Criando as rotas da aplicação front-end.

Nesta aula, vamos criar rotas da aplicação front-end.

Vamos para o arquivo routes.js

A primeira rota que criaremos é a rota “/”. O usuário entra no sistema e será direcionado para uma mensagem simples de seja bem vindo.

```
router.get("/",(req, res) =>{
    res.send("Seja bem Vindo ao nosso sistema de Tarefas.");
})
```

Testando a rota.

A próxima rota será a rota para acessar a view cadastrar da aplicação.

```
router.get("/tarefas/cadastrar",(req, res) =>{
  res.render("tarefas/cadastrar");
})
```

Executando nosso servidor.

Vamos começar a criar rotas que precisam de funções do arquivo services.

Vamos no arquivo services.js e faremos a criação de uma constante para usar a biblioteca axios e vamos iniciar a nossa classe.

```
const axios = require("axios");

module.exports = class Services{

}
```

A primeira função que iremos criar é a Create.

Vamos iniciar a função recebendo uma let chamada valores que receberá todos os dados do formulário de cadastro.

```
  static async TarefaCreate(req,res){
    let valores = req.body;
```

Próximo passo é criar um objeto para armazenar as informações que precisamos passar para o axios.

Sempre precisamos passar os seguintes comandos:

url: informa qual a url da nossa API.

method: informa qual o method usado na API(post, put, delete ou get).

data: guarda os valores que serão passados no formato JSON para a API.

Vamos ver como fica abaixo nosso código;

```
    const options = {
      url: 'https://listatarefas.antonioizo.repl.co/tarefas/Cadastrar',
      method: 'POST',
      data: valores
    };
```

Repare que a url é o caminho referente a rota de cadastro de tarefas da nossa API.

Coloquei o caminho da minha API, você deve incluir o seu caminho.

Depois de configurar o objeto que será passado para o axios, vamos chama-lo.

```
axios(options);
    const mensagem = "Cadastro realizado com sucesso!";
    res.render("mensagem",{mensagem});
  }
```

Nossa função create está pronta, criei uma const chamada mensagem e retornei para a página mensagem com um objeto para informar que o cadastro foi executado com sucesso.

Para testar, temos que ir a rota para a função que acabamos de criar e verificar no formulário de cadastro o nome da action.

```
1  <h1 class="text-center my-3">Cadastro de
   tarefas</h1>
2  <form action="/tarefas/Create"
   method="post">
3    <div class="row w-50 d-block m-auto g-
   3">
```

Ele está chamando o caminho “/tarefas/Create”, vamos criar uma rota com o mesmo nome no arquivo routes.js

```
//ROTA PARA SERVIÇO DE CREATE
router.post("/tarefas/Create",Services.TarefaCreate);
```

ATENÇÃO: Você precisa estar em outra aba com seu servidor da API rodando. O replit deixa o servidor de forma *online*, mas pode ser parado.

Agora podemos testar, ao executar o servidor você precisa chamar o /tarefas/cadastrar

Aparentemente está tudo certo.

Vamos criar nossa função para listar as tarefas da API.

```
//LISTAR
  static async TarefaListar(req,res){
    const options = {
      url: 'https://listatarefas.antonioizo.repl.co/tarefas',
      method: 'GET',
      data: {}
    };
    axios(options).then(response => {
      console.log(response.data);
      const tarefa =response.data

      res.render("tarefas/listar",{tarefa});
    });
  }
```

Houveram poucas mudanças da nossa função create, a principal é no objeto options que está vazio.

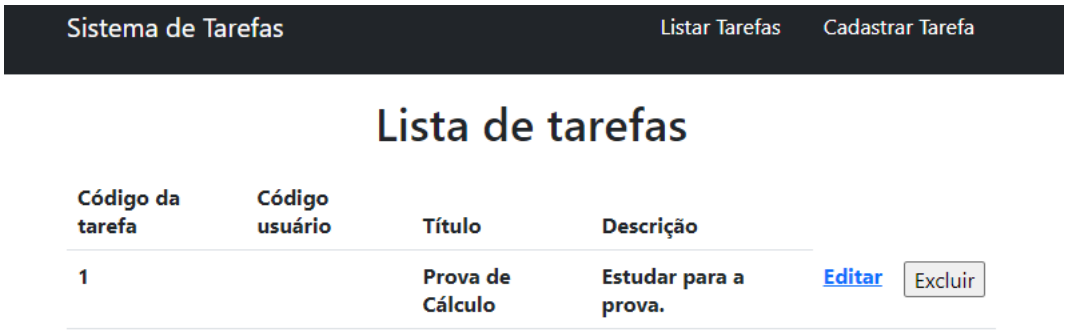
A ideia é receber valores em uma lista da API por isso ele está sem valores.

