

Relatório Desafio Back-end

Rafael Niederauer Meyer

Universidade Federal de Santa Catarina – UFSC

Florianópolis, 18 de junho de 2021

Sumário

1. Introdução
2. Código
 - 2.1 Front-end
 - 2.2 Back-end
3. Dificuldades
4. Conclusão e informações
5. Como rodar o projeto

1. Introdução

Este Relatório corresponde em analisar o código e suas lógicas por trás de cada linha escrita e também analisar as dificuldades obtidas ao longo do desafio.

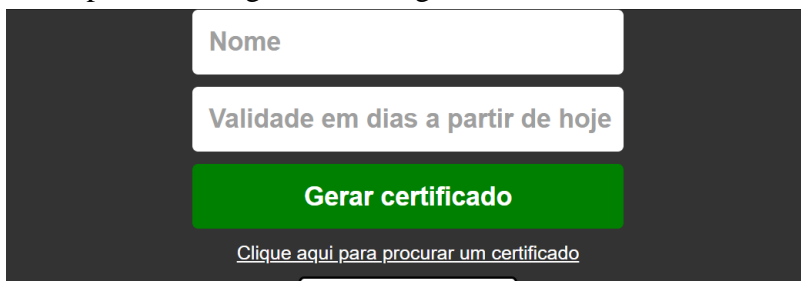
2. Código

O código está “ligado” em duas partes: O front-end e o back-end.

2.1 Front-end

Para o front-end foi utilizada a ferramenta React (a qual foi requerida no desafio).

Nesta parte, o código roda da seguinte maneira:

A imagem mostra uma interface de usuário com um fundo escuro. No topo, há um campo de entrada branco com o rótulo "Nome". Abaixo dele, outro campo de entrada branco com o rótulo "Validade em dias a partir de hoje". Um botão verde com o texto "Gerar certificado" em branco está posicionado abaixo dos campos. Na base da interface, há um link de texto "Clique aqui para procurar um certificado" em uma cor clara.

O usuário irá digitar o nome requerido e a quantidade de dias a partir da data atual (na data da escrita no caso). Após isso, ele irá clicar em gerar certificado e o certificado será gerado!

```

17 function generate(){
18   if(name==""||validate==""){
19     alert("Preencha todos os campos")
20     return
21   }
22   if(!Number.isInteger(parseFloat(validate))){
23     alert("Preencha a validade apenas com número e inteiro!")
24     return
25   }
26   setRequesting(true)
27   const body={name,validate}
28   const request = axios.post("http://localhost:8080/Project-1.0-SNAPSHOT/create-certificate", body)
29   request.then(r=>{
30     setTimeout(() => {
31       setRequesting(false)
32       setName("")
33       setValidate("")
34       alert("Certificado gerado!")
35     }, 2000)
36   })
37   request.catch()
38 }

```

Nesta parte é gerado um request com o método post para enviar os dados (nome e dias de validade) ao servidor.

The image shows a web form for searching certificates. It has a dark background. At the top, there is a white input field labeled "Nome". Below it are four more white input fields labeled "Dia de início", "Mês de início", "Dia de fim", and "Mês de fim". Below these is a prominent red button with white text that says "Procurar certificado". At the bottom, there are two lines of text: "Clique aqui para gerar um certificado" and "Clique nas informações abaixo para deletar o certificado clicado!".

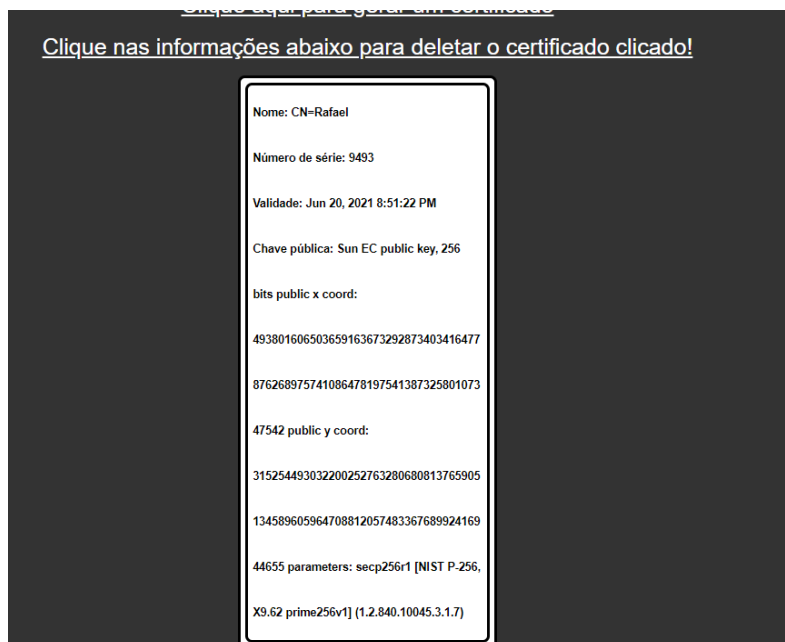
Na parte de busca, o usuário pode pesquisar apenas o nome do titular ou pesquisar o nome junto com um intervalo de validade (todos os intervalos colocados no ano de 2021 para fins de otimização de tempo). Após preencher os campos necessários, o usuário irá clicar em “Procurar certificado” e logo abaixo irão aparecer os certificados que deram “match” na busca.

```

39 function searchCertificated(){
40
41     var inDate = new Date(2021, parseInt(inMonth)-1, parseInt(inDay))
42     var finalDate = new Date(2021, parseInt(finalMonth)-1, parseInt(finalDay))
43     finalDate.setHours(23,59,59)
44
45     var inDateTime = inDate.getTime().toString()
46     var finalDateTime = finalDate.getTime().toString()
47
48     setRequesting(true)
49     const body={name, inDateTime, finalDateTime}
50     const request = axios.post("http://localhost:8080/Project-1.0-SNAPSHOT/search-data", body)
51     request.then(r=>{
52         setTimeout(() => {
53             setRequesting(false)
54             setName("")
55             setInDay("")
56             setInMonth("")
57             setFinalDay("")
58             setFinalMonth("")
59         }, 1000)
60         setJsonOb(r.data)
61     })
62     request.catch()
63 }
64

