte linha em **app.component.ts**:

```
import { HttpClient } from '@angular/common/http';
```

Criar um arquivo chamado **proxy.conf.json** na raíz do projeto com o seguinte conteúdo:

```
{
  "/api/*": {
    "target": "http://localhost:3000",
    "secure": false,
    "logLevel": "debug",
    "changeOrigin": true
      }
}
```

Inserir o seguinte construtor dentro da classe de **app.component.ts**:

```
constructor(private http: HttpClient) {}
```

Executando o método GET

Adicione o seguinte método em **app.component.ts**:

```
listarTodosItens() {
    this.http.get(`/api/getAll`).subscribe(resultado => console.log(resultado));
}
```

Defina o seguinte template em **app.component.html**:

```
<button (click)="listarTodosItens()">GET</button>
```

Inicialize o servidor do ambiente de desenvolvimento do app Angular através do seguinte comando:

```
ng serve --proxy-config proxy.conf.json
```

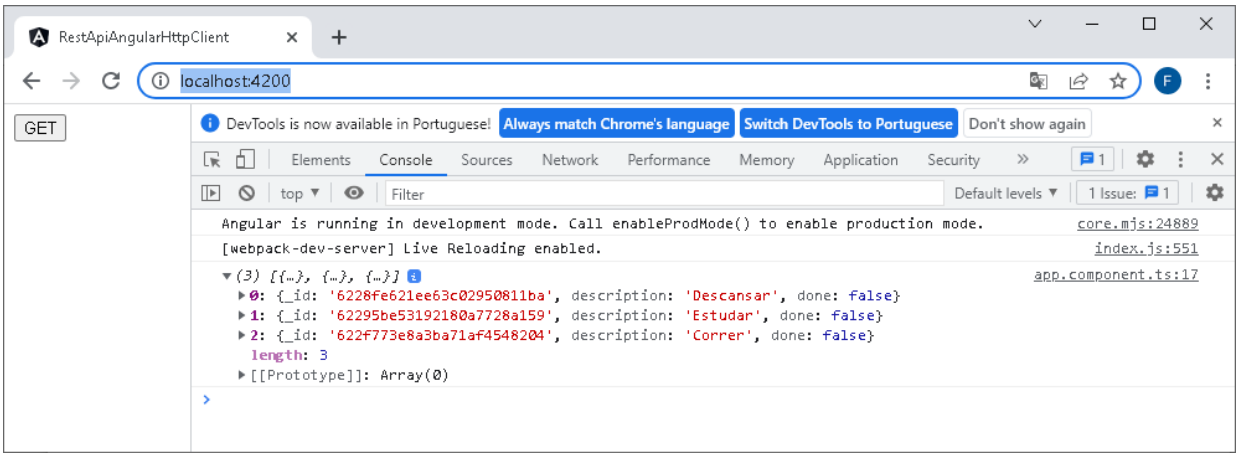
Acesse o app a partir do browser:

```
http://localhost:4200/
```

Aparecerá a seguinte aplicação:

Pressione F12 para apresentar o DevTools (Ferramentas do programador) e clique no botão "GET".

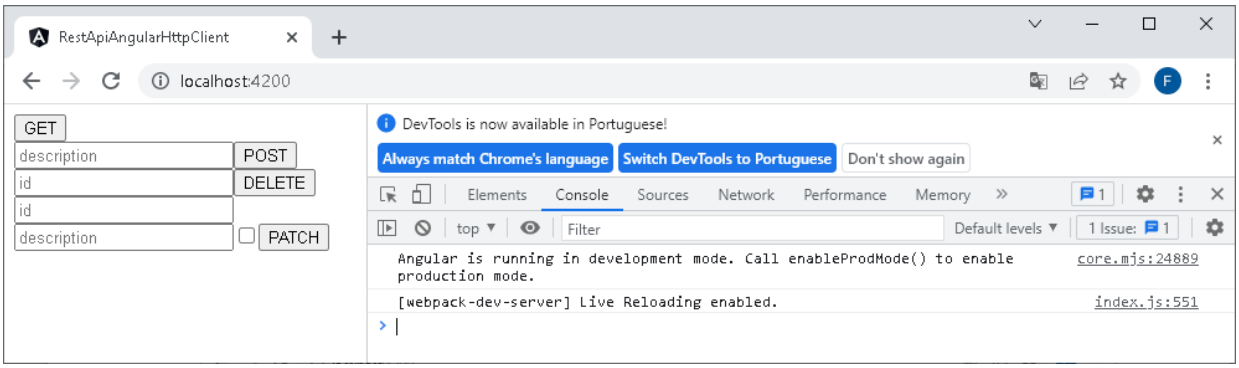
O console apresentará as tarefas que foram lidas a partir da REST API.



Exercício

Complete o app Angular, implementando as chamadas dos métodos para realizar as requisições do tipo POST, DELETE e PATCH.

Resultado desejado



Método para realização da requisição POST

```
adicionarItem(description: string) {
 var item = { description, done: false};
 this.http.post(`/api/post`,item).subscribe(resultado => {console.log(resultado)});
}
```

Método para a realização da requisição DELETE

```
removerItem(id: string) {
 this.http.delete(`/api/delete/${id}`).subscribe(resultado => {console.log(resultado)});
}
```

Método para a realização da requisição PATCH

```
atualizarItem(id: string, description: string, done: boolean) {
 var item = {description, done};
 this.http.patch(`/api/update/${id}`,item).subscribe(resultado => {console.log(resultado)});
}
```

Histórico de revisões

Revisão: 00
Data: 14/03/2022
Descrição das alterações:
Documento original