

Linguagem de Programação Visual

RICARDO HENDGES



Banco de Dados

Postgres

```
FROM postgres:13.3-buster
```

```
COPY init.sql /docker-entrypoint-initdb.d/
```

```
ENV POSTGRES_USER admin
```

```
ENV POSTGRES_PASSWORD 123456
```

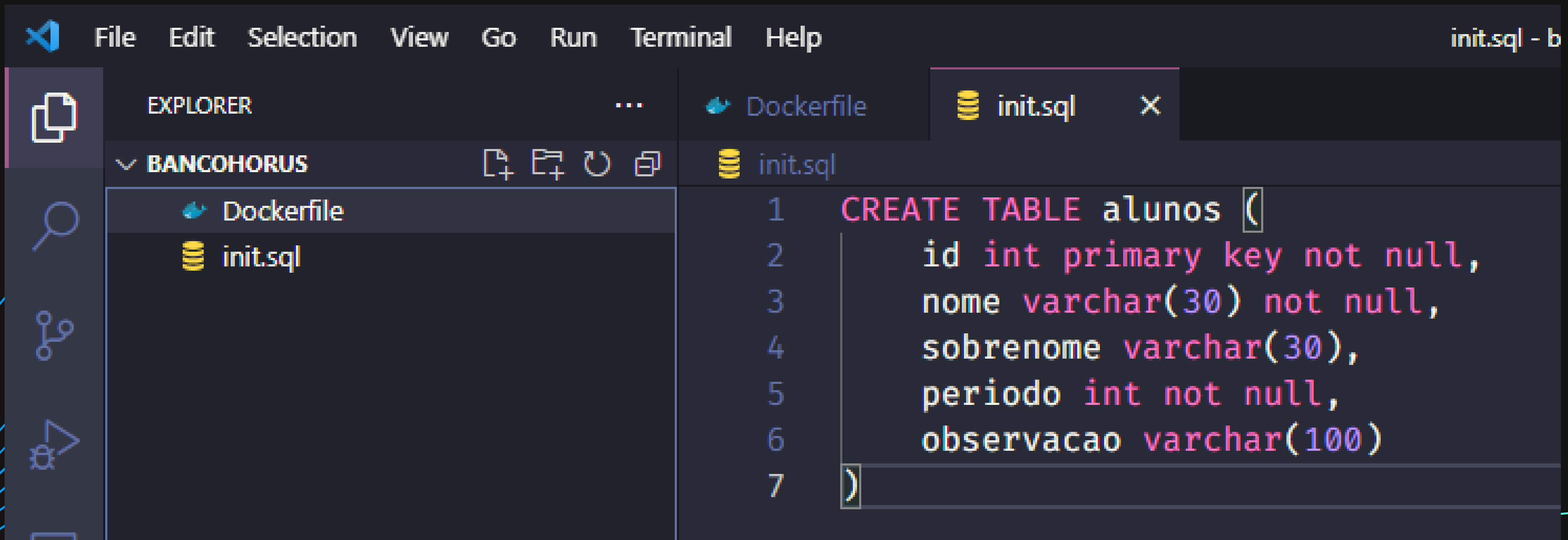
```
ENV POSTGRES_DB horus
```

Usaremos Postgres com Docker para não precisar instalar nada na maquina. Para isso crie um novo diretório, dentro dele um **Dockerfile** e adicione o código acima. Em seguida criaremos o arquivo **init.sql** contendo o SQL de criação da base.

Banco de Dados

Postgres

Pasta para criar banco é composta pelos dois arquivos abaixo.



```
File Edit Selection View Go Run Terminal Help
EXPLORER
BANCOHORUS
  Dockerfile
  init.sql
Dockerfile
init.sql
1 CREATE TABLE alunos (
2     id int primary key not null,
3     nome varchar(30) not null,
4     sobrenome varchar(30),
5     periodo int not null,
6     observacao varchar(100)
7 )
```



docker build and run

Para criar imagem com base no postgres com nosso arquivo init.sql

```
docker build -t horus .
```

Para rodar um container com a imagem criada acima

```
docker run -p 5432:5432 -d horus
```

PG

npm install pg

Node-postgres é uma coleção de módulos node.js para interface com seu banco de dados PostgreSQL. Ele tem suporte para retornos de chamada, promisses, async / await, pooling de conexão e muito mais!