```

Na parte de busca no código, são enviadas as informações necessárias para o servidor e após isso é retornado do banco de dados os certificados buscados.



Para deletar algum certificado, basta clicar em cima da “Div” do certificado e ele será deletado!

2.2 Back-end

Sendo esta a parte mais desafiadora (envolvendo bug atras de bug :D), acabou tudo dando certo.

Para a parte do certificado e das chaves, foi utilizado arquivos de outro desafio do labSec envolvendo criptografia (o qual eu não tinha conseguido terminar, porém nesse desafio deu tudo certo!).

A classe CORSfilter foi utilizada para filtrar o “Cross-Origin Resource Sharing”, pois como estou utilizando duas portas (:3000 para rodar o front-end e :8080 para o back-end), foi necessário a criação desta classe.

A classe Create certificate é a qual cria os certificados e insere em um banco de dados.

3. Dificuldades

Eu tive bastante dificuldade na parte do back-end (o qual foi o que mais tirou meu tempo). Nunca tinha criado um servidor, ainda mais em java! Com certeza ficaram faltando alguns (vários) refinamentos, porém não consegui refinar em quesito ao tempo. Todos os requisitos do trabalho foram executados.

4. Conclusão e informações

Concluo com este trabalho que mesmo com grandes dificuldades, as tarefas foram cumpridas (mesmo faltando alguns refinamentos). Na parte de gerar um banco de dados, certificado e o par de chaves, foi colocado o caminho de arquivo do meu computador para facilitar o desenvolvimento durante o desafio. O mesmo serve para configurar o servidor (glassfish 5). Ainda não testei o Docker pois nunca trabalhei com esta ferramenta. Vou testar agora que sobrou um tempinho, e caso eu não entregue com o Docker, pois bem; D, não deu certo!

