



*Inovação e Excelência desde 1902*

# **Sistemas Operacionais e Arquiteturas Cloud Native**



## Requisitos:

1º O servidor deverá operar dentro de uma das ferramentas de cloud computing.

Utilizamos Render para a plataforma Cloud e Docker no banco de dados **PostgreSQL**.

Segue abaixo imagem do arquivo **docker-compose** do nosso repositório que gera a imagem do banco localmente e logo após o **Dockerfile** do projeto com ele rodando:

```
docker-compose.yml x
docker-compose.yml
To-Mat, 2 months ago | 1 author (To-Mat)
1  services:
2    postgres:
3      image: postgres:14-alpine3.14
4      environment:
5        POSTGRES_USER: ${DATABASE_USER}
6        POSTGRES_PASSWORD: ${DATABASE_PASS}
7        POSTGRES_DB: ${DATABASE_NAME}
8        PG_DATA: /var/lib/postgresql/data
9      volumes:
10     - /var/lib/postgresql/data
11     ports:
12     - '6222:5432' To-Mat, 2 months ago • feat: initial
```

```
1 # Estágio 1: Compilar a aplicação
2 FROM node:18-alpine AS build
3
4 # COPY package.json and package-lock.json files
5 COPY package*.json ./
6
7 # generated prisma files
8 COPY prisma ./prisma/
9
10 # COPY ENV variable
11 COPY .env ./
12
13 # COPY tsconfig.json file
14 COPY tsconfig.json ./
15
16 # COPY
17 COPY . .
18
19 RUN npm install
20 RUN npx prisma generate
21
22 # Run and expose the server on port 3000
23 EXPOSE 3000
24
25 # A command to start the server
26 CMD npm start
27
```



Inovação e Excelência **desde 1902**

```
total: 272.1MB
➤ pass-manager-api git:(main) ✖ sudo docker build -t my-app .
[+] Building 9.9s (13/13) FINISHED
=> [internal] load build definition from dockerfile
=> => transferring dockerfile: 466B
=> [internal] load .dockerignore
=> => transferring context: 2B
=> [internal] load metadata for docker.io/library/node:18-alpine
=> CACHED [1/8] FROM docker.io/library/node:18-alpine@sha256:1ccc70acda680aa4ba47f53e7c40b2d4d6892de74817128e0662d32647dd7f4d
=> [internal] load build context
=> => transferring context: 273.58MB
=> [2/8] COPY package*.json ./
=> [3/8] COPY prisma ./prisma/
=> [4/8] COPY .env ./
=> [5/8] COPY tsconfig.json ./
=> [6/8] COPY . .
=> [7/8] RUN npm install
=> [8/8] RUN npx prisma generate
=> exporting to image
=> => exporting layers
=> => writing image sha256:63a5ec93f6d20225cdd4c3b614af70fdbf642c432d93b092c5dca4f018be69de
=> => naming to docker.io/library/my-app
➤ pass-manager-api git:(main) ✖ sudo docker run --env-file .env my-app

> pass-manager-api@1.0.0 start
> node dist/index.js

Server listening on port 3000
[]
```

```
PROBLEMS OUTPUT TERMINAL DEBUG CONSOLE GITLens
➤ ➤ pass-manager-api git:(main) ✖ sudo docker ps
[sudo] password for mateus:
CONTAINER ID   IMAGE     COMMAND                  CREATED        STATUS        PORTS           NAMES
3eebea584c1d   my-app    "docker-entrypoint.s..." 4 minutes ago  Up 4 minutes  3000/tcp        trusting_clarke
```

2º Apresentar ao menos um banco de dados com uma tabela:

Segue abaixo tabelas de usuários e tabela para cadastro das senhas de usuários respectivamente:

User ✕ +						
⌂ Filters None Fields All Showing 1 of 1 Add record						
<input type="checkbox"/>	id #	email A	username A	password A	createdAt 📅	updatedAt 📅
<input type="checkbox"/>	1	mateusquent2003@gmail...	mateusKent123	\$2b\$04\$berMHSuyVJhHrd...	2023-05-06T18:40:11.9...	2023-05-06T18:40:11.9...
						0 Credentials



*Inovação e Excelência desde 1902*

Credentials X +				
⌂	Filters	None	Fields	All
	Showing	0 of 0	Add record	
<input type="checkbox"/>	id #	emailCredential A?	usernameCredential A?	credentialPassword A
websiteName A				

⚙				
websiteUrl A?	user {}	userId #	createdAt 📅	updatedAt

3º Deve conter um código de servidor mínimo para responder o aplicativo mobile:

Segue abaixo trechos de códigos que demonstram a conexão com o servidor:

```
1 import { Injectable } from '@angular/core';
2 import { HttpClient } from '@angular/common/http';
3 import { environment } from 'src/environments/environments';
4 import { map } from 'rxjs';
5 import { signUpUserDTO } from '../DTOs/signUpUserDTO';
6 import { signInUserDTO } from '../DTOs/signInUserDTO';
7 import { User } from '../../user/models/User';
8 import jwt_decode from 'jwt-decode';
9
10 interface AuthResponse {
11   username: string;
12   email: string;
13   password: string;
14   token: string;
15 }
16
17 @Injectable({
18   providedIn: 'root',
19 })
20 export class AuthService {
21   url: string;
22
23   constructor(private http: HttpClient) {
24     this.url = `${environment.url}/users`;
25   }
26
27   signUp(input: signUpUserDTO) {
28     return this.http.post<AuthResponse>(`${this.url}/create`, input).pipe(
29       map((data) => {
30         sessionStorage.setItem('token', data.token);
31         return data;
32       })
33     );
34   }
35
36   signIn(input: signInUserDTO) {
37     return this.http.post<AuthResponse>(`${this.url}/auth`, input).pipe(
38       map((data) => {
39         sessionStorage.setItem('token', data.token);
40         return data;
41       })
42     );
43   }
44
45   logout() {
46     sessionStorage.removeItem('token');
47   }
48
49   isAuthenticated() {
50     return !!sessionStorage.getItem('token');
51   }
52
53   getToken() {
54     return sessionStorage.getItem('token');
55   }
56
57   getUserIdFromToken() {
58     const decodedToken: any = jwt_decode(sessionStorage.getItem('token')!);
59     return decodedToken.sub.userId;
60   }
61 }
62
```



```
1 import { HttpClient, HttpHeaders } from '@angular/common/http';
2 import { Injectable } from '@angular/core';
3 import { map } from 'rxjs';
4 import { environment } from 'src/environments/environments';
5 import { registerAndUpdateCredentialDTO } from '../DTOs/registerAndUpdateCredentialDTO';
6 import { Credential } from '../models/Credential';
7
8 import { AuthService } from '../../auth/services/auth.service';
9
10 @Injectable({
11   providedIn: 'root',
12 })
13 export class CredentialService {
14   url: string;
15
16   constructor(private http: HttpClient, private authService: AuthService) {
17     this.url = `${environment.url}/credentials`;
18   }
19
20   register(input: registerAndUpdateCredentialDTO) {
21     return this.http.post(`${this.url}/create`, input).pipe(
22       map((data) => {
23         return data;
24       })
25     );
26   }
27
28   obtain(id: number) {
29     return this.http.get<Credential>(`${this.url}/getCredential/${id}`).pipe(
30       map((data) => {
31         return data;
32       })
33     );
34   }
35
36   list() {
37     return this.http.get<Credential[]>(`${this.url}/getCredentials`).pipe(
38       map((data) => {
39         return data;
40       })
41     );
42   }
43
44   update(id: number, input: registerAndUpdateCredentialDTO) {
45     return this.http.put(`${this.url}/updateCredential/${id}`, input).pipe(
46       map((data) => {
47         return data;
48       })
49     );
50   }
51
52   delete(id: number) {
53     return this.http.delete(`${this.url}/deleteCredential/${id}`).pipe(
54       map((data) => {
55         return data;
56       })
57     );
58   }
59 }
60
```

```
1 import { HttpClient } from '@angular/common/http';
2 import { Injectable } from '@angular/core';
3 import { map } from 'rxjs';
4 import { environment } from 'src/environments/environments';
5 import { UpdateUserDTO } from '../DTOs/updateUserDTO';
6 import { User } from '../models/User';
7
8 @Injectable({
9   providedIn: 'root',
10 })
11 export class UserService {
12   url: string;
13
14   constructor(private http: HttpClient) {
15     this.url = `${environment.url}/users`;
16   }
17
18   obtain(id: number) {
19     return this.http.get<User>(`${this.url}/getUser/${id}`).pipe(
20       map((data) => {
21         return data;
22       })
23     );
24   }
25
26   update(id: number, input: UpdateUserDTO) {
27     return this.http.put(`${this.url}/update/${id}`, input).pipe(
28       map((data) => {
29         return data;
30       })
31     );
32   }
33
34   delete(id: number) {
35     return this.http.delete(`${this.url}/delete/${id}`).pipe(
36       map((data) => {
37         return data;
38       })
39     );
40   }
41 }
42
```