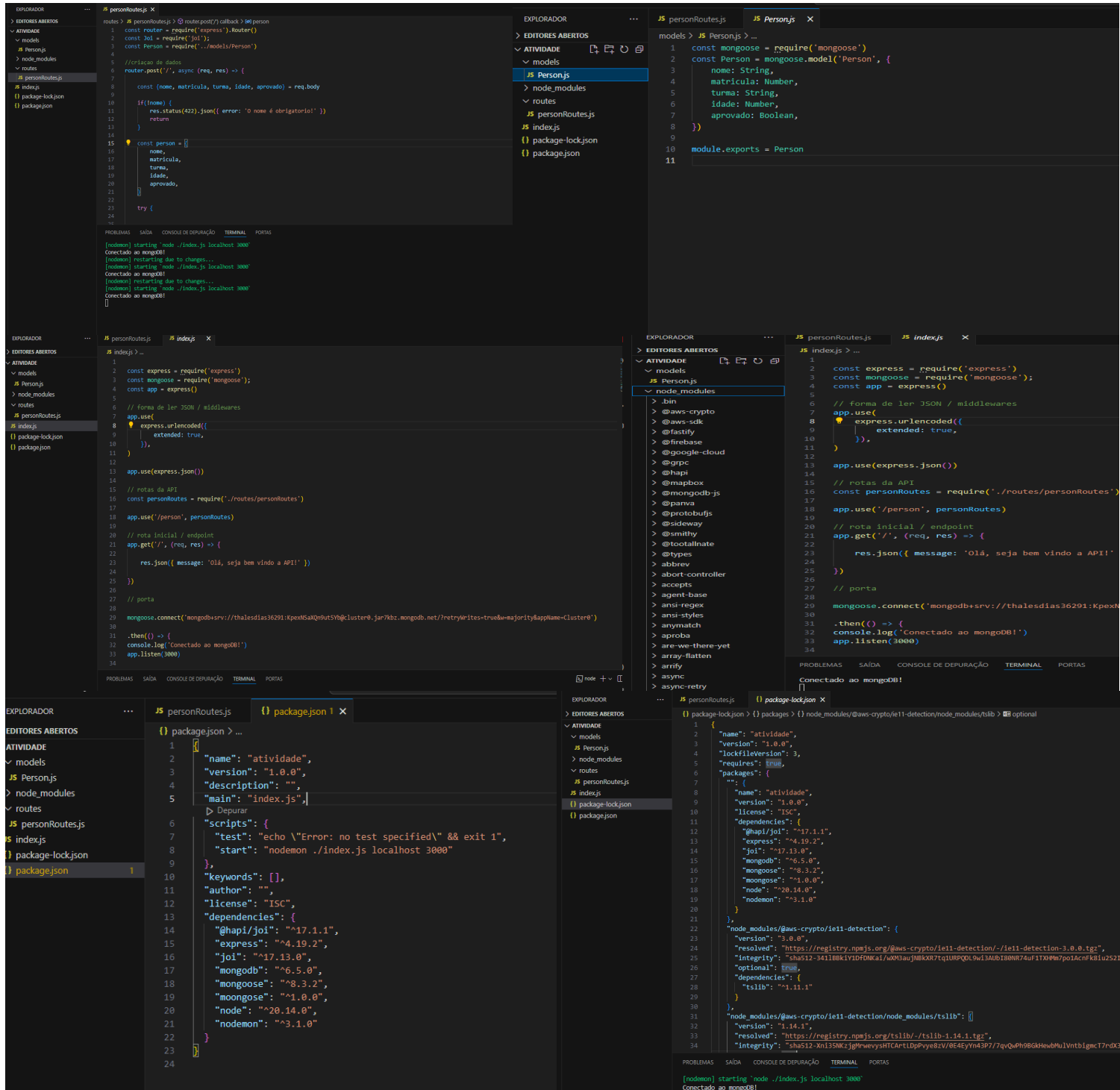


Nome: Thales Dias Prudencio

Tarefa: Desenvolva uma solução no formato RESTful API.

