02

Faça como eu fiz: Controllers e rotas

Parte das rotas que usaremos para teste dos outros modelos será muito similar à rota de Pessoas que fizemos anteriormente, e a mesma coisa vale para os controladores. Então vamos deixar estes arquivos disponíveis para você inserir no seu projeto. São as rotas de Niveis e Turmas. A parte das matrículas será diferente, então faremos separado no próximo vídeo.

Lembrando que a estrutura de pastas segue a do projeto, caso você tenha feito alguma modificação durante seus estudos, não esqueça de revisar os caminhos correspondentes.

Primeiro, vamos criar os arquivos de controladores para Niveis e Turmas.

- controllers/NivelController.js
- controllers/TurmaController.js

Os verbos HTTP e os métodos de classe que vamos usar serão os mesmos, só que agora eles se referem às Turmas e Niveis. Confira os nomes das classes e métodos que estão nos arquivos que deixamos prontos:

```
// controllers/TurmaController.js

class TurmaController {

static async pegaTodasAsTurmas(req, res) {
  try {
    const todasAsTurmas = await database.Turmas.findAll()
    return res.status(200).json(todasAsTurmas)
  } catch (error) {
    return res.status(500).json(error.message);
  }
}
//etc


//controllers/NivelController.js

class NivelController {

static async pegaTodosOsNiveis(req, res) {
  try {
    const todosOsNiveis = await database.Niveis.findAll()
    return res.status(200).json(todosOsNiveis)
  } catch (error) {
    return res.status(500).json(error.message);
  }
}
//etc
```

Agora vamos criar os arquivos de rotas referentes a esses dois modelos:

- routes/niveisRoute.js

- routes/turmasRoute.js

O conteúdo dos arquivos também está disponível na pasta `arquivos_base`. Como sempre, não esqueça de conferir se estão corretos os nomes dos arquivos, nomes dos métodos e caminhos (por exemplo, se o caminho em `require('../controllers/NivelController')` segue a mesma estrutura de pastas do seu projeto.

As rotas dos níveis ficarão dessa forma:

```
//routes/niveisRoute.js

const { Router } = require('express')
const NivelController = require('../controllers/NivelController')

const router = Router()
router
 .get('/niveis', NivelController.pegaTodosOsNiveis)
 .get('/niveis/:id', NivelController.pegaUmNivel)
 .post('/niveis', NivelController.criaNivel)
 .put('/niveis/:id', NivelController.atualizaNivel)
 .delete('/niveis/:id', NivelController.apagaNivel)
module.exports = router
```

E as de Turmas:

```
//routes/turmasRoute.js

const { Router } = require('express')
const TurmaController = require('../controllers/TurmaController')

const router = Router()
router
 .get('/turmas', TurmaController.pegaTodasAsTurmas)
 .get('/turmas/:id', TurmaController.pegaUmaTurma)
 .post('/turmas', TurmaController.criaTurma)
 .put('/turmas/:id', TurmaController.atualizaTurma)
 .delete('/turmas/:id', TurmaController.apagaTurma)
module.exports = router
```

Por último, precisamos avisar o routes/index.js que novas rotas foram adicionadas à aplicação. O arquivo final ficará da seguinte forma (você pode deletar os comentários!):

```
//routes/index.js

const bodyParser = require('body-parser')

const pessoas = require('./pessoasRoute')
//adicionamos as rotas de niveis e turmas
const niveis = require('./niveisRoute')
const turmas = require('./turmasRoute')

//adicionamos as instâncias de níveis e turmas
//e refatoramos um pouco a função
module.exports = app => {
```

```
app.use(
  bodyParser.json(),
  pessoas,
  niveis,
  turmas
  )
}
```

Não esqueça de testar suas rotas no Postman! Caso tenha dúvidas, só perguntar no fórum.