By - node-postgres

<https://www.npmjs.com/package/pg>

<https://node-postgres.com/>

```
"dependencies": {  
  "express": "^4.17.1",  
  "pg": "^8.6.0"  
},
```

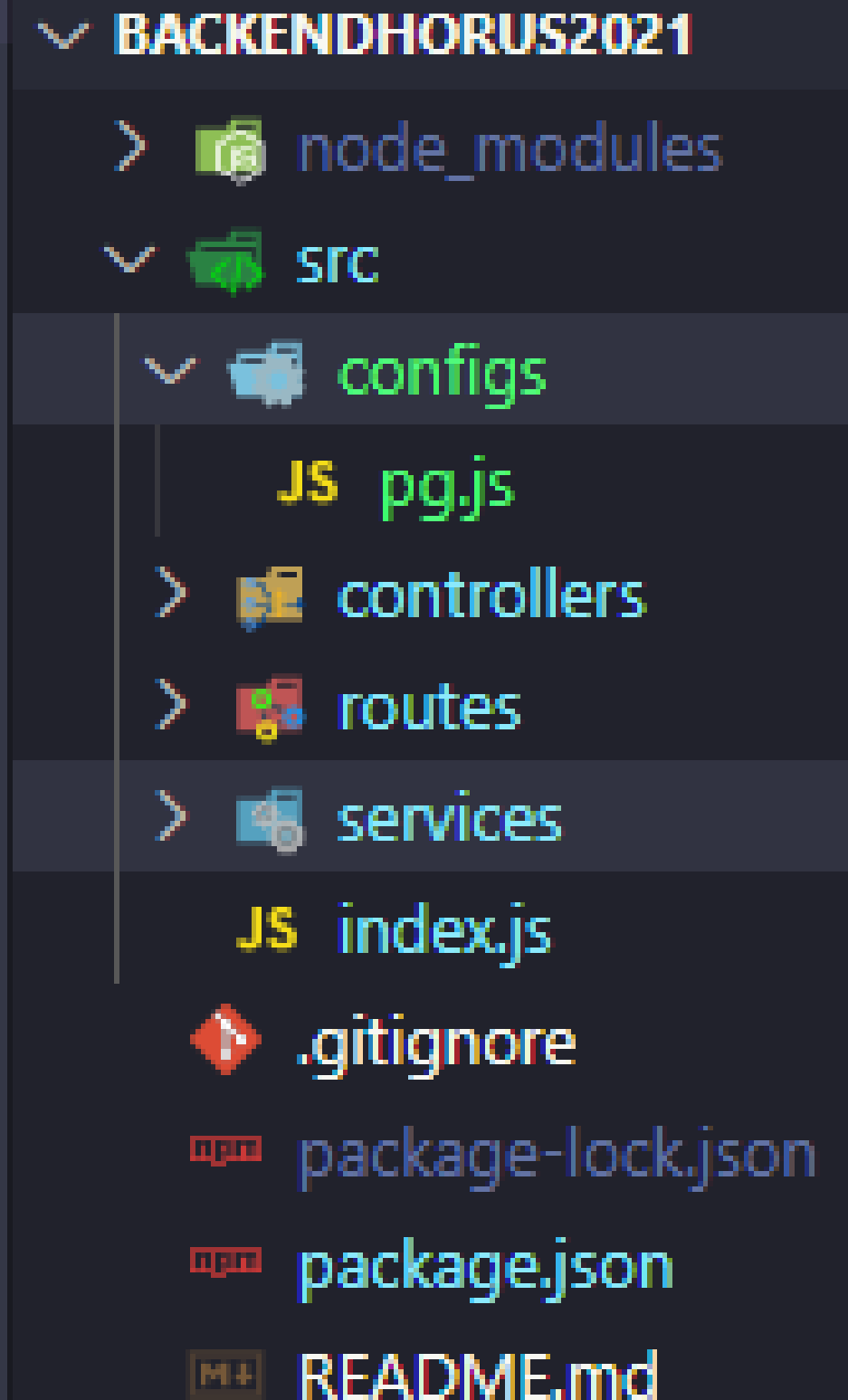
Conexão

Para conectar com o banco criado, dentro do projeto crie um arquivo que chamaremos de `pg.js` dentro de uma pasta `configs` na pasta `src` como esta na imagem ao lado >

```
const { Pool } = require('pg')
```

```
const pool = new Pool({  
  user: 'admin',  
  host: 'localhost',  
  database: 'horus',  
  password: '123456',  
  port: 5432,  
})
```

```
module.exports = { query: (text, params) => pool.query(text, params) }
```