No final recebemos a lista e passamos para a constante tarefa que envia para a rota da página listar com um objeto.

Vamos incluir nossa rota no routes antes de testar.

```
//ROTA PARA O SERVIÇO LISTAR
router.get("/tarefas/listar",Services.TarefaListar);
```

Agora vamos testar chamando o caminho “/tarefas/listar”



Vamos criar agora nossa função update no services.js.

```
router.get("/tarefas/Atualizar/:id_tarefa/:titulo/:descricao",(req, res) =>{

  let tarefas = {
    id_tarefa : req.params.id_tarefa,
    titulo : req.params.titulo,
    descricao : req.params.descricao
  }
  res.render("tarefas/update",{tarefas});
})
```

A função update recebe uma let tarefas que cria um objeto com os valores recebidos via get, repare que nosso router.get recebe os parâmetros via url. Depois ele envia como objeto os dados para a página tarefas/update.

Vamos configurar nossa view listar para o caminho correto da nossa view.

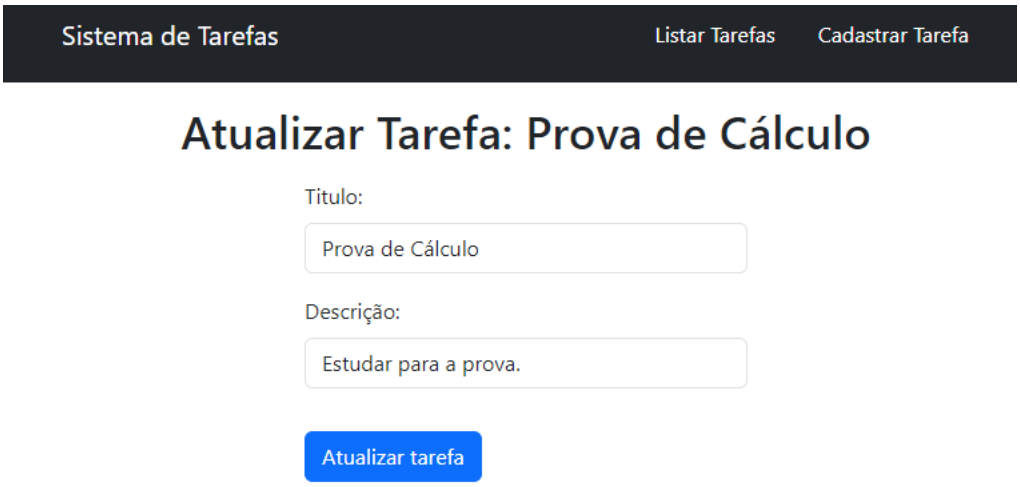
```
    <th>{{this.titulo}}</th>
    <th>{{this.descricao}}</th>
    <th><a href="/tarefas/alterar/{{this.id_tarefa}}">Edita
</i></a></th>
```

Altere o caminho de /tarefas/alterar para

"/tarefas/Atualizar/{{this.id_tarefa}}/{{this.titulo}}/{{this.descricao}}"

Vamos passar todos os valores via get.

Vamos chamar nossa rota tarefas/listar e tentar editar agora.



A nossa rota funcionou, agora precisamos terminar a rota de atualizar para enviar os dados atualizados para o banco.

Vamos criar a função de update no services.js.

```
//Update
  static async TarefaUpdate(req,res){

    let valores = req.body;
    const options = {
      url: 'https://listatarefas.antonioizo.repl.co/tarefas/'+valores.id_tarefa,
      method: 'PUT',
      data: valores
    };
    axios(options);
    const mensagem = "Registro atualizado com sucesso";
    res.render("mensagem",{mensagem});
  }
```

Recebemos um valor via body que são os dados do formulário que iremos atualizar.

