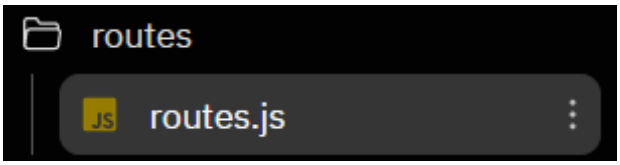
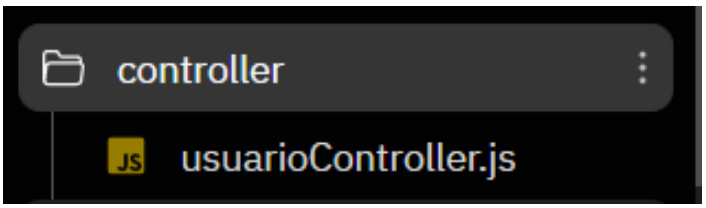
Criando o routes e o controller da API.

Nesta aula, vamos criar o routes e os controllers da API de um sistema de lista de tarefas que utilizaremos para exemplificar o uso de cookies.

Vamos começar criando uma pasta chamada routes e um arquivo chamado **routes.js** que cuidará das nossas rotas.



O arquivo **routes.js** será responsável por todas as rotas referente ao controllers.
Vamos agora criar uma pasta controller e um arquivo chamado **usuarioController.js**



O arquivo **usuarioController.js** armazenará todas as funções do nosso CRUD usuário.
Já vimos o arquivo controller de um CRUD, vou separar abaixo os principais pontos para relembrarmos cada código acrescentado.

Incluir o módulo **usuarioModel** e criar o **module.exports** para utilizar a função na API.

```
const Usuario = require("../model/usuarioModel");
module.exports = class usuarioController{
```

Função **UsuarioCreate** responsável pela criação do usuário no banco.

```
//CREATE
  static async UsuarioCreate(req,res){
    let nome = req.body.nome;
    let email = req.body.email;
    let senha = req.body.senha;

    const usuario = {
      nome: nome,
      email: email,
      senha: senha
    }
    await Usuario.create(usuario);
    res.json({message: "Usuário cadastrado com sucesso!"});
  }
```

Função **UsuarioListar** responsável pelo ***READ*** do usuário.

```
//READ - LISTAR
  static async UsuarioListar(req,res){
    const id_usuario = req.params.id;
    if(id_usuario){
      const usuario = await Usuario.findOne({where: {id_usuario: id_usuario}});
      res.json(usuario);
    }else{
      const usuario = await Usuario.findAll({raw:true});
      res.json(usuario);
    }
}
```

Função **UsuarioUpdate** responsável pela atualização dos campos na tabela usuário.

```
//UPDATE
  static async UsuarioUpdate(req, res){
    const id_usuario = req.params.id;
    let nome = req.body.nome;
    let email = req.body.email;
    let senha = req.body.senha;
    const usuario = {
      nome: nome,
      email: email,
      senha: senha
```

```
    };
    await Usuario.update(usuario,{where: {id_usuario:id_usuario}});
    res.json({message: "Cadastro atualizado com sucesso! Foram atualizados as
sequintes informações: ", dados: usuario});
  }
```

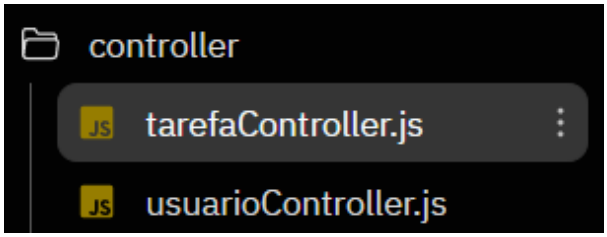
Função **UsuarioDelete** responsável pela exclusão do usuário.

```
  static async UsuarioDelete(req,res){
    const id_usuario = req.params.id;
    await Usuario.destroy({where:{id_usuario: id_usuario}});

    res.json({message: "Usuário excluído com sucesso!"});
  }
}//chave do fechamento da classe UsuarioController
```

Vamos criar agora, o arquivo **tarefaController.js** dentro da pasta controller.

O arquivo será responsável pela criação do CRUD do model tarefa.



