

Nome: Thales Dias Prudencio

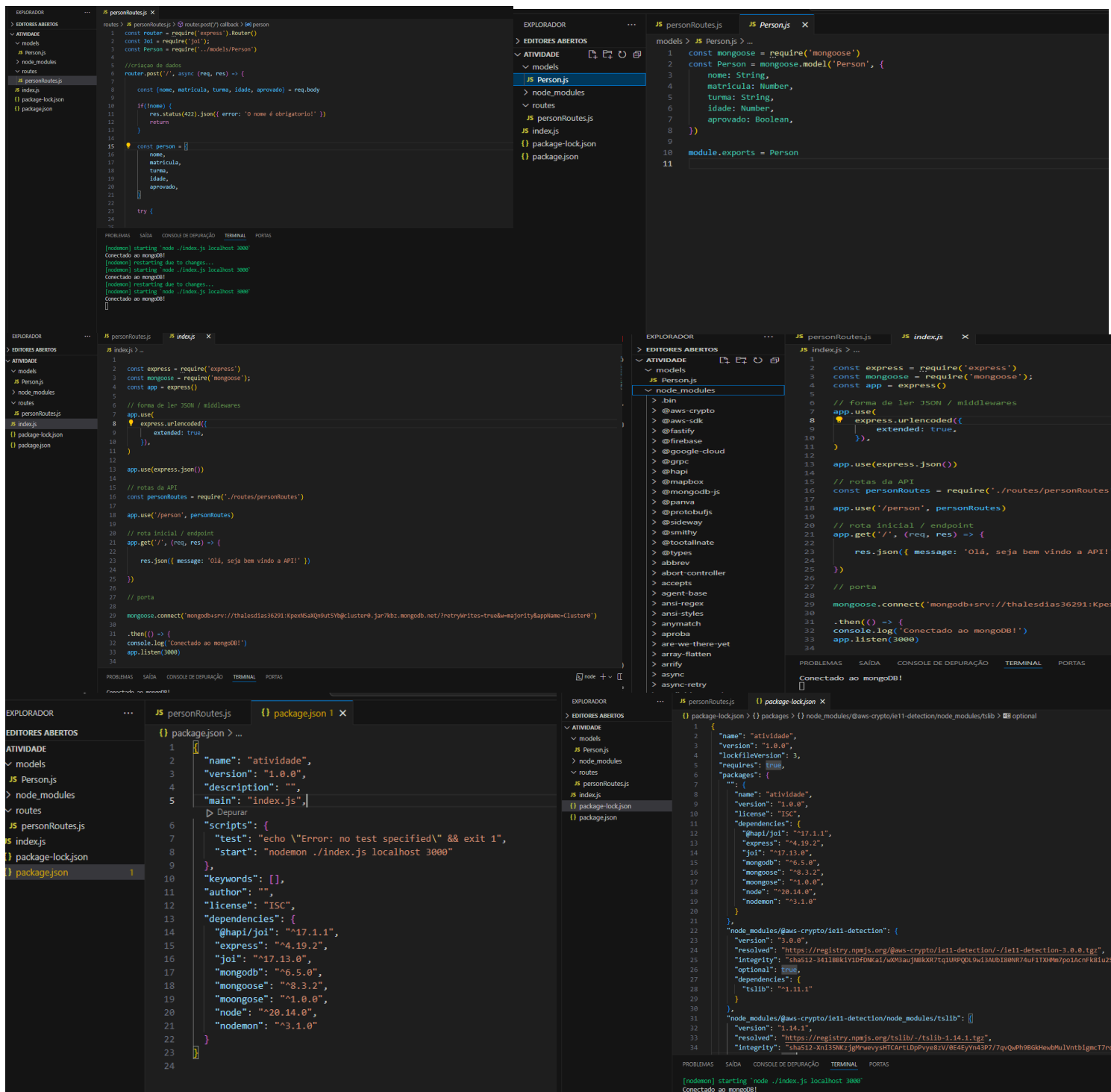
Matrícula: 202222101

Curso: Análise e desenvolvimento de sistemas

Tarefa: Desenvolva uma solução no formato RESTFul API.