Abaixo algumas imagens do meu código no VS code:



```

// 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 { name, matricula, turma, idade, aprovado } = req.body

  if(!nome) {
    res.status(422).json({ error: 'O nome é obrigatório!' })
    return
  }

  const person = {
    nome,
    matricula,
    turma,
    idade,
    aprovado,
  }

  try {
    // 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!' })
    })

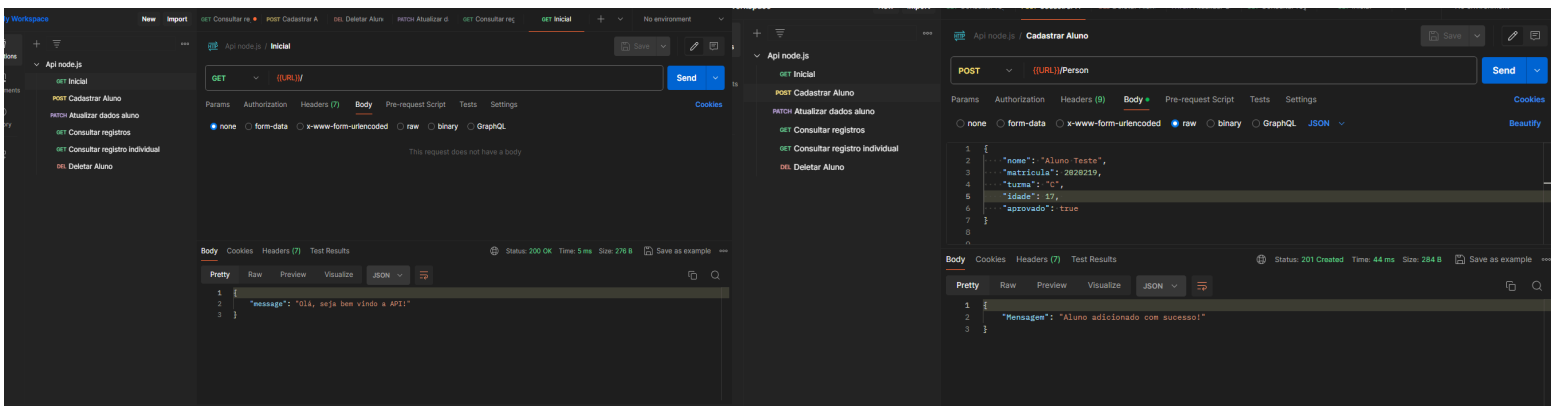
    // porta
    mongoose.connect('mongodb+srv://thalesdias36291:kpxh5AKn0ut5Yb@cluster0-jar7kbs.mongodb.net/?retryWrites=true&majorityOpinion=strict')
    .then(() => {
      console.log('Conectado ao MongoDB!')
      app.listen(3000)
    })
  } catch (error) {
    console.log(error)
  }
})

// package-lock.json
{
  "name": "atividade",
  "version": "1.0.0",
  "description": "",
  "main": "index.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1",
    "start": "nodemon ./index.js localhost 3000"
  },
  "keywords": [],
  "author": "",
  "license": "ISC",
  "dependencies": {
    "@hapi/joi": "^17.1.1",
    "express": "^4.19.2",
    "joi": "^17.13.0",
    "mongodb": "^6.5.0",
    "mongoose": "^8.3.2",
    "mongoose": "^4.0.0",
    "node": "^20.14.0",
    "nodemon": "^3.1.0"
  }
}

