BACKEND

Criar uma api rest, implementando um CRUD para o cadastro de Clientes(ver campos abaixo).
A API deve ser feita preferencialmente em SpringBoot (versão maior ou igual a 3.0.0), java 17 e o banco de dados (preferencialmente Postgres).

Tabela :

CLIENTE
id -> inteiro e sequencial
nome -> String com 255 caracteres
email -> String com 255 caracteres
cpf -> String com 11 caracteres
renda -> decimal com 7 na parte inteira e 2 na parte fracionária
telefone -> string deve ser capaz de gravar o ddd + número do telefone
data_criacao -> compo data em formato de data e hora, na api deve ser tratado como UTC e precisa ser parte da api do cadastro - Não gerar automaticamente.

- **Apenas o campo de telefone não é obrigatório.**

Desafios:

- API com autenticação, dois usuários: **admin e comum**, apenas **admin** pode realizar exclusão - Fazer validação preferencialmente no código.
- Usar record como DTO.
- Usar campo sequencial no banco.
- Usar as devidas validações para os campos tanto na api e no banco de dados.
- Mensagens adequadas para exibir no frontend.
- Não aceitar nomes repetidos.
- Não aceitar cpf repetidos.
- Rodar o banco em docker.
- Configurar a API para ser distribuída dentro de um docker.

FRONTEND

Criar uma interface com duas telas: Cadastro de Cliente e uma de login nas seguintes condições:

- Tela de Login: Composta de usuário e senha, onde o usuário é uma lista de dois com valores fixos (**admin e comum**) e a senha será validada no código da API.

- A tela de Cadastro de Cliente precisa ter os seguintes campos:

Listagem precisa exibir os seguintes campos:

- **nome - com no máximo 255 caracteres**
- **e-mail - com no máximo 255 caracteres**
- **cpf - com no máximo 11 números.**
- **renda -> decimal com 7 na parte inteira e 2 na parte fracionária**
- **telefone -> ddd + número do telefone**
- **data de criação -> campo data em formato de data e hora.**

- **Apenas o campo de telefone não é obrigatório.**

Desafios:

- Validação no campo E-mail.
- cpf com validação de tamanho, tipo e máscara adequada.
- renda com validação de tamanho, tipo e máscara adequada.
- telefone se preenchido, com validação de tamanho, tipo e máscara adequada.
- Data de Criação com componente de calendário adequado para selecionar data e hora.
- Preferencialmente feito em React com rotas e adequada transformação dos dados em json para comunicação com a API.
- Tratamento das mensagens adequadas para validação de campos e erros retornados pela API.
- A data de criação não pode ser inferior a data corrente, mas o horário pode Exemplo: Data_Hoje: 18/09/2023 19:50:50 a Data_Criacao = 18/09/2023 01:50:50 é válida.