

# Criando API com NodeJS

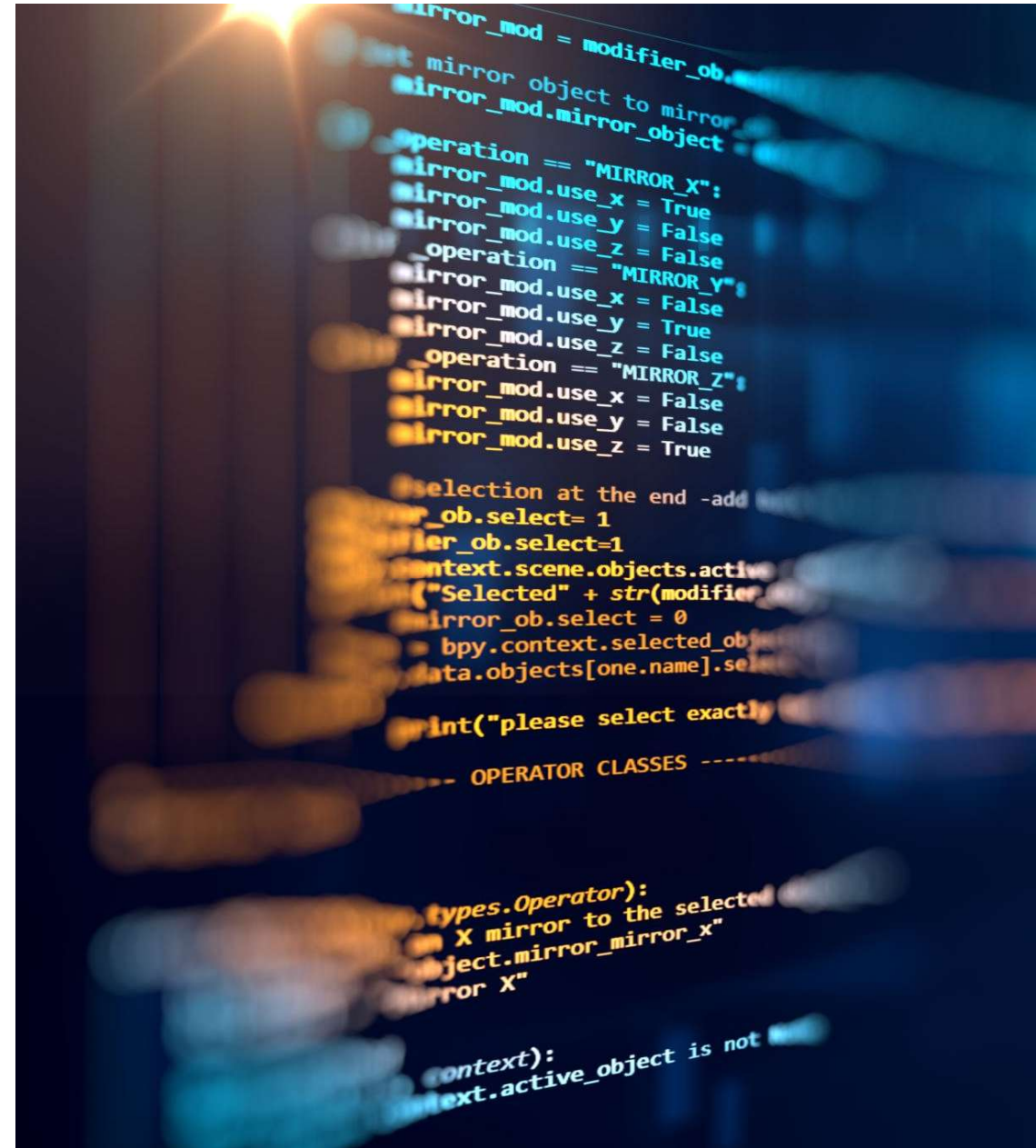
---

Fabrício Tonetto Londero



# Criando o projeto

- Abra uma pasta do computador no VS Code
- Abra o terminal do VS Code ou equivalente nesse diretório
- No Terminal:
  - `npm init -y`
  - Isso cria o `package.json`



# Criando o projeto – instalando dependencias

- No Terminal
  - `npm install express mongoose dotenv morgan cors`
- `express` → cria a API
- `mongoose` → conecta ao MongoDB
- `dotenv` → variáveis de ambiente (.env)
- `morgan` → log de requisições (para debug)
- `cors` → permite acessar sua API de outros lugares