Utilizando a conexão

Service

Adicionaremos uma nova rota para inserir registros na tabela alunos. Abaixo a adição da chamada do metodo `query` do arquivo `pg.js`

```
const db = require('..../configs/pg')

const sql =
  `insert into alunos (id, nome, sobrenome, periodo, observacao)
  values ($1, $2, $3, $4, $5) `
const postAlunos = async (params) => {
  const { id, nome, sobrenome, periodo, observacao } = params
  await db.query(sql, [id, nome, sobrenome, periodo, observacao])
}

module.exports.postAlunos = postAlunos
```

Utilizando a conexão

Controller

Controller igual ao metodo **GET** mudando que os parametro de entrada do **POST** é via **body**

```
const alunoService = require('../services/aluno')

const postAlunos = async(req, res, next) => {
  try {
    const retorno = await alunoService.postAlunos(req.body);
    res.status(201).json(retorno);
  } catch(err) {
    res.status(500).send(err.message);
  }
}

module.exports.postAlunos = postAlunos
```


Utilizando a conexão

Router

```
const alunoController = require(' ../controllers/aluno');  
  
module.exports = (app) => {  
  app.post('/aluno', alunoController.postAlunos)  
}
```

Acesso ao body da req

```
const express = require('express')
const app = express()
app.use(express.json())

require('./routes')(app)
app.get('/', (req, res) => res.status(200).send('Hello World'))

app.listen(3000)
```

You, 6 hours ago • rotas

Postman

Postman é uma plataforma de colaboração para desenvolvimento de API. Os recursos do Postman simplificam cada etapa da construção de uma API e otimizam a colaboração para que você possa criar APIs melhores - mais rapidamente. by - Postman



<https://www.postman.com/downloads/>



Configurando...

POST http://localhost:3...



...

http://localhost:3000/aluno

POST



http://localhost:3000/aluno

Params

Authorization

Headers (8)

Body



Pre-request Script

Tests

Settings

☐ none

☐ form-data

☐ x-www-form-urlencoded

☒ raw

☐ binary

☐ GraphQL

JSON



```
1 {  
2   .... "id": 1,  
3   .... "nome": "Solange",  
4   .... "sobrenome": "Machado",  
5   .... "periodo": 7,  
6   .... "observacao": "Aprovado"  
7 }
```

Service Final

Ajuste do SQL de
GET dos alunos

```
const db = require('../configs/pg')

const sql_get =
  `select id,
        nome,
        sobrenome,
        periodo,
        observacao
   from alunos `

const getAluno = async () => {
  let aluno = {}
  await db.query(sql_get)
    .then(ret => aluno = { total: ret.rows.length, alunos: ret.rows })
    .catch(err => aluno = err.stack )
  return aluno
}

const sql_insert =
  `insert into alunos (id, nome, sobrenome, periodo, observacao)
   values ($1, $2, $3, $4, $5) `

const postAlunos = async (params) => {
  const { id, nome, sobrenome, periodo, observacao } = params
  await db.query(sql_insert, [id, nome, sobrenome, periodo, observacao])
}

module.exports.getAluno = getAluno
module.exports.postAlunos = postAlunos
```

Controller Final

Ajuste na remoção de
uso de params do GET

```
let alunoService = require(' ../services/aluno')

const getAlunos = async (req, res, next) => {
  try {
    await alunoService.getAluno()
      .then(ret => res.status(201).send(ret))
      .catch(err => res.status(500).send(err))
  } catch (err) {
    next(err);
  }
}

const postAlunos = async (req, res, next) => {
  try {
    await alunoService.postAlunos(req.body)
      .then(ret => res.status(201).send(ret))
      .catch(err => res.status(500).send(err))
  } catch (err) {
    next(err);
  }
}

module.exports.getAlunos = getAlunos
module.exports.postAlunos = postAlunos
```

Routes

Final

```
const alunoController = require(' ../controllers/aluno');  
  
module.exports = (app) => {  
  app.get('/aluno', alunoController.getAlunos)  
  app.post('/aluno', alunoController.postAlunos)  
}
```

You, seconds ago • Uncommitted changes

Testando rotas - Postman

