API RESTful para Sistema de Usuários de Carros

Criar aplicação que exponha uma API RESTful de criação de usuários e carros com login.

Atributos do usuário:

Nome	Tipo	Descrição
firstName	String	Nome do usuário
lastName	String	Sobrenome do usuário
email	String	E-mail do usuário
birthday	Date	Data de nascimento do usuário
login	String	Login do usuário
password	String	Senha do usuário
phone	String	Telefone do usuário
cars	List	Lista de carros do usuário

Atributos do carro:

Nome	Tipo	Descrição
year	Int	Ano de fabricação do carro
licensePlate	String	Placa do carro
model	String	Modelo do carro
color	String	Cor predominante do carro

As seguintes rotas devem ser implementadas:

a) Rotas que NÃO exigem autenticação

Rota	Descrição	Erros possíveis
/api/signin	Esta rota espera um objeto com os campos login e password, e deve retornar o token de acesso da API (JWT) com as informações do usuário logado.	1
/api/users	Listar todos os usuários	
/api/users	Cadastrar um novo usuário	2,3,4,5
/api/users/{id}	Buscar um usuário pelo id	
/api/users/{id}	Remover um usuário pelo id	
/api/users/{id}	Atualizar um usuário pelo id	2,3,4,5

Erros possíveis:

1. **Login inexistente ou senha inválida**: retornar um erro com a mensagem "Invalid login or password";
2. **E-mail já existente:** retornar um erro com a mensagem "Email already exists";
3. **Login já existente:** retornar um erro com a mensagem "Login already exists";
4. **Campos inválidos:** retornar um erro com a mensagem "Invalid fields";
5. **Campos não preenchidos:** retornar um erro com a mensagem "Missing fields".

Exemplo de JSON para criação de usuário:

```
{
     "firstName": "Hello",
     "lastName": "World",
     "email": "hello@world.com",
     "birthday": "1990-05-01",
     "login": "hello.world",
```

```
    "password": "h3ll0",
    "phone": "988888888",
    "cars": [
    {
        "year": 2018,
        "licensePlate": "PDV-0625",
        "model": "Audi",
        "color": "White"
    }
  ]
}
```

b) Rotas que exigem autenticação

Todas as rotas abaixo esperam o token de acesso da API (JWT) via header Authorization

Rota	Descrição	Erros possíveis
/api/me	Retornar as informações do usuário logado (firstName, lastName, email, birthday, login, phone, cars) + createdAt (data da criação do usuário) + lastLogin (data da última vez que o usuário realizou login).	1,2
/api/cars	Listar todos os carros do usuário logado	1,2
/api/cars	Cadastrar um novo carro para o usuário logado	1,2,3,4,5
/api/cars/{id}	Buscar um carro do usuário logado pelo id	1,2
/api/cars/{id}	Remover um carro do usuário logado pelo id	1,2
/api/cars/{id}	Atualizar um carro do usuário logado pelo id	1,2,3,4,5

Erros possíveis:

1. **Token não enviado**: retornar um erro com a mensagem "Unauthorized";
2. **Token expirado**: retornar um erro com a mensagem "Unauthorized - invalid session";
3. **Placa já existente:** retornar um erro com a mensagem "License plate already exists";
4. **Campos inválidos:** retornar um erro com a mensagem "Invalid fields";
5. **Campos não preenchidos:** retornar um erro com a mensagem "Missing fields".

Exemplo de JSON para criação de carro:

```
{
    "year": 2018,
    "licensePlate": "PDV-0625",
    "model": "Audi",
    "color": "White"
}
```

Observações:

- Criar um front-end mínimo com tecnologia baseada em JS (preferencialmente em Angular);
- A aplicação deve aceitar e responder apenas em JSON;
- O id do usuário e o id do carro podem ser sequenciais gerados pelo banco ou identificadores únicos;
- Todas as rotas devem responder o código de status HTTP apropriado;
- Espera-se que as mensagens de erro tenham o seguinte formato:

```
{ "message": "Error message", "errorCode": 123 }
```

Requisitos:

- JWT como token;
- Servidor deve estar embutido na aplicação (Tomcat, Undertow ou Jetty);
- Processo de build via Maven;
- Banco de dados em memória, como H2;

- Framework Spring;
- Utilizar no mínimo Java 8;
- Persistência com JPA/Hibernate;
- Disponibilizar a API rodando em algum host (Heroku, AWS, Digital Ocean, etc);
- Testes unitários;
- Criar um repositório público em alguma ferramenta de git (Github, Gitlab, Bitbucket, etc). O desenvolvimento deve simular uma mini-sprint (Scrum), dividindo o desafio em estórias de usuário (story users).
    - O README.md do projeto deverá ter uma seção **ESTÓRIAS DE USUÁRIO** com a lista numerada de estórias de usuário que foram concebidas para a implementação do desafio. A primeira linha de cada commit do repositório deve utilizar a descrição da estória de usuário associada.
    - O README.md do projeto deverá ter uma seção **SOLUÇÃO** com as justificativas e defesa técnica da solução que está sendo entregue.
    - O README.md do projeto deve ser claro e mostrar tudo que precisa ser feito para executar o build do projeto, deploy, testes, etc.

Requisitos desejáveis:

- Criação de rota para upload da fotografia do usuário e do carro;
- Agendamento de tarefas (Ex: @Scheduled do Spring);
- Senha deve ser criptografada;
- Documentação (Javadoc);
- Design Pattern;
- Pipeline no Jenkins;
- Integração com SonarQube;
- Integração com JFrog Artifactory ou Nexus;
- Balanceador de cargas, Service Registry e/ou Proxy (Eureka, Zuul, Camel, Helix, etc.);
- Swagger.

Requisito extra (bonus stage):

- Construir uma estrutura de dados ordenada que sempre que um carro for utilizado (rota **/api/cars/{id}** for invocada), um contador referente ao carro específico será incrementado. Os carros de um usuário deverão ser ordenados (ordem decrescente) de acordo com o total de utilizações e os usuários deverão ser ordenados (ordem decrescente) de acordo com o somatório total de utilizações de todos os seus carros;
- Caso os usuários possuam a mesma quantidade total de utilizações de seus carros, o **login** deverá ser utilizado como critério de desempate (ordem alfabética);
- Caso os carros possuam a mesma quantidade de utilizações, o **modelo** deverá ser utilizado como critério de desempate (ordem alfabética);
- No exemplo ilustrado na figura abaixo, os usuários de login **User 1** e **User 4** estão em posições invertidas, uma vez que possuem a mesma quantidade total de utilizações de seus carros (9 + 3 + 3 = 5 + 5 + 5 = 15) e o login **User 1** precede alfabeticamente o login **User 4**.
- No exemplo ilustrado na figura abaixo, os carros **Ford** e **Audi** do usuário de login **User 1** estão em posições invertidas, uma vez que o carro **Audi** possui mais utilizações que o carro **Ford** (5 > 3).