```

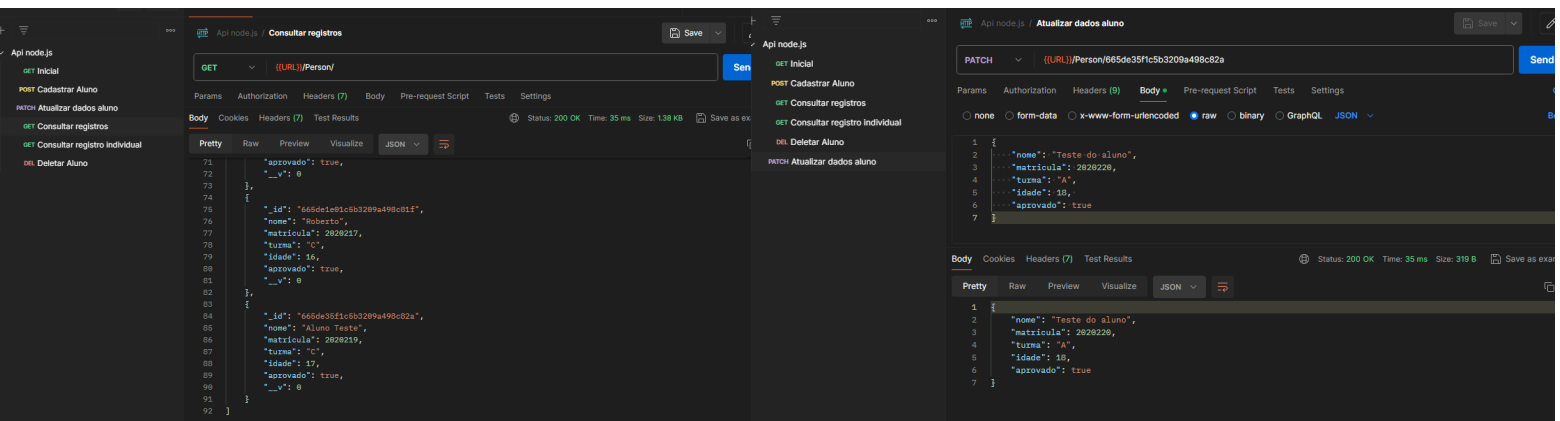
Tela inicial postman

Cadastro aluno



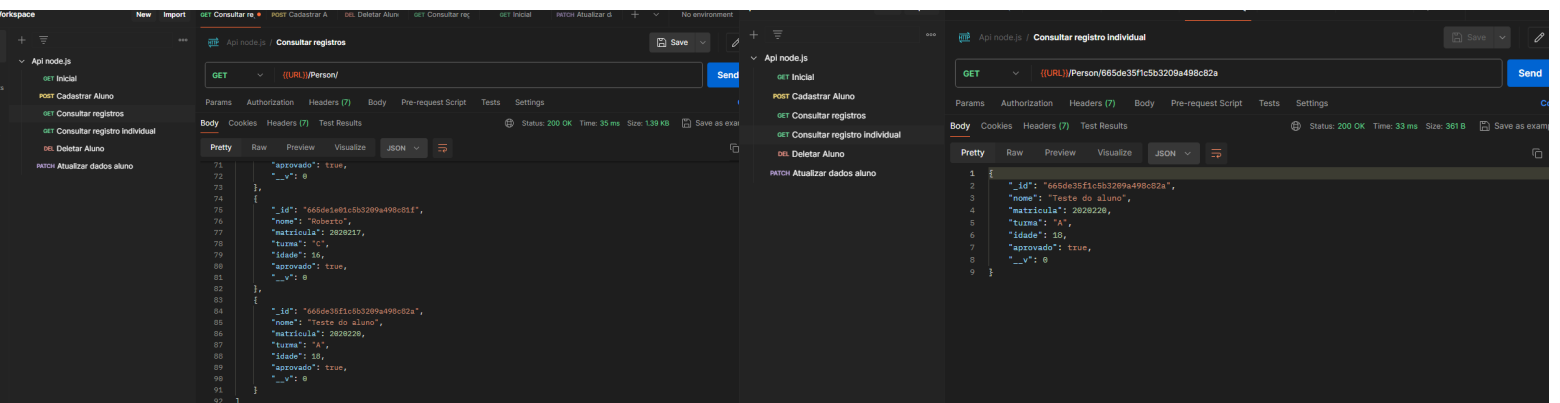
Registros geral

Editar dados aluno(ID)



