Desenvolvimento WEB

-

Full Stack Completo: Java + React

Prof. Roberto Alencar

Configurando Dados do Ambiente em Arquivos Locais

Configurando Dados do Ambiente em Arquivos Locais

- Muitas vezes precisamos colocar em um arquivo de configurações dados confidenciais de acesso a determinados recursos do projeto como, por exemplo, o banco de dados do cliente, login/senha de servidores de e-mail, dados de contas de *storages* privados e etc.

- Neste caso, não é recomendado que esses dados sejam compartilhados através do versionamento do Git.

- Para resolver esse problema, recomenda-se criar arquivos locais e adicionar esses arquivos no `.gitignore`

Configurando Dados do Ambiente em Arquivos Locais

- Vamos fazer isso em nosso projeto !!!
- Na raiz do projeto **do Back-end**, crie um arquivo chamado `.env`
  - OBS 1: `env` é uma abreviação de *environment* (ambiente em inglês)
  - OBS 2: o arquivo poderia ter qualquer outro nome

Configurando Dados do Ambiente em Arquivos Locais

- Conteúdo do arquivo `.env`

```
# Dados Banco de Dados

HOST_DB=localhost
PORT_DB=5443
NAME_DB=oxefood
PASS_DB=oxefood

# Dados do Serviço de Emails

EMAIL_USER=seumail@ifpe.edu.br
EMAIL_PASSWORD=suasenha
```

Basicamente é um arquivo com variáveis e valores.

Neste exemplo, estou definindo variáveis que serão utilizadas posteriormente no projeto para acesso ao banco de dados e para a conta de e-mail.

Configurando Dados do Ambiente em Arquivos Locais

- No arquivo `application.properties` que é onde definimos as configurações do nosso projeto, perceba que há uma linha (**em negrito**) indicando ao spring que o projeto pode ter configurações definidas em um arquivo opcional cuja a extensão é `.env`

```
server.port=8082

spring.config.import=optional:file:.env[.properties]

# Datasource ( levantando a aplicação sem container )
spring.datasource.driver-class-name=org.postgresql.Driver
spring.datasource.url=jdbc:postgresql://localhost:5434/oxefood
spring.datasource.username=postgres
spring.datasource.password=oxefood

# JPA
spring.jpa.hibernate.ddl-auto=update
spring.jpa.show-sql=false
spring.jpa.properties.hibernate.dialect=org.hibernate.dialect.PostgreSQLDialect
spring.jpa.properties.hibernate.temp.use_jdbc_metadata_defaults = false
spring.jpa.database-platform=org.hibernate.dialect.PostgreSQL9Dialect
spring.jpa.generate-ddl=true
```

Configurando Dados do Ambiente em Arquivos Locais

- Com isso, podemos usar as variáveis definidas no arquivo `.env` dentro do nosso `application.properties`

```
server.port=8082

spring.config.import=optional:file:.env[.properties]

# Datasource ( levantando a aplicação sem container )
spring.datasource.driver-class-name=org.postgresql.Driver
spring.datasource.url=jdbc:postgresql://${HOST_DB}:${PORT_DB}/${NAME_DB}
spring.datasource.username=postgres
spring.datasource.password=${PASS_DB}

# JPA
spring.jpa.hibernate.ddl-auto=update
spring.jpa.show-sql=false
spring.jpa.properties.hibernate.dialect=org.hibernate.dialect.PostgreSQLDialect
spring.jpa.properties.hibernate.temp.use_jdbc_metadata_defaults = false
spring.jpa.database-platform=org.hibernate.dialect.PostgreSQL9Dialect
spring.jpa.generate-ddl=true
```

Configurando Dados do Ambiente em Arquivos Locais

- Também podemos utilizar as variáveis no arquivo `.docker-compose.yml`

```
version: "3"
services:

  db:
    image: postgres:9.6
    restart: always
    environment:
      POSTGRES_PASSWORD: ${PASS_DB}
      POSTGRES_DB: ${NAME_DB}
    ports:
      - ${PORT_DB}:5432
    volumes:
      - ./postgres-data:/bitnami/postgresql/data
```

