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:

```dockerfile
# Estágio 1: Compilar a aplicação
FROM node:18-alpine AS build

# COPY package.json and package-lock.json files
COPY package*.json ./

# generated prisma files
COPY prisma ./prisma/

# COPY ENV variable
COPY .env ./

# COPY tsconfig.json file
COPY tsconfig.json ./

# COPY
COPY . .

RUN npm install
RUN npx prisma generate

# Run and expose the server on port 3000
EXPOSE 3000

# A command to start the server
CMD npm start

```

```
pass-manager-api git:(main) ✗ sudo docker build -t my-app .
[+] Building 9.9s (13/13) FINISHED
 => [internal] load build definition from dockerfile                                                                0.0s
 => => transferring dockerfile: 466B                                                                                0.0s
 => [internal] load .dockerignore                                                                                   0.0s
 => => transferring context: 2B                                                                                     0.0s
 => [internal] load metadata for docker.io/library/node:18-alpine                                                   0.5s
 => CACHED [1/8] FROM docker.io/library/node:18-alpine@sha256:1ccc70acda680aa4ba47f53e7c40b2d4d6892de74817128e0662d32647dd7f4d  0.0s
 => [internal] load build context                                                                                   1.4s
 => => transferring context: 273.58MB                                                                               1.4s
 => [2/8] COPY package*.json ./                                                                                     0.4s
 => [3/8] COPY prisma ./prisma/                                                                                     0.0s
 => [4/8] COPY .env ./                                                                                              0.0s
 => [5/8] COPY tsconfig.json ./                                                                                     0.0s
 => [6/8] COPY . .                                                                                                  1.2s
 => [7/8] RUN npm install                                                                                           1.4s
 => [8/8] RUN npx prisma generate                                                                                   3.7s
 => exporting to image                                                                                              1.2s
 => => exporting layers                                                                                             1.2s
 => => writing image sha256:63a5ec93f6d20225cdd4c3b614af70fdbf642c432d93b092c5dca4f018be69de                        0.0s
 => => naming to docker.io/library/my-app                                                                           0.0s
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
```

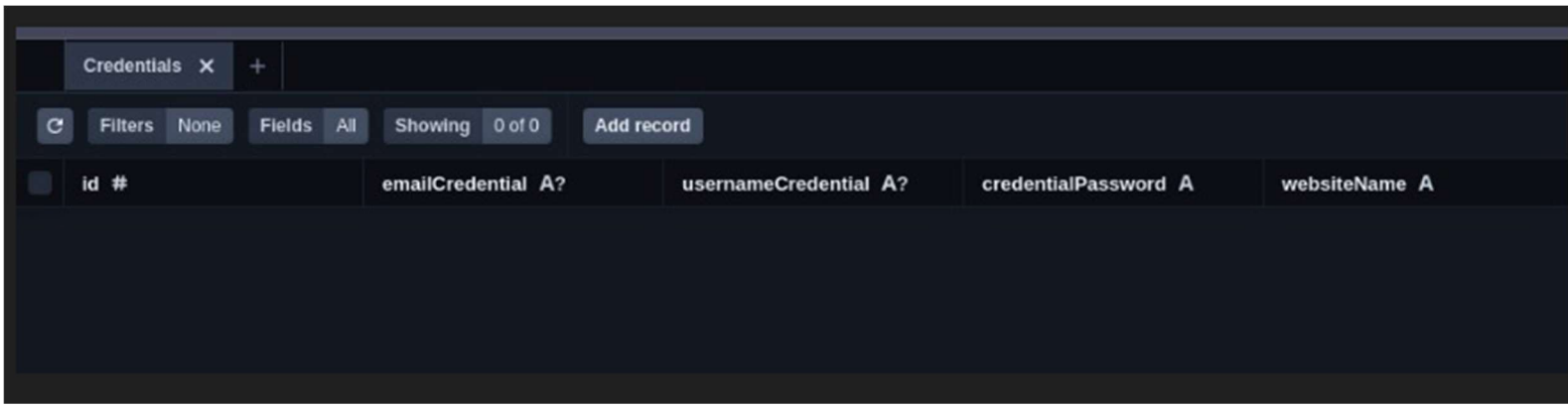
2º Apresentar ao menos um banco de dados com uma tabela:

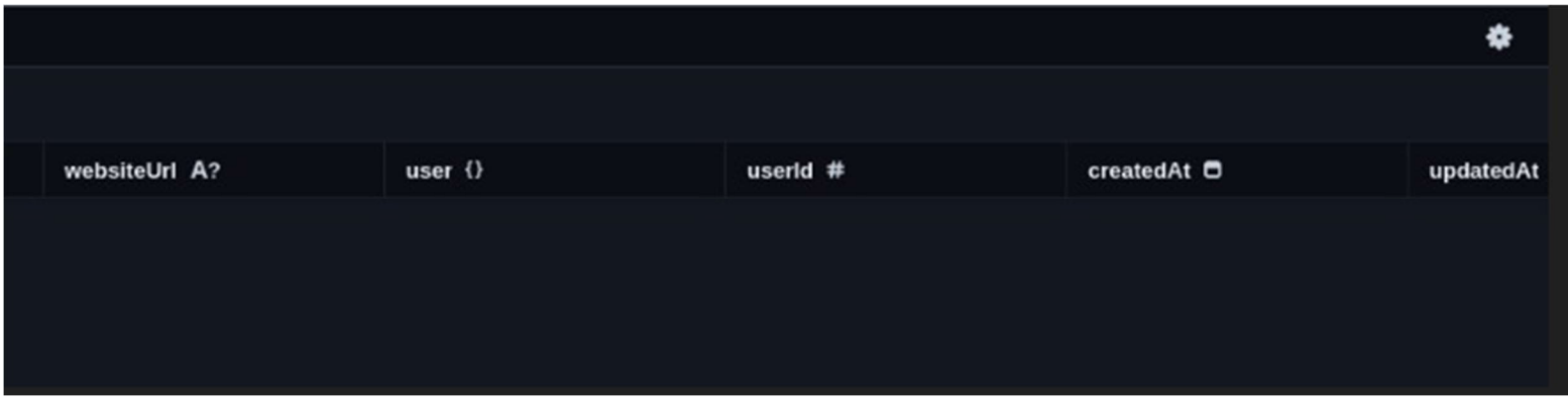
Segue abaixo tabelas de usuários e tabela para cadastro das senhas de usuários respectivamente:

User ×

Filters None | Fields All | Showing 1 of 1 | Add record

id #	email A	username A	password A	createdAt	updatedAt	Credentials []
1	mateusquent2003@gmail…	mateusKent123	$2b$04$berMHSUyVJhHrd…	2023-05-06T18:40:11.9…	2023-05-06T18:40:11.9…	0 Credentials





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:

```typescript
import { Injectable } from '@angular/core';
import { HttpClient } from '@angular/common/http';
import { environment } from 'src/environments/environments';
import { map } from 'rxjs';
import { signUpUserDTO } from '../DTOs/signUpUserDTO';
import { signInUserDTO } from '../DTOs/signInUserDTO';
import { User } from '../../user/models/User';
import jwt_decode from 'jwt-decode';

interface AuthResponse {
  username: string;
  email: string;
  password: string;
  token: string;
}

@Injectable({
  providedIn: 'root',
})
export class AuthService {
  url: string;

  constructor(private http: HttpClient) {
    this.url = `${environment.url}/users`;
  }

  signUp(input: signUpUserDTO) {
    return this.http.post<AuthResponse>(`${this.url}/create`, input).pipe(
      map((data) => {
        sessionStorage.setItem('token', data.token);
        return data;
      })
    );
  }

  signIn(input: signInUserDTO) {
    return this.http.post<AuthResponse>(`${this.url}/auth`, input).pipe(
      map((data) => {
        sessionStorage.setItem('token', data.token);
        return data;
      })
    );
  }

  logout() {
    sessionStorage.removeItem('token');
  }

  isAuthenticated() {
    return !!sessionStorage.getItem('token');
  }

  getToken() {
    return sessionStorage.getItem('token');
  }

  getUserIdFromToken() {
    const decodedToken: any = jwt_decode(sessionStorage.getItem('token')!);
    return decodedToken.sub.userId;
  }
}
```

```typescript
import { HttpClient, HttpHeaders } from '@angular/common/http';
import { Injectable } from '@angular/core';
import { map } from 'rxjs';
import { environment } from 'src/environments/environments';
import { registerAndUpdateCredentialDTO } from '../DTOs/registerAndUpdateCredentialDTO';
import { Credential } from '../models/Credential';

import { AuthService } from '../../auth/services/auth.service';

@Injectable({
  providedIn: 'root',
})
export class CredentialService {
  url: string;

  constructor(private http: HttpClient, private authService: AuthService) {
    this.url = `${environment.url}/credentials`;
  }

  register(input: registerAndUpdateCredentialDTO) {
    return this.http.post(`${this.url}/create`, input).pipe(
      map((data) => {
        return data;
      })
    );
  }

  obtain(id: number) {
    return this.http.get<Credential>(`${this.url}/getCredential/${id}`).pipe(
      map((data) => {
        return data;
      })
    );
  }

  list() {
    return this.http.get<Credential[]>(`${this.url}/getCredentials`).pipe(
      map((data) => {
        return data;
      })
    );
  }

  update(id: number, input: registerAndUpdateCredentialDTO) {
    return this.http.put(`${this.url}/updateCredential/${id}`, input).pipe(
      map((data) => {
        return data;
      })
    );
  }

  delete(id: number) {
    return this.http.delete(`${this.url}/deleteCredential/${id}`).pipe(
      map((data) => {
        return data;
      })
    );
  }
}

```

```angular2html
import { HttpClient } from '@angular/common/http';
import { Injectable } from '@angular/core';
import { map } from 'rxjs';
import { environment } from 'src/environments/environments';
import { UpdateUserDTO } from '../DTOs/updateUserDTO';
import { User } from '../models/User';

@Injectable({
  providedIn: 'root',
})
export class UserService {
  url: string;

  constructor(private http: HttpClient) {
    this.url = `${environment.url}/users`;
  }

  obtain(id: number) {
    return this.http.get<User>(`${this.url}/getUser/${id}`).pipe(
      map((data) => {
        return data;
      })
    );
  }

  update(id: number, input: UpdateUserDTO) {
    return this.http.put(`${this.url}/update/${id}`, input).pipe(
      map((data) => {
        return data;
      })
    );
  }

  delete(id: number) {
    return this.http.delete(`${this.url}/delete/${id}`).pipe(
      map((data) => {
        return data;
      })
    );
  }
}

```