Relatório - Passos de construção da aplicação *Petshop*
Nome: Arthur Francisco Caruso

Parte 1: Desenvolvimento do *back-end* da aplicação:

- **Passo 1**: Geração do *template* do *back-end* através do site https://start.spring.io/

- **Passo 2**: Seleção das dependências necessárias para o funcionamento do servidor *back-end* utilizando *Spring*:
  - *spring-boot-starter-data-jpa*
  - *spring-boot-starter-validation*
  - *spring-boot-starter-web*
  - *postgresql*
  - *lombok*

- **Passo 3**: Após gerado o *template*, são criadas as pastas e módulos com todas as funcionalidades do *back-end* ainda não implementadas

- **Passo 4**: Tendo o código necessário para que o *back-end* funcione, application.properties é configurada para iniciar o servidor *back-end* na *URL* http://localhost:8080/ com o banco de dados ativo em http://localhost:5432/petshop

- **Passo 5**: Implementação do *back-end* de acordo com os requisitos do trabalho:
  Criação do banco de dados do trabalho:
  `sudo -iu postgres psql createdb petshop`
  Execução do back-end através do terminal:
  `mvn install package && java -jar ./target/petshop-0.0.1-SNAPSHOT.jar`

- **Passo 6**: Testes do *back-end* de acordo com os requisitos do trabalho:
  Teste de requisição de todos os objetos do banco de dados:
  `curl -X GET http://localhost:8080/Pet/getAll`
  Teste de criação de objeto no banco de dados:
  `curl -X POST -H "Content-Type: application/json" -d @./test_data.json http://localhost:8080/Pet/create`
  ***UUID do objeto para testes criado: b41c17cd-b9f2-43c1-b9a0-9d25e7db4062**

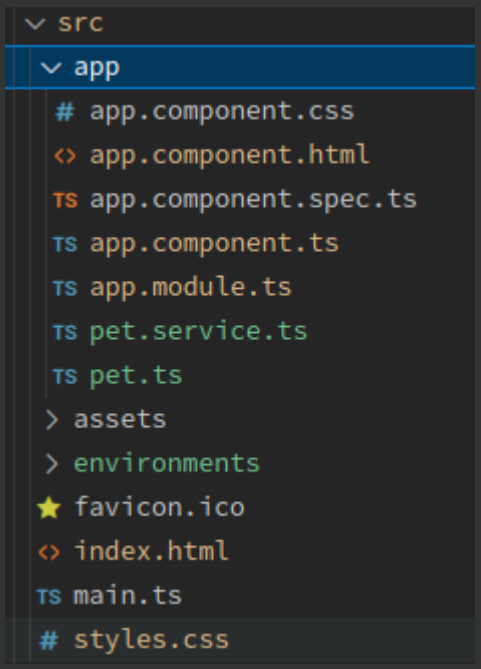
  Teste de requisição de um objeto específico do banco de dados:
  `curl -X GET http://localhost:8080/Pet/findById/b41c17cd-b9f2-43c1-b9a0-9d25e7db4062`
  Teste de atualização de um objeto no banco de dados:
  `curl -X PUT -H "Content-Type: application/json" -d @./test_data.json http://localhost:8080/Pet/update`
  Teste de deleção de um objeto no banco de dados:
  `curl -X DELETE http://localhost:8080/Pet/delete/b41c17cd-b9f2-43c1-b9a0-9d25e7db4062`

Passo 7: Configurar o *CORS* no *back-end* em via de permitir requisições da aplicação *front-end* à *API*:

```java
@Bean
public CorsFilter corsFilter() {
    CorsConfiguration corsConfiguration = new CorsConfiguration();
    corsConfiguration.setAllowCredentials(true);
    corsConfiguration.setAllowedOrigins(Arrays.asList("http://localhost:4200"));
    corsConfiguration.setAllowedHeaders(Arrays.asList("Origin", "Access-Control-Allow-Origin", "Content-Type",
            "Access-Control-Request-Method", "Access-Control-Request-Headers"));
    corsConfiguration.setExposedHeaders(Arrays.asList("Origin", "Content-Type", "Accept", "Authorization",
            "Access-Control-Allow-Origin", "Access-Control-Allow-Origin", "Access-Control-Allow-Credentials"));
    corsConfiguration.setAllowedMethods(Arrays.asList("GET", "POST", "PUT", "DELETE", "OPTIONS"));
    UrlBasedCorsConfigurationSource urlBasedCorsConfigurationSource = new UrlBasedCorsConfigurationSource();
    urlBasedCorsConfigurationSource.registerCorsConfiguration("/**", corsConfiguration);
    return new CorsFilter(urlBasedCorsConfigurationSource);
}
```