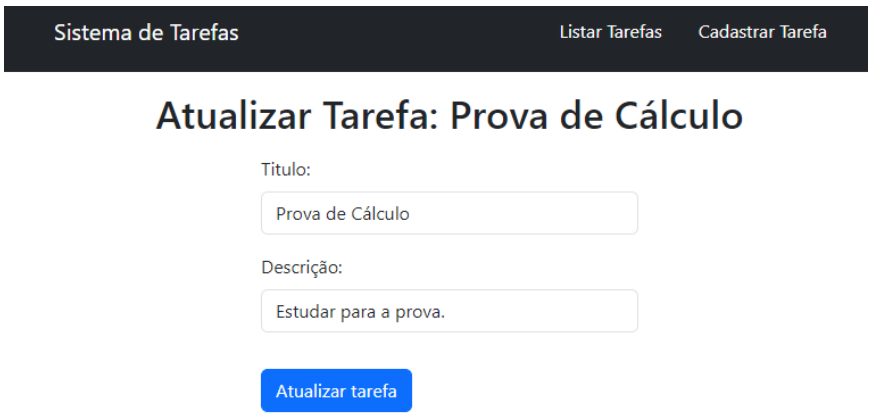
Chamamos a url da API referente a atualização de valores e enviamos qual o id da tarefa que deve ser atualizada.

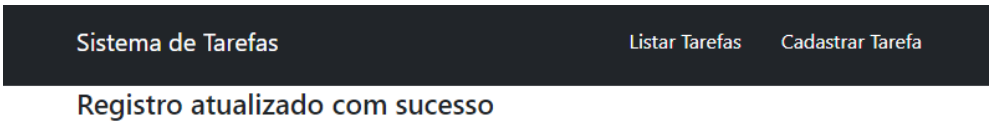
Depois passamos os valores pelo method PUT para a API.

Vamos agora criar nossa rota no arquivo routes.js.

```
router.post("/tarefas/Update",Services.TarefaUpdate);
```

Testando.





Vamos listar para ver se foi atualizado.

Vamos agora na função delete.

```
//Delete
  static async TarefaDelete(req,res){
    let id_tarefa = req.body.id_tarefa;
    const options = {
      url: 'https://listatarefas.antonioizo.repl.co/tarefas/'+id_tarefa,
      method: 'DELETE'
    };
    axios(options);
    const mensagem = "Tarefa excluída com sucesso!";
    res.render("mensagem",{mensagem});
  }
```

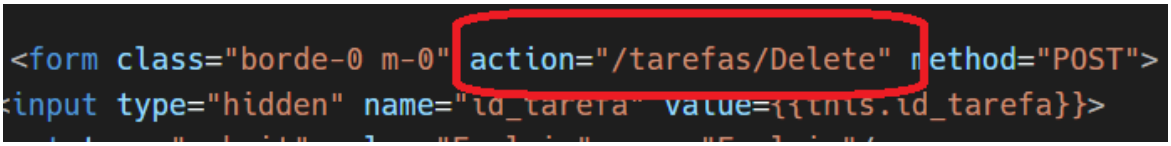
Pegamos o valor do id_tarefa e passamos no caminho da url da api.

Vamos incluir a rota agora.

```
//ROTA PARA O SERVIÇO DE DELETE
router.post("/tarefas/Delete",Services.TarefaDelete);
```

Vamos na view listar para configurar o action para nossa exclusão.

Vamos trocar /tarefas/remover por tarefas/Delete.



Vamos testar nosso delete.

Para finalizar, vamos no main.handlebars que está localizado dentro da pasta views/layouts e vamos alterar os caminhos da listar tarefas e cadastrar tarefas.

```
<li class="nav-item">
  <a class="nav-link text-white" href="/">Listar Tarefas</a>
</li>
<li class="nav-item">
  <a class="nav-link text-white" href="/cadastrar">Cadastrar Tarefa</a>
</li>
```

Vai ficar assim.

```
<li class="nav-item">
  <a class="nav-link text-white" href="/tarefas/listar">Listar Tarefas</a>
</li>
<li class="nav-item">
  <a class="nav-link text-white" href="/tarefas/cadastrar">Cadastrar Tarefa</a>
</li>
```

Pronto agora você já pode testar e melhorar o código que fizemos.

Criamos uma API e consumimos com os seus serviços.

Espero que tenha gostado das aulas da semana.

Até a nossa próxima.