Proposta

Crie um aplicativo backend que irá expor uma API RESTful de criação de sing up/sign in.

Todos os endpoints devem somente aceitar e somente enviar JSONs. O servidor deverá retornar JSON para os casos de endpoint não encontrado também.

O aplicativo deverá persistir os dados (ver detalhes em requisitos).

Todas as respostas de erro devem retornar o objeto:

```
{
 "mensagem": "mensagem de erro"
}
```

Prazo

- 3 dias corridos.

Sign up

- Este endpoint deverá receber um usuário com os seguintes campos: nome, email, senha e uma lista de objetos telefone. Seguem os modelos:

```
{
 "nome": "string",
 "email": "string",
 "senha": "senha",
 "telefones": [
  {
   "numero": "123456789",
   "ddd": "11"
  }
 ]
}
```

- Usar status codes de acordo
- Em caso de sucesso irá retornar um usuário mais os campos:
- id: id do usuário (pode ser o próprio gerado pelo banco, porém seria interessante se fosse um GUID)
- data_criacao: data da criação do usuário
- data_atualizacao: data da última atualização do usuário
- ultimo_login: data do último login (no caso da criação, será a mesma que a criação)
- token: token de acesso da API (pode ser um GUID ou um JWT)
- Caso o e-mail já exista, deverá retornar erro com a mensagem "E-mail já existente".
- O token deverá ser persistido junto com o usuário

Sign in

- Este endpoint irá receber um objeto com e-mail e senha.
- Caso o e-mail exista e a senha seja a mesma que a senha persistida, retornar igual ao endpoint de sign_up.
- Caso o e-mail não exista, retornar erro com status apropriado mais a mensagem "Usuário e/ou senha inválidos"
- Caso o e-mail exista mas a senha não bata, retornar o status apropriado 401 mais a mensagem "Usuário e/ou senha inválidos"

Buscar usuário

- Chamadas para este endpoint devem conter um header na requisição de Authentication com o valor "Bearer {token}" onde {token} é o valor do token passado na criação ou sign in de um usuário.
- Caso o token não exista, retornar erro com status apropriado com a mensagem "Não autorizado".
- Caso o token exista, buscar o usuário pelo user_id passado no path e comparar se o token no modelo é igual ao token passado no header.
- Caso não seja o mesmo token, retornar erro com status apropriado e mensagem "Não autorizado"
- Caso seja o mesmo token, verificar se o último login foi a MENOS que 30 minutos atrás.
- Caso não seja a MENOS que 30 minutos atrás, retornar erro com status apropriado com mensagem "Sessão inválida".
- Caso tudo esteja ok, retornar o usuário.

Requisitos

- Persistência de dados
- Gestão de dependências via gerenciador de pacotes (npm)
- Utilização de Eslint
- API: Express, Hapi ou similares.
- Utilizar banco nosql

Requisitos desejáveis

- JWT como token
- Testes unitários
- Criptografia não reversível (hash) na senha e no token
- Mongo
- Utilização de sintaxe de JS ES6
- Utilização de padrões de projeto para organização do código (arquitetura hexagonal, Domain Driven Design, etc.)

Submissão

- O desafio deve ser entregue pelo GitHub/Bitbucket.