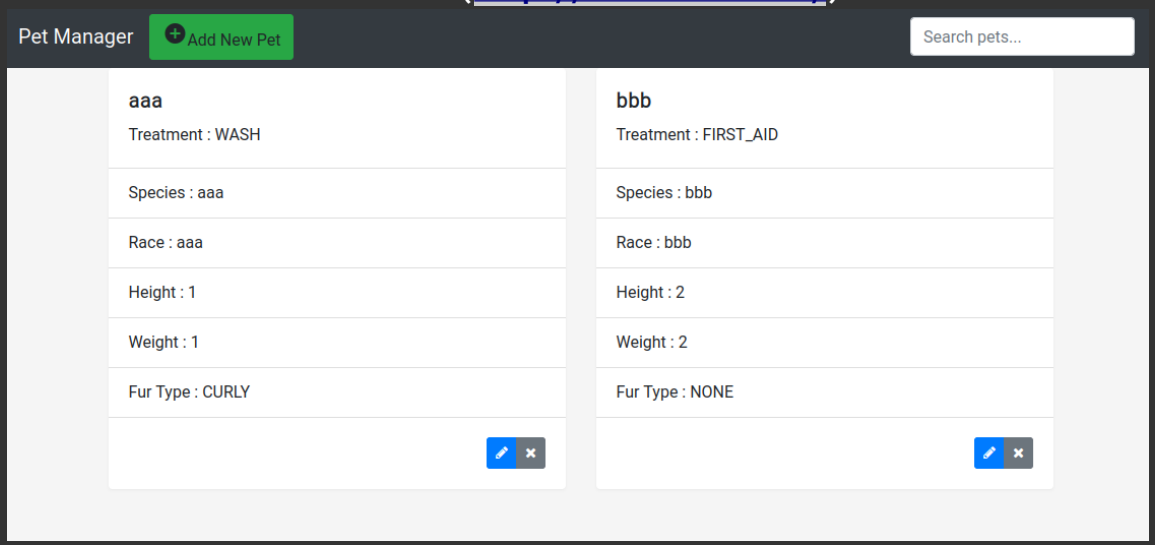
Parte 2: Desenvolvimento do primeiro *front-end* (escrito em *typescript* e usando *angular*):

- **Passo 1**: Inicializar o gerenciador de pacotes *npm*, instalar e inicializar o pacote *typescript* e *angular/cli*
  - Iniciar o npm:
    - `npm init -y`
  - Instalar o typescript:
    - `npm install typescript`
  - Inicializar projeto typescript:
    - `npx tsc –init`
  - Instalar o angular/cli:
    - `npm install @angular/cli`
  - Inicializar projeto angular:
    - `npx ng new petshop`
  - Iniciar o servidor front-end:
    - `cd petshop && npx ng serve`



- **Passo 2**: Implementar o modelo, serviço e controlador que representará os objetos vindos do *back-end* para o *front-end* e vice-versa, junto da página principal e formulários modais.