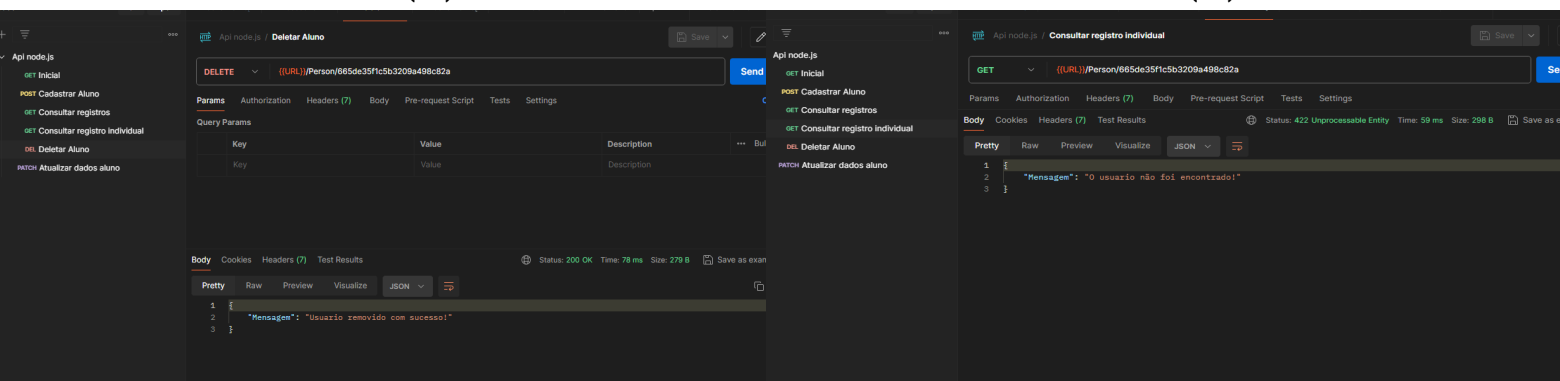
Dados atualizados(ID)

Consultar aluno individual(ID)

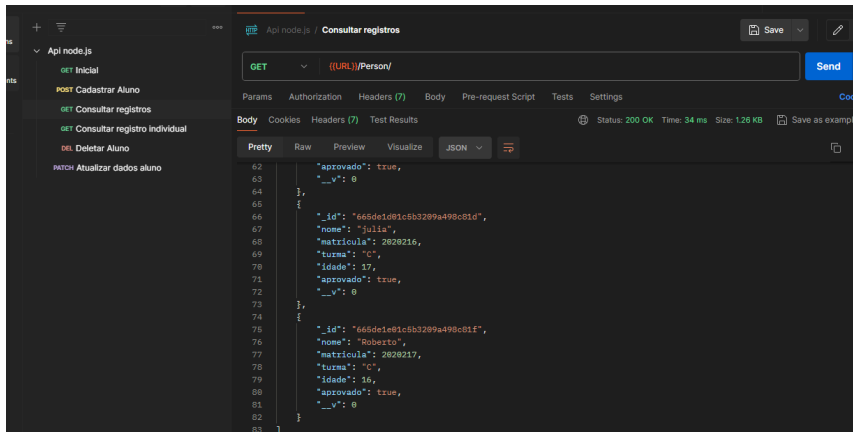


Remover aluno(ID)

Aluno removido(ID)



