Lista 25 – Spring Rest

1. Na aula 55, do dia 17.02, foi desenvolvido um *web service* que recebia 2 números e os somava. Foram feitas versões do *web service* funcionando por meio de *query params* e também por meio de *path params*. A apresentação de apoio a este conteúdo é a 'aula 46b - spring rest (02.02.2023)'. Baseado na aula e na apresentação, crie uma aplicação consistindo de uma API Rest e seguindo as instruções abaixo:
    a. Crie uma aplicação Spring Boot com os *starters* DevTools e Spring Web.
    b. Além da soma, a API deve implementar as seguintes operações matemáticas: Subtração, multiplicação, divisão, média entre 2 números, potenciação e raiz quadrada. Cada operação deverá ser implementada em um *end-point* próprio.
    c. Faça duas versões de cada método, uma com *query params* e outra com *path params*.
    d. Use os métodos `ehNumerico()` e `converteParaDouble()` da apresentação de forma a aceitar valores decimais com ponto ou vírgula.
    e. O papel do controlador é somente tratar as requisições e não realizar tarefas de conversão numérica. Portanto, os métodos `ehNumerico()` e `converteParaDouble()` deveriam estar em outras classes. Crie uma outra classe chamada `ConversorNumerico` e mova esses métodos para ela. Declare esses métodos como estáticos de forma que possam ser chamados diretamente pela classe `ConversorNumerico` sem necessidade de fazer sua instanciação. Refatore a classe *controller* de forma a utilizar a classe `ConversorNumerico`.
    f. Da mesma forma, o controlador não deveria estar realizando as operações matemáticas propriamente ditas, como `converteParaDouble(numero1) + converteParaDouble(numero2)`. As operações deveriam ser realizadas por uma classe específica. Crie uma classe chamada `Calculadora` que implemente as operações matemáticas do item (a). Refatore a classe *controller* de forma a chamar os métodos da classe `Calculadora` para realizar as operações matemáticas.
    g. A raiz quadrada de um número negativo gera um `NaN` (*Not a Number*), a divisão por 0.0 (ponto flutuante) gera um resultado do tipo `Infinity` e a divisão por 0 (inteiro) lança uma exceção do tipo `java.lang.ArithmeticException`. Verifique essas condições nos métodos da classe Calculadora que implementam a divisão e a raiz quadrada. Ao detectar essas condições especiais os métodos deverão lançar exceções (`throws`), que por sua vez devem ser tratadas no *controller* com o bloco `try...catch`. Comece utilizando classes de exceção padrão do Java e depois crie suas próprias exceções personalizadas. Para relembrar como fazer isso, reveja a apresentação 'aula 33a - tratamento de exceções (18.08.2022)'.
    h. Use a classe `Logger` do pacote `java.util.logging` para mostrar os resultados também no console.
    i. Organize as classes em pacotes específicos, para uma melhor organização do código.
    j. Utilize o Postman para todos os métodos desta API. Faça um *print* das telas do Postman em um arquivo e exporte-o em PDF para posterior entrega.

2. Após finalizar o sistema desenvolvido no item 1 acima e se certificar que ele está funcionando corretamente, crie um repositório no GitHub para armazená-lo na nuvem. Utilize o GitBash para criar um repositório local e conectá-lo ao repositório no GitHub. Faça o *commit* com a versão atual do sistema e então realize o *push* para enviar o projeto para o repositório remoto.