Desenvolvi o projeto utilizando Node.js e MongoDB(atlas) - Node, Express e e Mongoose. Utilizei o postman para testar a funcionalidade da API. Os dashboards foram feitos diretamente pelo mongoDB atlas.

Abaixo algumas imagens do meu código no VS code:

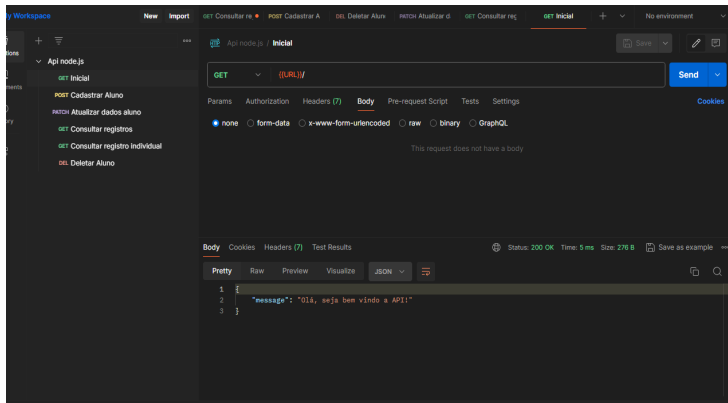


The collage displays three main parts of the project code in VS Code:

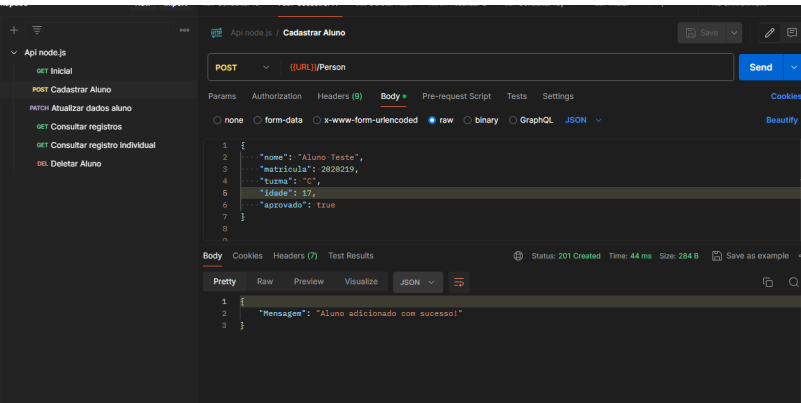
- personRoutes.js:** Contains a REST API for person management. It includes endpoints for creating a person (POST /person), retrieving a person by ID (GET /person/:id), and updating a person (PUT /person/:id). The code uses Express.js for routing and Mongoose for database operations.
- index.js:** The main entry point of the application. It sets up the Express app, applies middleware (express.json(), express.urlencoded()), and registers the personRoutes. It also connects to the MongoDB database using Mongoose and starts the server on port 3000.
- package.json:** The project's manifest file. It lists the project name as 'atividade', the version as '1.0.0', and the main script as 'index.js'. It also lists the dependencies: express, mongoose, and nodemon.

Imagens dos testes realizados na API pelo postman:

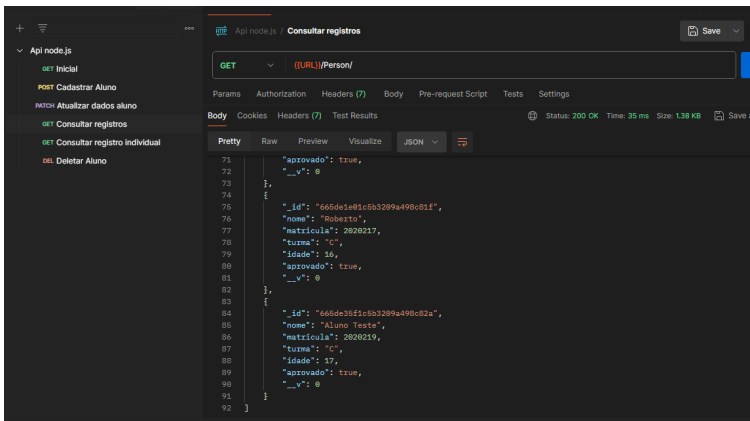
Tela inicial da API



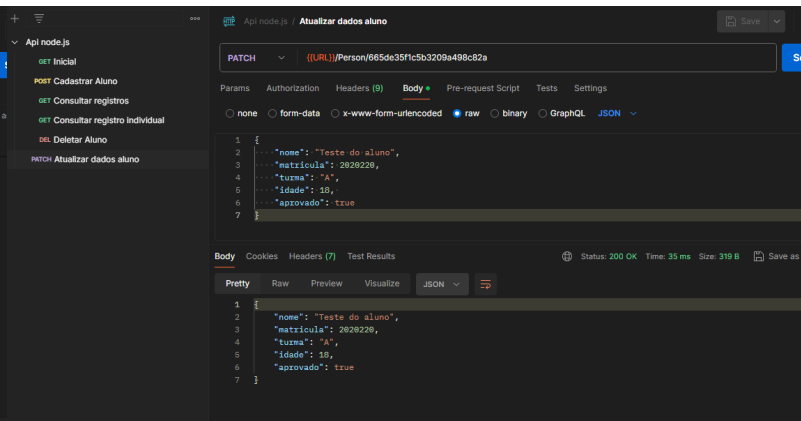
Tela de cadastrar novo aluno



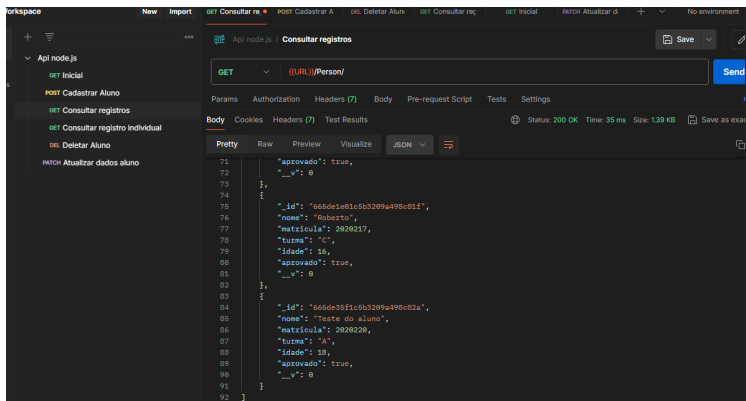
Tela de registros gerais



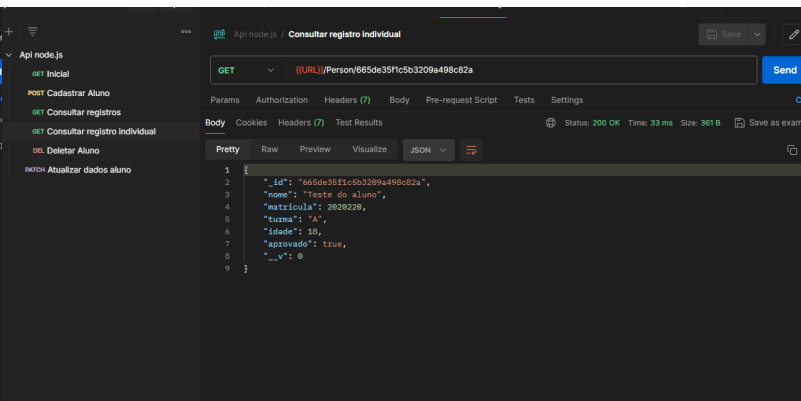
Atualização de dados(ID)