Registro após remoção do aluno



Dashboards

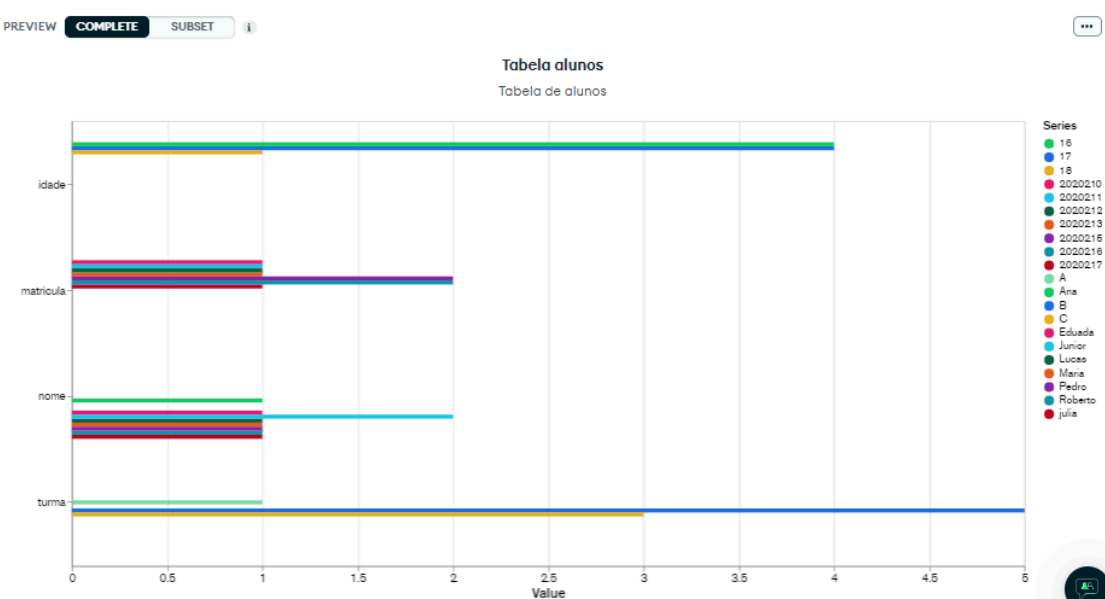