# Criando o projeto

- Crie o arquivo “.env” – Configura os dados secretos de conexão com o banco
  - MONGODB\_URI=mongodb+srv://USUARIO:SENHA@cluster0.iurgocm.mongodb.net/aula?retryWrites=true&w=majority
  - PORT=3000

# Criando a API

- Crie um arquivo “serve.js”

## server.js – parte 1

```
import 'dotenv/config';  
import express from 'express';  
import mongoose from 'mongoose';  
  
const app = express();  
app.use(express.json()); // API aceita JSON
```

## server.js – parte 2

```
// Conectar no MongoDB Atlas  
mongoose.connect(process.env.MONGODB_URI, { dbName: 'Aula' })  
  .then(() => console.log('Conectado ao MongoDB'))  
  .catch(err => console.error('Erro na conexão:', err.message));
```

# server.js – parte 3

- Modelo – devemos fazer para cada entidade existente no banco/projeto

```
// Modelo Aluno
const alunoSchema = new mongoose.Schema({
  nome: { type: String, required: true, trim: true, minlength: 2 },
  idade: { type: Number, required: true, min: 0, max: 120 },
  curso: { type: String, required: true, trim: true },
  notas: { type: [Number], default: [], validate: v => v.every(n => n >= 0 && n <= 10) }
}, { collection: 'Alunos', timestamps: true });
const Aluno = mongoose.model('Aluno', alunoSchema, 'Alunos');

```



## server.js – parte 4 - rotas

```
// Rota inicial
app.get('/', (req, res) => res.json({ msg: 'API rodando' }));

// Criar aluno
app.post('/alunos', async (req, res) => {
  const aluno = await Aluno.create(req.body);
  res.status(201).json(aluno);
});
```

## server.js – parte 5

```
// Listar alunos
app.get('/alunos', async (req, res) => {
  const alunos = await Aluno.find();
  res.json(alunos);
});

// Iniciar servidor
app.listen(process.env.PORT, () =>
  console.log(`Servidor rodando em http://localhost:${process.env.PORT}` )
);
```

# Verificar as permissões do MongoDB

The screenshot shows the MongoDB Atlas interface for a project named 'FACENS > PROJECT 0'. The 'Network Access' section is active, with tabs for 'IP Access List', 'Peering', and 'Private Endpoint'. A warning message states: 'You will only be able to connect to your cluster from the following list of IP Addresses:'. Below this, a table lists the current IP addresses:

IP Address
45.6.166.245/32
45.6.166.239/32 (includes your current IP address)

An 'Add IP Access List Entry' modal is open on the right. It contains the following fields and options:

- ALLOW ACCESS FROM ANYWHERE** (button)
- Access List Entry:** Text input field with '0.0.0.0/0' entered.
- Comment:** Text input field with placeholder text 'Optional comment describing this entry'.
- Temporary Entry:** A toggle switch is currently off. Next to it is the text 'This entry is temporary and will be deleted in' followed by a dropdown menu showing '6 hours'.
- Buttons:** 'Cancel' and 'Confirm' buttons at the bottom right.

# Rodando a API

- No terminar:
  - `node server.js`

GET ▼ http://localhost:3000/alunos Send 200 OK 99 ms 607 B

ParamsBodyAuthHeaders3ScriptsDocs

URL PREVIEW

http://localhost:3000/alunos

QUERY PARAMETERS

Import from URLBulk Edit

+ AddDelete allDescription

name	value
------	-------

PATH PARAMETERS

Path parameters are url path segments that start with a colon ':' e.g. 'id'

PreviewHeaders7CookiesTests0/0Mock

Preview

```
1 [
2   {
3     "_id": "68bc703ee09ef66fe5932f02",
4     "nome": "Maria Silva",
5     "idade": 20,
6     "curso": "Engenharia",
7     "notas": [
8       7.5,
9       8,
10      9.2
11    ]
12  },
13  {
14    "_id": "68bc7072e09ef66fe5932f03",
15    "nome": "João Souza",
16    "idade": 22,
17    "curso": "Medicina",
18    "notas": [
19      7.5,
20      8,
21      6.5
22    ]
23  },
24  {
25    "_id": "68bc7247e09ef66fe5932f05",
26    "nome": "Ana Lima",
27    "idade": 19,
28    "curso": "Direito",
29    "notas": [
30      9,
31      8.8,
32      9.5
33    ]
34  },
35  {
36    "_id": "68bc7817e09ef66fe5932f06",
37    "nome": "Fabrício",
38    "idade": 35,
39    "curso": "ADS",
40    "notas": [
41      7.5,
42      9,
43      9.5
44    ]
45  }
46 ]
```