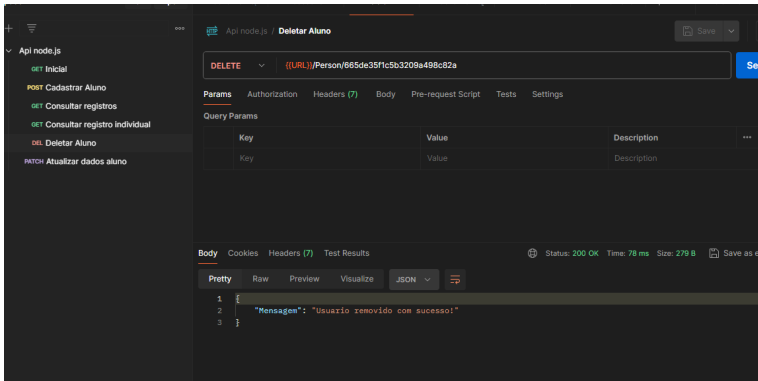
Registros após a atualização



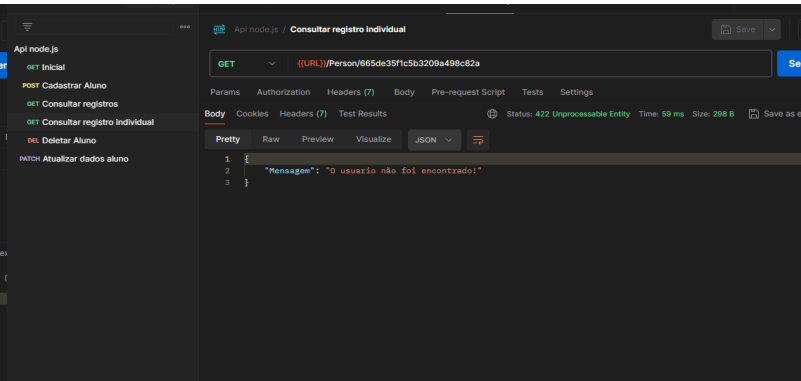
Registro individual (busca por ID)



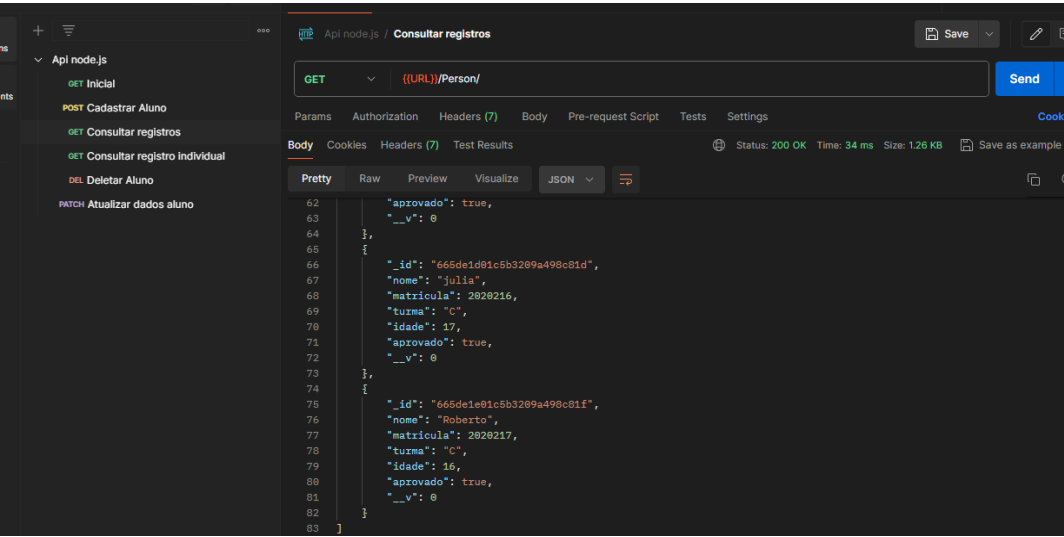
Remover Aluno(ID)



