Relatório Técnico [Marcos Goes]

Revisão do Plano de Trabalho

- Como previsto utilizei Java para desenvolvimento da aplicação backend.

- A utilização do framework Twitter4J me ajudou no sentido de abstrair toda a complexidade de estabelecer conexão REST com a API do Twitter, inclusive no que diz respeito à implementação de segurança OAuth.

- Foi necessário criar uma conta de Developer para obter os tokens de utilização da API do Twitter.

- Não consegui tempo suficiente para pesquisar sobre MongoDB que era a minha intenção de NoSQL e percebendo que usar um banco de dados relacional também não seria uma boa opção, optei pela simplicidade em manter tudo em memória.

- Inicialmente eu nem estava considerando o uso do Docker para montagem do ambiente pois nunca o tinha utilizado antes. Pesquisei um pouco e achei muito fácil de trabalhar. Criei uma imagem baseada no Ubuntu e fiz os scripts para instalar tudo o resto que eu precisaria. Provavelmente não está da melhor forma mas está funcionando.

- A camada front-end web estava inicialmente planejada para ser feita em Spring MVC mas para isso eu teria que escrever uma outra aplicação. No backend eu sempre prefiro usar EJB e CDI, por isso acabo usando JEE. Já no frontend quando uso Spring não gosto de misturá-lo com o JEE em razão do framework de injeção de dependência do Spring trabalhar muito melhor com o próprio Spring do que o CDI. Optei então por usar AngularJS rodando na mesma aplicação do backend. Entendo que seria melhor colocá-lo em outra camada, num Apache por exemplo mas deixei tudo junto para facilitar.

- Sobre uma solução de API Gateway eu havia optado por utilizar o Apiman em conjunto com o Keycloak, ferramentas que podem ser totalmente integradas ao Wildfly. Cheguei a instalá-lo e eles estão inclusive disponíveis na imagem do Docker mas não consegui utilizá-los. Apesar da interface bastante amigável eu percebi que teria que fazer muitas configurações e não consegui pesquisar sobre tudo.

Tecnologias utilizadas

Sistema Operacional Desenvolvimento: Linux Mint 18.1
Sistema Operacional Docker: Linux Ubuntu 16.04
IDE: Eclipse Neon R.2

Linguagem Backend: Java JDK8
Frameworks Backend: Twitter4J, CDI, JAX-RS, EJB 3.1, Maven
Frameworks Frontend: AngularJS, Bootstrap
Servidor de Aplicação: Wildfly 10.1.0

Modelagem da base de dados

Não se aplica pois não utilizei nenhuma base de dados. A natureza dos dados que estavam sendo trabalhados não condiziam com uma abordagem relacional. Certamente a solução NoSQL proposta faria mais sentido mas eu não tenho conhecimento suficiente e não houve tempo para pesquisa. Os dados para pesquisa são mantidos em memória na própria aplicação.

Código fonte

Disponível no meu perfil do GitHub:
https://github.com/marcos-goes/tweet-reader-backend

Imagem Docker

Disponível no meu perfil do DockerHub:
https://hub.docker.com/r/mpsgoes/tweet-reader/

```
Terminal:
docker pull mpsgoes/tweet-reader
docker run -it --rm -p 8080:8080 mpsgoes/tweet-reader

Browser:
http://localhost:8080/tweet-reader
```