O arquivo tarefaController.js é mostrado abaixo:

```
const Tarefa = require("../model/tarefaModel");

module.exports = class tarefaController{

//CREATE
  static async TarefaCreate(req,res){
    let titulo = req.body.titulo;
    let descricao = req.body.descricao;

    const tarefa = {
      titulo: titulo,
      descricao: descricao
    }
    await Tarefa.create(tarefa);
    res.json({message: "Tarefa cadastrada com sucesso!"});
```

```
  }
//READ - LISTAR
  static async TarefaListar(req,res){
          const tarefa = await Tarefa.findAll();
      res.json(tarefa);
  }

  //UPDATE
  static async TarefaUpdate(req, res){
    const id_tarefa = req.params.id;
    let titulo = req.body.titulo;
    let descricao = req.body.descricao;
    const tarefa = {
      titulo: titulo,
      descricao: descricao
    };
    await Tarefa.update(tarefa,{where: {id_tarefa:id_tarefa}});
    res.json({message: "Tarefa atualizada com sucesso! Foram atualizados as
sequintes informações: ", dados: tarefa});
  }

  static async TarefaDelete(req,res){
    const id_tarefa = req.params.id;
    await Tarefa.destroy({where:{id_tarefa: id_tarefa}});

    res.json({message: "Tarefa excluída com sucesso!"});
  }
}
```

Vamos criar agora nossas rotas no arquivo **routes.js**

O arquivo routes.js terá as rotas para as funções dos controllers usuário e tarefa. Colocaremos também a rota principal que será mostrada para nosso usuário quando acessar nossa API pelo “/”.

```
////////////////MÓDULOS //////////////////////
const express = require("express");
const router = express.Router();
```

```
////////////MODELS/////////////////
const usuarioController = require("../controller/usuarioController");
const tarefaController = require("../controller/tarefaController");

////////////////Requisições HTTP Principal ////////////////////
router.get("/",(req, res) =>{
  return res.json({message: "Sistema de Lista de Tarefas"});
})
////////////////Requisições HTTP Usuario ////////////////////

//POST - CADASTRAR
router.post("/usuarios/Cadastrar", usuarioController.UsuarioCreate);

//GET - LISTAR
router.get("/usuarios/:id?", usuarioController.UsuarioListar);

//PUT - UPDATE
router.put("/usuarios/:id", usuarioController.UsuarioUpdate);

// DELETE
router.delete("/usuarios/:id", usuarioController.UsuarioDelete);

////////////////Requisições HTTP Tarefa ////////////////////
//POST - CADASTRAR
router.post("/tarefas/Cadastrar",tarefaController.TarefaCreate);

//GET - LISTAR
router.get("/tarefas/:id?", tarefaController.TarefaListar);

//PUT - UPDATE
router.put("/tarefas/:id", tarefaController.TarefaUpdate);

// DELETE
router.delete("/tarefas/:id", tarefaController.TarefaDelete);

module.exports = router;
```

Vamos agora no arquivo index.js e terminar a configuração da API.

Temos que incluir o módulo do arquivo routes.js.

```
const routes = require("./routes/routes");
```

Vamos colocar também a informação que utilizaremos JSON com **urlencoded.**

```
app.use(express.urlencoded({ extended: true }));
app.use(express.json());
```

Vamos alterar a rota principal e informar que utilizaremos o modeulo routes.

```
//ROTA PRINCIPAL
app.use("/",routes);
```

O arquivo index.js completo está abaixo.

```
//BIBLIOTECAS/MODULOS UTILIZADOS
const database = require("./db/db");
const routes = require("./routes/routes");
const express = require("express");
const app = express();
//const usuarioController = require("./controller/usuarioController");
//CODIFICAÇÃO UTILIZAÇÃO DO JSON
app.use(express.urlencoded({ extended: true }));
app.use(express.json());
//MODELS
const Usuario = require("./model/usuarioModel");
const Tarefa = require("./model/tarefasModel");

//SINCRONISMO COM O BANCO DE DADOS
try {
    database.sync().then(() => {
    })
}
catch(erro) {
    console.log("Houve uma falha ao sincronizar com o banco de dados. ", erro);
};
//ROTA PRINCIPAL
app.use("/",routes);
app.listen(3000);
```

Na próxima aula, vamos criar uma aplicação **front-end** para consumir nossa API.

até a próxima aula.