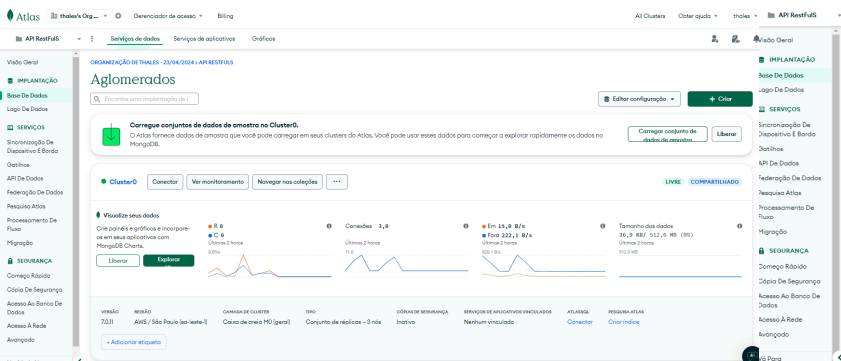
Aluno removido



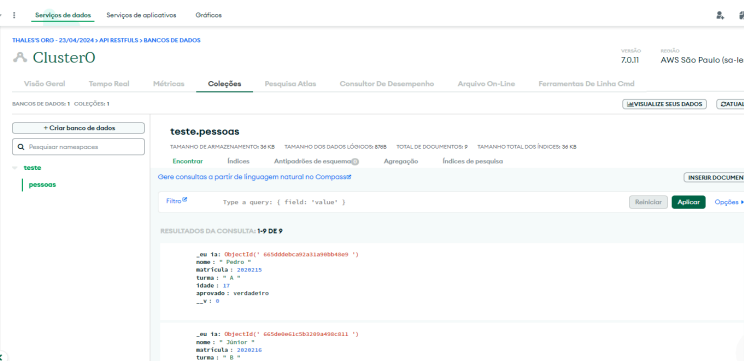
Registros após o aluno ser removido



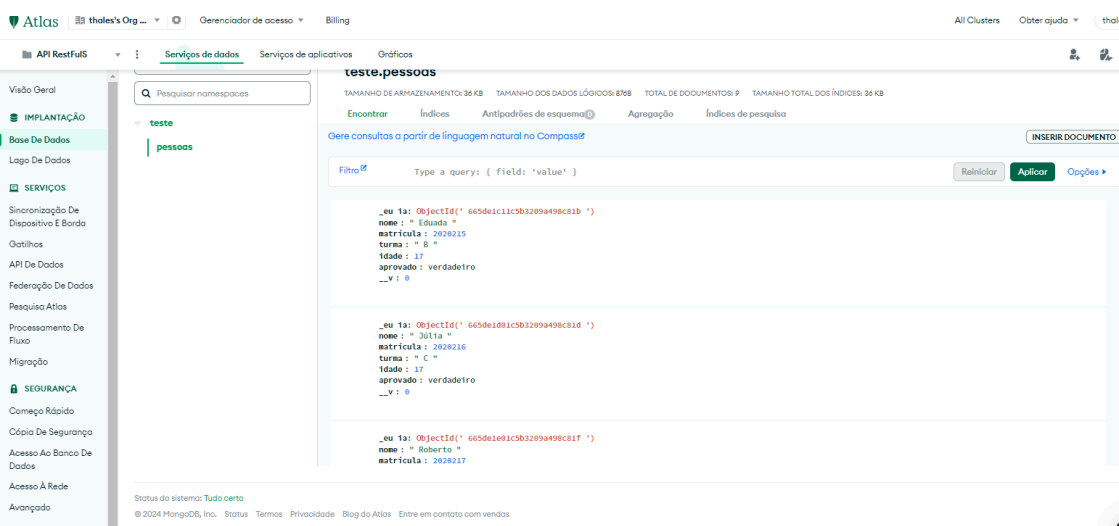
Banco de dados pelo Atlas:



Alunos cadastrados na API



Alunos cadastrados no banco de dados



Dashboards - Atlas



Código:

Person.js

```
const mongoose = require('mongoose')

const Person = mongoose.model('Person', {

  nome: String,

  matricula: Number,

  turma: String,

  idade: Number,

  aprovado: Boolean,

})

module.exports = Person
```

PersonRoutes.js

```
const router = require('express').Router()

const Joi = require('joi');

const Person = require('../models/Person')

//criação de dados

router.post('/', async (req, res) => {

  const {nome, matricula, turma, idade, aprovado} = req.body

  if(!nome) {

    res.status(422).json({ error: 'O nome é obrigatorio!' })
```

```
        return

    }

    const person = {

        nome,

        matricula,

        turma,

        idade,

        aprovado,

    }

    try {

const personSchema = Joi.object({

    nome: Joi.string().required(),

    matricula: Joi.number().required(),

    turma: Joi.string().required(),

    idade: Joi.number().required(),

    aprovado: Joi.boolean().required(),

});

    function validarDadosPerson(req, res, next) {

        const { error } = personSchema.validate(req.body);

        if (error) {
```

```
        return res.status(400).send(error.details[0].message);
    }

    next();
}

// criando dados

await Person.create(person)

res.status(201).json({ Mensagem: 'Aluno adicionado com
sucesso!' })

} catch (error) {

    res.status(500).json({error: error})

}

}))

//leitura de dados

router.get('/', async (req, res) => {

    try {

        const people = await Person.find()

        res.status(200).json(people)
```

```
    } catch (error) {

        res.status(500).json({error: error})

    }

}))

router.get('/:id', async (req, res) => {

    // extrair do dado da requisição, pela url = req.params

    const id = req.params.id

    try {

        const person = await Person.findOne({ _id: id })

        if(!person) {

            res.status(422).json({ Mensagem: 'O usuario não foi encontrado!' })

            return

        }

        res.status(200).json(person)

    } catch (error) {

        res.status(500).json({error: error})

    }

})
```



```
    }  
  })  
  
  // atualização de dados  
  router.patch('/:id', async (req, res) => {  
  
    const id = req.params.id  
  
    const {nome, matricula, turma, idade, aprovado} = req.body  
  
    const person = {  
      nome,  
      matricula,  
      turma,  
      idade,  
      aprovado,  
    }  
  
    try {  
  
      const updatedPerson = await Person.updateOne({ _id: id}, person  
    )  
  
      if(updatedPerson.matchedCount === 0 ){
```

```
        res.status(422).json({ Mensagem: 'O usuario não foi encontrado!' })

        return

    }

    res.status(200).json(person)

} catch (error) {

    res.status(500).json({error: error})

}

}))

//deletar dados

router.delete('/:id', async (req, res) => {

    const id = req.params.id

    const person = await Person.findOne({ _id: id })

    if(!person) {

        res.status(422).json({ Mensagem: 'O usuario não foi encontrado!' })

        return

    }

})
```

```

    }

    try {

        await Person.deleteOne({ _id: id })

        res.status(200).json({ Mensagem: 'Usuario removido com
sucesso!' })

    } catch (error) {

        res.status(500).json({error: error})

    }

}))

module.exports = router

```

index.js

```

const express = require('express')

const mongoose = require('mongoose');

const app = express()

// forma de ler JSON / middlewares

app.use(

```

```
express.urlencoded({
  extended: true,
}),
)

app.use(express.json())

// rotas da API

const personRoutes = require('./routes/personRoutes')

app.use('/person', personRoutes)

// rota inicial / endpoint

app.get('/', (req, res) => {

  res.json({ message: 'Olá, seja bem vindo a API!' })

})

// porta

mongoose.connect('mongodb+srv://thalesdias36291:KpexNSaXQn9ut5Yb@cluster0.jar7kbz.mongodb.net/?retryWrites=true&w=majority&appName=Cluster0')

.then(() => {
```

```
console.log('Conectado ao mongoDB!')  
  
app.listen(3000)  
  
}))  
  
.catch((err) => console.log(err))
```

Github: <https://github.com/OdiasThales23>

<https://github.com/OdiasThales23/Atividade-avaliativa>