RESULTADO (http://localhost:4200/):

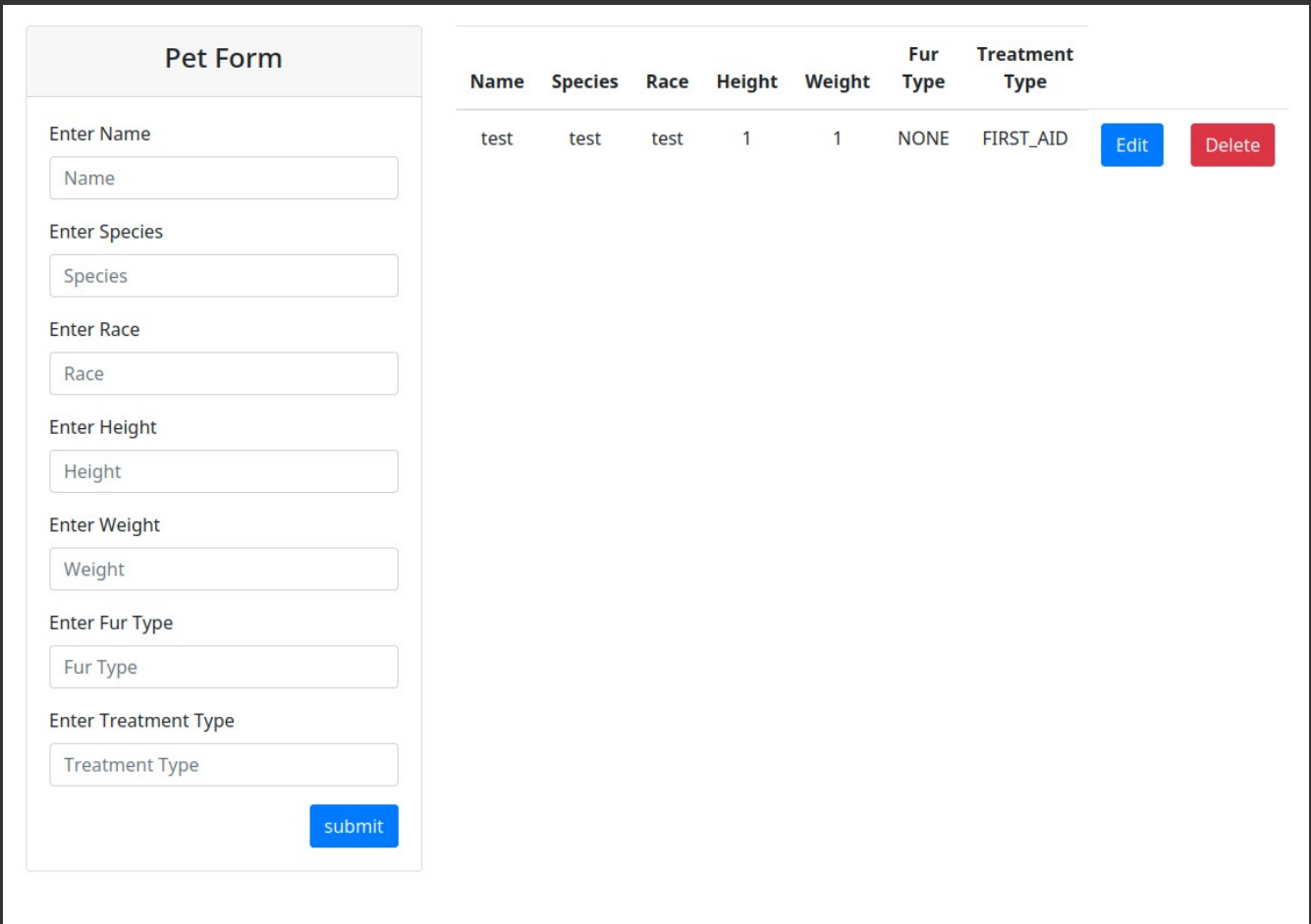


Parte 3: Desenvolvimento do segundo front-end (escrito em *html*, *css* e *javascript*):

- **Passo 1**: Configurar servidor front-end local para testes utilizando *CORS*:
  Iniciando um servidor python em http://localhost:4200/:
  `python -m http.server -b "127.0.0.1" 4200 --cgi`

- **Passo 2**: Implementar chamadas à *API* através do método *fetch*

RESULTADO (http://localhost:4200/)



Parte 4: ideias de implantação futura:

- Permitir o *front-end* escrito em *javascript* pesquisar por registros específicos;

- Consertar as *responses* da *API* para permitir melhor ideia do andamento das transações entre o *front-end* e o *back-end;*

- Paginação dos dados e limitação de requisições;

- Unificação do *design* dos dois *front-ends.*