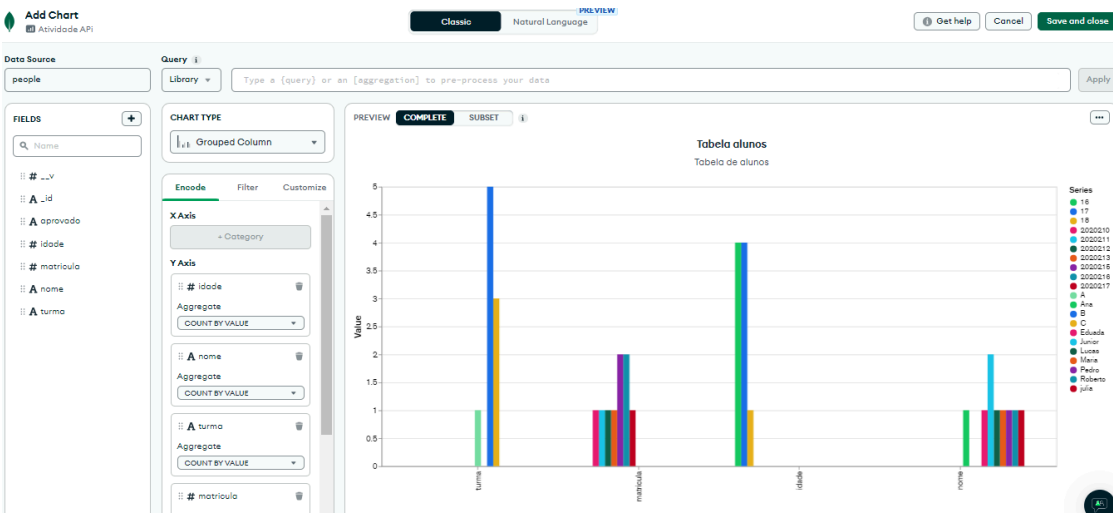
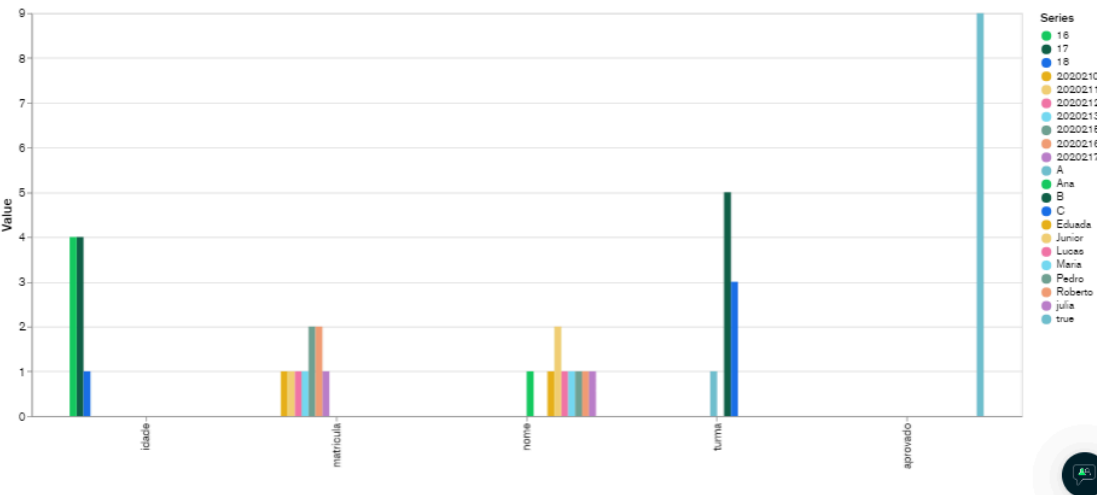
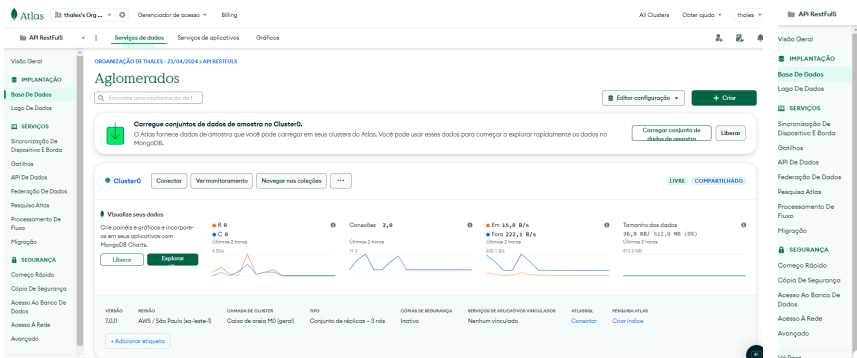


