COMO CRIAR ARQUIVOS DE OCNFIGURAÇÃO E DEFINIR OS PROFILES:

1. CRIAR ARQUIVOS “PROPERTIES”
   1. Neste projeto, na pasta src>java>resources foram criados os arquivos **application-dev.properties** e **application-prod.properties** para indicar os perfis de desenvolvimento e produção respectivamente. No arquivo **application.properties** são adicionadas configurações padrão ou comuns aos dois perfis.
   2. Em seguida, foi criado um pacote “config”, dentro do pacote “com.digitalinnovationone.sprintbootconfiguration” e em seguida, foi criada a class de configuração “DBconfiguration”.
      1. Dentro da class “DBConfiguration” foi utilizada a anotação @Configuration para indicar ao spring que se trata de um arquivo de configuração.
      2. Em seguida, foi utilizada a anotação @ConfigurationProperties para indicar o prefixo das configurações que serão alteradas. No nosso caso, o prefixo era “spring.datasource”. Isto significa que os atributos iniciados com “spring.datasource” serão alterados. Em nosso caso, foram os atributos “driverClassName”,“url”, “username” e “password”, como podemos ver no **application-dev.properties** e **application-prod.properties**
      3. Para a correta utilização e manipulação dos atributos, funções GETTER and SETTER devem ser configuradas. No intuito de simplificar essa utilização, foi instalado o lombok, como uma das dependências do projeto. Para a instalação do lombok, foi acrescentada a seguinte dependência no arquivo pom.xml:

<dependency>

<groupId>org.projectlombok</groupId>

<artifactId>lombok</artifactId>

<version>1.18.22</version>

<scope>provided</scope>

</dependency>

O lombok permite a utilização da anotação @GETTER E @SETTER ao invés de criarmos todas as funções getters and setters para cada atributo.

* + 1. Em seguida, foi utilizada a anotação @Profile(“<perfil>”) para indicar que, caso o perfil ativo seja o indicado em <perfil>, a função logo abaixo da anotação será executada. Caso contrário, a função não será executada.
    2. Foi utilizada a anotação @Bean, para indicar que a execução da função subsequente deve ser realizada no início do carregamento do spring.
  1. Para a correta simulação, foi criada a class AppController, como a anotação @RestController, para indicar que é um controlador de requisições Rest.
     1. Ainda em AppController, foi definido o atributo appMessage e utilizada a anotação @Value (“<valor>”). Essa anotação indica que o atributo receberá o conteúdo indicado em <valor>. Em nosso caso,o <valor> refere-se à variável app.message, que é instanciada nos arquivos **application-dev.properties** e **application-prod.properties.** Isto significa que app.message recebe valores diferentes conforme o perfil ativo (“dev” ou “prod”), e esse valor será repassado ao atributo appMessage.
     2. A anotação @GetMapping(“<rota>”) é utilizada para construção de rotas no spring. Em nosso caso, foi estabelecida a rota padrão “/”.