# Finalizando o CRUD

```
app.put('/alunos/:id', async (req, res) => {
  try {
    if (!mongoose.isValidObjectId(req.params.id)) {
      return res.status(400).json({ error: 'ID inválido' });
    }
    const aluno = await Aluno.findByIdAndUpdate(
      req.params.id,
      req.body,
      { new: true, runValidators: true, overwrite: true }
    );
    if (!aluno) return res.status(404).json({ error: 'Aluno não encontrado' });
    res.json(aluno);
  } catch (err) {
    res.status(400).json({ error: err.message });
  }
});
```

# Delete

```
app.delete('/alunos/:id', async (req, res) => {
  try {
    if (!mongoose.isValidObjectId(req.params.id)) {
      return res.status(400).json({ error: 'ID inválido' });
    }
    const aluno = await Aluno.findByIdAndDelete(req.params.id);
    if (!aluno) return res.status(404).json({ error: 'Aluno não encontrado' });
    res.json({ ok: true });
  } catch (err) {
    res.status(500).json({ error: err.message });
  }
});
```



# Consulta pelo ID

```
app.get('/alunos/:id', async (req, res) => {
  try {
    if (!mongoose.isValidObjectId(req.params.id)) {
      return res.status(400).json({ error: 'ID inválido' });
    }
    const aluno = await Aluno.findById(req.params.id);
    if (!aluno) return res.status(404).json({ error: 'Aluno não encontrado' });
    res.json(aluno);
  } catch (err) {
    res.status(500).json({ error: err.message });
  }
});
```



# Testando

The screenshot shows the Postman application interface. At the top, there is a tab bar with several tabs: "Scratch Pad", "POST New Request", "GET New Request", "POST New Request", "POST New Request", "GET New Request", and "POST New Request". The active tab is "POST New Request".

The main interface is divided into two main sections. The left section is for configuring the request, and the right section is for viewing the response.

**Request Configuration:**


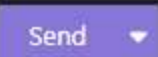
- Method:** POST
- URL:** http://localhost:3000/alunos
- Body:** Selected, containing a JSON object:



```
1 {
2   "nome": "Tiago",
3   "idade": 28,
4   "curso": "ADS",
5   "notas": [9.0, 8.8, 9.5]
6 }
7
```
- Headers:** 4 headers are listed.
- Scripts:** No scripts are present.
- Docs:** No documentation is present.


**Response:**


- Status:** 201 Created
- Time:** 82 ms
- Size:** 180 B
- Preview:** The response is a JSON object:

```
1 {
2   "nome": "Tiago",
3   "idade": 28,
4   "curso": "ADS",
5   "notas": [
6     9,
7     8.8,
8     9.5
9   ],
10  "_id": "68bc856f10bd87073abf35b7",
11  "createdAt": "2025-09-06T19:03:11.256Z",
12  "updatedAt": "2025-09-06T19:03:11.256Z",
13  "__v": 0
14 }
```

GET  http://localhost:3000/alunos/68bc856f10bd87073abf35b7  200 OK 47 ms 180 B

Params Body Auth Headers  Scripts Docs Preview Headers  Cookies Tests 0/0 → Mock Console

No Body 

Preview 

```
1 {
2   "_id": "68bc856f10bd87073abf35b7",
3   "nome": "Tiago",
4   "idade": 28,
5   "curso": "ADS",
6   "notas": [
7     9,
8     8.8,
9     9.5
10  ],
11   "createdAt": "2025-09-06T19:03:11.256Z",
12   "updatedAt": "2025-09-06T19:03:11.256Z",
13   "__v": 0
14 }
```

PUT ▼ http://localhost:3000/alunos/68bc856f10bd87073abf35b7

Send ▼

200 OK

65 ms

180 B

Params

Body ●

Auth

Headers 4

Scripts

Docs

Preview

Headers 7

Cookies

Tests 0 / 0

→ Mock

JSON ▼

```
1 {  
2   "notas": [  
3     9.9,  
4     8.0,  
5     9.5  
6   ]  
7 }
```

Preview ▼

```
1 {  
2   "_id": "68bc856f10bd87073abf35b7",  
3   "nome": "Tiago",  
4   "idade": 28,  
5   "curso": "ADS",  
6   "notas": [  
7     9.9,  
8     8,  
9     9.5  
10  ],  
11   "createdAt": "2025-09-06T19:03:11.256Z",  
12   "updatedAt": "2025-09-06T19:07:46.399Z",  
13   "__v": 0  
14 }
```