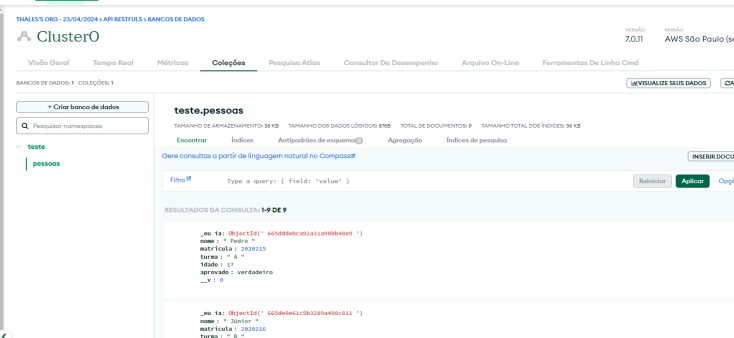
Tabela alunos
Tabela de alunos



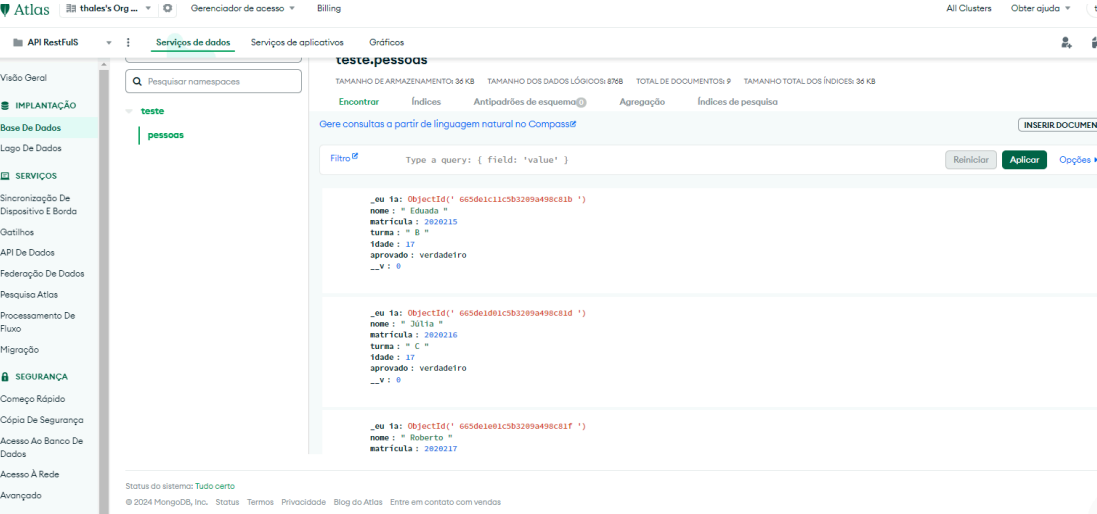
Banco de dados



Alunos registrados no banco de dados



Alunos do banco de dados



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>

Obs: Depois de várias tentativas não foi possível colocar no repositório o **NODE_MODULES**, por ser um arquivo muito grande o repositório não está aceitando.

Documento técnico

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

Realizei a tarefa com todos os requisitos solicitados, com as opções de Adicionar, Excluir, Editar, Consultar registro individualmente e Consultar um grupo de registros.

A API se baseia em um sistema de alunos com o nome, turma, idade, matrícula e se está aprovado, onde se tem a opção de cadastrar novos alunos e ao cadastrar é gerado um ID para cada aluno, você consegue fazer modificações através desse ID gerado no cadastro, como:

Editar pelo seu ID conseguimos realizar a atualização de dados desse aluno e ao atualizar já fica registrado rapidamente os novos dados do aluno.

Excluir um aluno pelo seu ID, ao buscar o ID do aluno que deseja excluir ele será removido do sistema e dos registros rapidamente.

Consultando o registro individualmente, o registro é buscado pelo ID gerado por cada aluno, ao inserir o ID do aluno que desejar e buscar ele já retorna os dados desse aluno.

Consultando os registros gerais, ele seleciona todos alunos e seus dados no sistema e retorna todos os alunos cadastrados.

A API foi conectada pelo mongoDB(atlas), sua conexão foi feita por código e ao modificar qualquer dado cadastrado dos alunos, as atualizações vão diretamente pro banco de dados.

Os testes foram realizados pelo postman, com os métodos, GET, POST, DEL e PATCH.

Os dashboards foram feitos diretamente com os dados do banco de dados pelo atlas, coloquei algumas opções de gráficos em colunas para melhor visualização dos dados.

Neste PDF anexei algumas imagens do meu código, também imagens dos testes realizados no postman e no banco de dados, para melhor visualização do código deixei o link do meu repositório no git.