Configurando Dados do Ambiente em Arquivos Locais

- Adicione o arquivo `.env` no arquivo `.gitignore` que está localizado na raiz do projeto

```
HELP.md
target/
!.mvn/wrapper/maven-wrapper.jar
!**/src/main/**/target/
!**/src/test/**/target/

### STS ###
.apt_generated
.classpath
.factorypath
.project
.settings
.springBeans
.sts4-cache

### IntelliJ IDEA ###
...

### VS Code ###
.vscode/

.env
```

Lembrando: arquivos e/ou diretórios adicionados ao `.gitignore` não são versionados no Git.

Configurando Dados do Ambiente em Arquivos Locais

- É interessante criar também um arquivo de template para o .env para que pessoas que baixarem o projeto do Git, saibam quais variáveis precisam ser preenchidas para a configuração inicial do ambiente.

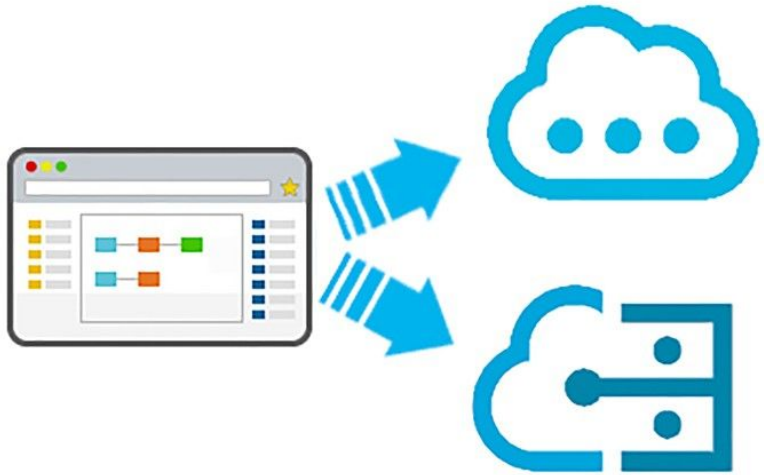
```
# Dados Banco de Dados

HOST_DB=localhost
PORT_DB=5434
NAME_DB=oxefood
PASS_DB=oxefood

# Dados do Serviço de Emails

EMAIL_USER=roberto@ifpe.edu.br
EMAIL_PASSWORD=123b123
```

Exemplo do arquivo `.env` que não será versionado pois está no `.gitignore`

```
# Dados Banco de Dados

HOST_DB=
PORT_DB=
NAME_DB=
PASS_DB=

# Dados do Serviço de Emails

EMAIL_USER=
EMAIL_PASSWORD=
```

Exemplo do arquivo `.env.template` que será versionado

Fazendo o Deploy do Projeto no Servidor



Fazendo o Deploy do Projeto Back-end no Servidor

- Pré-requisitos para o ambiente:
  - Instalar o JDK 17
  - Instalar o Git
  - Instalar o Docker / Docker-compose

- Para rodar o projeto siga os passos abaixo e execute os comandos:

  1) Baixe o projeto do repositório git, exemplo:

  ```
  git clone https://github.com/robertoalencar/oxefood-api.git
  ```

  2) Entre na pasta do projeto e execute o comando abaixo para levantar o banco de dados:

  ```
  docker-compose up -d
  ```

  3) Ainda na pasta do projeto, execute o comando abaixo para rodar o projeto:

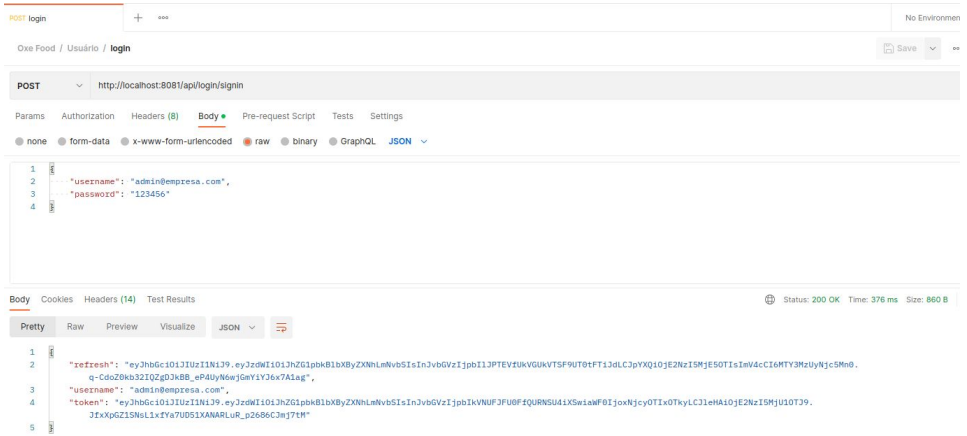
  ```
  ./mvnw spring-boot:run
  ```

Fazendo o Deploy do Projeto Back-end no Servidor

- Verifique se o projeto foi startado com sucesso no terminal

```
pring.jpa.open-in-view to disable this warning
2023-01-05 09:32:21.601  INFO 13027 --- [  restartedMain] o.s.b.w.embedded.tomcat.TomcatWebServer  : Tomcat started on port(s): 8081 (http) with context path ''
2023-01-05 09:32:21.797  INFO 13027 --- [  restartedMain] b.c.i.o.OxefoodNoiteApplication          : Started OxefoodNoiteApplication in 4.085 seconds (JVM running for 4.392)
2023-01-05 09:33:12.355  INFO 13027 --- [nio-8081-exec-2] o.a.c.c.C.[Tomcat].[localhost].[/]       : Initializing Spring DispatcherServlet 'dispatcherServlet'
2023-01-05 09:33:12.355  INFO 13027 --- [nio-8081-exec-2] o.s.web.servlet.DispatcherServlet        : Initializing Servlet 'dispatcherServlet'
2023-01-05 09:33:12.357  INFO 13027 --- [nio-8081-exec-2] o.s.web.servlet.DispatcherServlet        : Completed initialization in 1 ms
2023-01-05 09:33:12.363 DEBUG 13027 --- [nio-8081-exec-2] o.s.security.web.FilterChainProxy        : Securing POST /api/login/signin
2023-01-05 09:33:12.365 DEBUG 13027 --- [nio-8081-exec-2] s.s.w.c.SecurityContextPersistenceFilter : Set SecurityContextHolder to empty SecurityContext
2023-01-05 09:33:12.371 DEBUG 13027 --- [nio-8081-exec-2] o.s.s.w.a.AnonymousAuthenticationFilter  : Set SecurityContextHolder to anonymous SecurityContext
2023-01-05 09:33:12.375 DEBUG 13027 --- [nio-8081-exec-2] o.s.s.w.a.i.FilterSecurityInterceptor    : Authorized filter invocation [POST /api/login/signin] with attributes [permitAll]
2023-01-05 09:33:12.375 DEBUG 13027 --- [nio-8081-exec-2] o.s.security.web.FilterChainProxy        : Secured POST /api/login/signin
2023-01-05 09:33:12.615 DEBUG 13027 --- [nio-8081-exec-2] o.s.s.a.dao.DaoAuthenticationProvider    : Authenticated user
2023-01-05 09:33:12.680 DEBUG 13027 --- [nio-8081-exec-2] s.s.w.c.SecurityContextPersistenceFilter : Cleared SecurityContextHolder to complete request
```

- Teste o projeto via Postman



Fazendo o Deploy do Projeto Front-end no Servidor

- Pré-requisitos para o ambiente:
  - Instalar o NodeJS
  - Instalar o Git

- Para rodar o projeto siga os passos abaixo e execute os comandos:

  1) Baixe o projeto do repositório git, exemplo:

  ```
  git clone https://github.com/robertoalencar/oxefood-web.git
  ```

  2) Entre na pasta do projeto e execute o comando abaixo para rodar o projeto:

  ```
  npm start
  ```

Dúvidas

Obrigado !