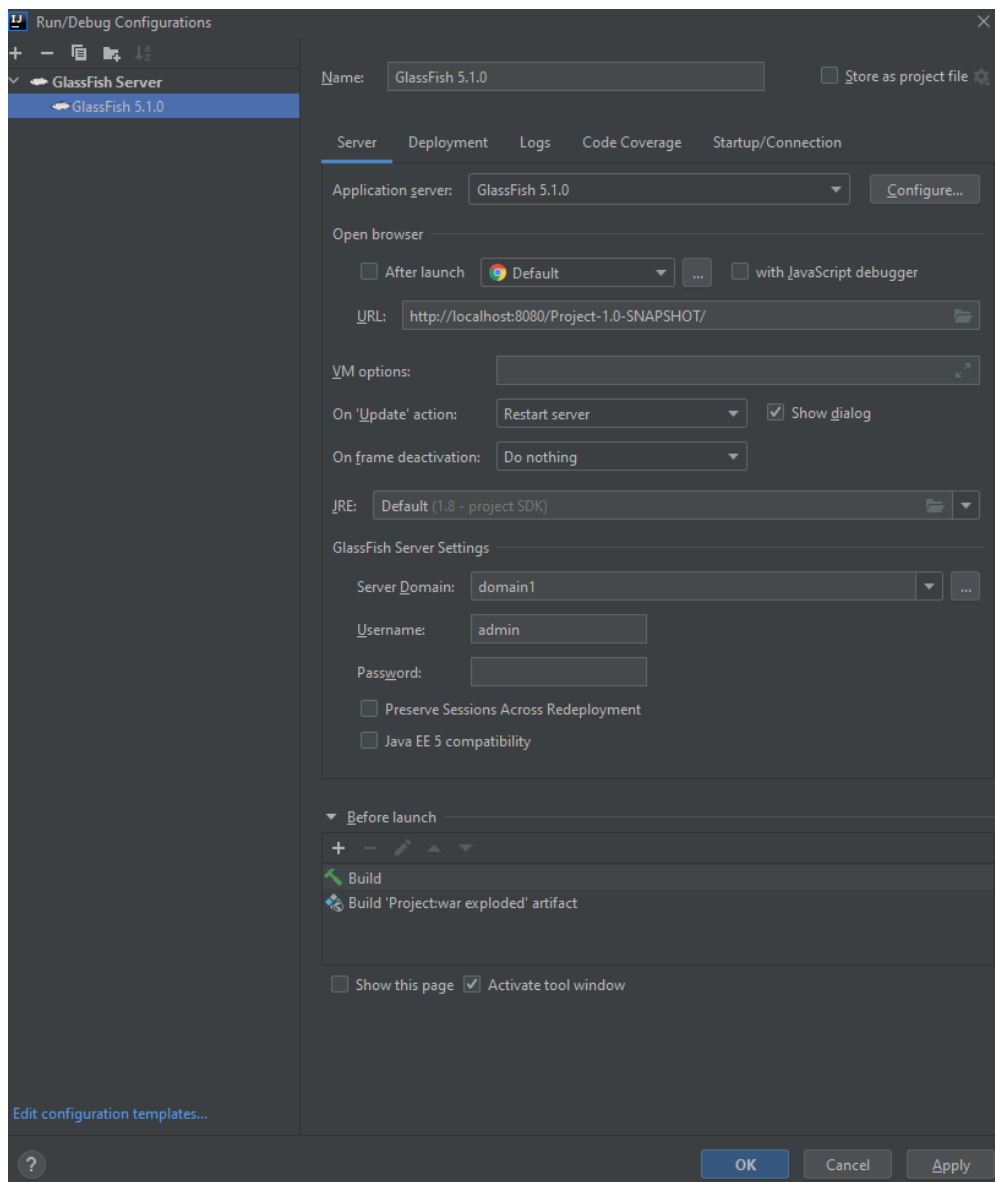
5. Como rodar o projeto

Você irá rodar o projeto em duas portas: o React na porta 3000 e o servidor (glassfish) na porta 8080.

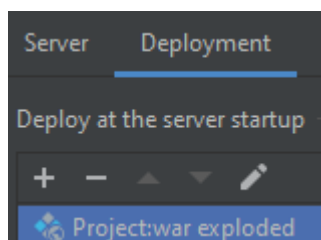
Para rodar o React basta estar no terminal e no diretório onde contém o React (/ui) e escrever a linha de código: “npm start”. Assim o React vai iniciar na porta 3000.

Agora para rodar o servidor você deve baixar o glassfish 5.1.0 e configurá-lo da seguinte maneira (utilizando a interface intellij):

Você deve clicar em run->edit configurations e adicionar o glassfish local.



Após isso você deve clicar em deployment e adicionar o artefato “Project:war exploded”.



Se tudo ocorrer corretamente, basta inicializar o glassfish pelo atalho alt+shift+f10 e rodar ele. E assim ele irá rodar na porta 8080.

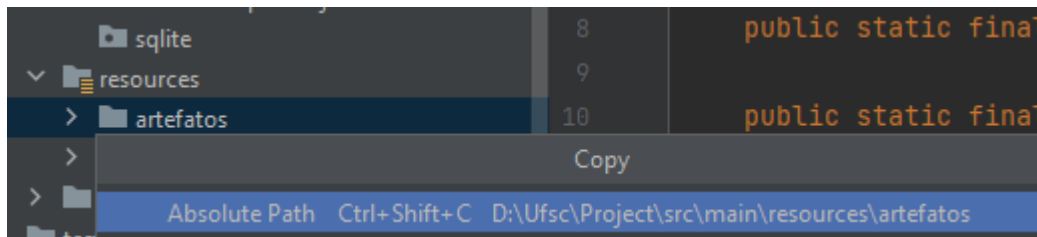
Caso ocorra durante a execução erros de diretório, basta mudar o diretório no arquivo “Constantes.java”

```

1 package br.ufsc.labsec.pbad.hiring;
2
3 public class Constantes {
4
5     public static final String algoritmoChave = "EC";
6     public static final String formatoCertificado = "X.509";
7
8     public static final String caminhoDb = "jdbc:sqlite:C://sqlite/db/certificateInfos.db";
9
10    public static final String caminhoArtefatos =
11        "D:/Ufsc/Project/src/main/resources/artefatos/";
12
13    public static final String caminhoChavePublicaUsuario =
14        caminhoArtefatos + "chaves/";
15    public static final String caminhoChavePrivadaUsuario =
16        caminhoArtefatos + "chaves/";
17
18    public static final String caminhoCertificadoUsuario =
19        caminhoArtefatos + "certificados/";
20 }

```

Você pode tanto mudar o caminho do banco de dados como o caminho dos certificados (caminhoArtefatos). Provavelmente quem for testar terá que mudar o caminhoArtefatos. Eu recomendo clicar com o botão direito na pasta artefatos e copiar o caminho absoluto do arquivo e colar no código (com mostra a imagem abaixo).



E se ocorrer algum erro no caminho do bando de dados mesmo após trocar o local do arquivo, recomendo deixar o mesmo diretório da pasta e criar as pastas necessárias para o caminho igual o da foto.