Sistemas Distribuídos

Projeto TP1 (versão: 5/Abril)

Engenharia Informática

Ano lectivo: 2018/2019, 2º Semestre

Prazos

```
1ª Trabalho - 2 de Maio, 23h59 (online - código + relatório/formulário)
2ª Trabalho - 5 de Junho, 23h59 (online - código + relatório/formulário)
```

Objetivo

O objetivo do trabalho consiste em desenvolver a infra-estrutura de suporte de uma aplicação móvel, inspirada na rede social Instagram, doravante denominada Microgram.

Através da aplicação Microgram, os utilizadores podem tirar e publicar fotografias.

Podem procurar outros utilizadores e ver as suas fotografias; e, eventualmente, marcá-las com um "gosto".

Tratando-se de uma rede social, um utilizador tanto pode seguir, como ser seguido por outros utilizadores.

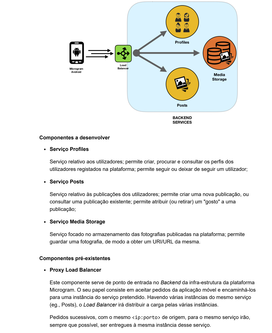
Em cada momento, a aplicação permite visualizar as fotografias publicadas pelo próprio, bem como a sua *feed* - a lista das publicações dos utilizadores seguidos pelo utilizador.

Cada utilizador tem um perfil público, que além de dados pessoais sumariza quantas publicações o utilizador já fez, quantos utilizadores segue e por quantos utilizadores é seguido.

Arquitetura

A infra-estrutura da aplicação móvel Microgram, estará organizada numa arquitecura em camadas. Considerando a aplicação móvel a camada de apresentação, o trabalho terá como alvo, exclusivamente, as duas camadas seguintes: a camada aplicacional e a camada de armazenamento, eventualmente fundidas numa só - o denominado *backend* da aplicação móvel, constituído pelos serviços que dão suporte às funcionalidades da aplicação.

A figura seguinte fornece uma visão global de alto-nível da arquitectura da aplicação...



Componentes a desenvolver

- **Serviço Profiles**

  Serviço relativo aos utilizadores; permite criar, procurar e consultar os perfis dos utilizadores registados na plataforma; permite seguir ou deixar de seguir um utilizador;

- **Serviço Posts**

  Serviço relativo às publicações dos utilizadores; permite criar uma nova publicação, ou uma nova publicação existente; permite atribuir (ou retirar) um "gosto" a uma publicação;

- **Serviço Media Storage**

  Serviço focado no armazenamento das fotografias publicadas na plataforma; permite guardar uma fotografia, de modo a obter um URI/URL da mesma.

Componentes pré-existentes

- **Proxy Load Balancer**

  Este componente serve de ponto de entrada no *Backend* da infra-estrutura da plataforma Microgram. O seu papel consiste em aceitar pedidos da aplicação móvel e encaminhá-los para uma instância do serviço pretendido. Havendo várias instâncias do mesmo serviço (eg., Posts), o *Load Balancer* irá distribuir a carga pelas várias instâncias.

  Pedidos sucessivos, com o mesmo `<ip:porto>` de origem, para o mesmo serviço irão, sempre que possível, ser entregues à mesma instância desse serviço.

- **Aplicação Microgram**

  Aplicação utilizada pelo utilizador final da plataforma Microgram. A interação do utilizador com a aplicação resulta em variados pedidos à infra-estrutura de suporte por intermédio do *Proxy Load Balancer*.

Interfaces de programação

Os serviços *backend* da aplicação Microgram serão desenvolvidos por recurso a tecnologia REST (JAX-RS) e (opcionalmente) através de WebServices SOAP (JAX-WS), exceto o Serviço Media Storage que apenas terá uma versão REST.

Para garantir a inter-operabilidade com os componentes pré-existentes, os serviços terão que ser implementados respeitando interfaces de programação pré-definidas e os efeitos esperados das suas operações.

A especificação dos serviços de *backend* está disponível num repositório externo, onde serão publicadas eventuais correções ou esclarecimentos, caso surja essa necessidade.

O repositório contém um projeto Eclipse (Java) com os seguintes pacotes:

- **microgram.api** - Definições de Profile e Post;
- **microgram.api.java** - Documenta de forma genérica a semântica das operações e os erros a reportar;
- **microgram.api.rest** - Documenta a API Rest dos serviços de backend;
- **microgram.api.soap** - Documenta a API SOAP dos serviços de backend.

Nota: As interfaces podem ser modificadas apenas introduzindo novas operações, ou através de parâmetros opcionais nas operações pré-existentes).

Auto-configuração

Os serviços de *backend* deverão ser capazes de autoconfigurar-se, não podendo depender de endereços IP fixos.

Para tal, deverão implementar um mecanismo de descoberta baseado em **anúncios periodicos** (por parte dos servidores) e comunicação IP multicast. Este mecanismo também será o meio utilizado pelo componente *Proxy Load Balancer* para descobrir as instâncias dos serviços de backend presentes na rede local.

O protocolo de descoberta consiste em enviar periodicamente para o endereço IP multicast e porto pré-acordados, uma mensagem, contendo uma string, com o seguinte formato:

```
<nome-do-serviço>;<uri-do-servidor>
```

O campo `<nome-do-serviço>` a usar em cada caso, será: *Microgram-Profiles*, *Microgram-Posts*, *Microgram-MediaStorage*.

O `<uri-do-servidor>` deve terminar com `/rest` ou `/soap` para indicar, respetivamente, que se trata de um servidor REST ou SOAP.

Requisitos da solução

Premissas gerais

O foco deste primeiro trabalho será centrado nas tecnologias de Invocação Remota na Web e na Distribuição.

Cada um dos três serviços do *backend* deverá ser suportado por servidores dedicados, sendo de evitar uma solução monolítica.

A concepção da solução deverá aspirar a atender aos requisitos de escalabilidade da aplicação, procurando incluir uma estratégia de distribuição visando servir um número elevado de utilizadores e de dados.

A solução não precisa de tolerar falhas de componentes. As únicas falhas a contemplar são as falhas (temporárias) de comunicação. (Não há, assim, necessidade de introduzir replicação de componentes. Essa temática será tratada no segundo trabalho.)

Os serviços Profiles e Posts não necessitam de persistir o seu estado, podendo manter os seus dados em memória.

A compatibilidade com as interfaces e operações pré-definidas tem que ser observada. No entanto, poderá ser necessário adicionar mais operações para atender alguns dos requisitos da solução.

O serviço Media Storage não será valorizado, uma vez que será usado nos materiais de apoio às aulas prática para ilustrar as matérias lecionadas.

Requisitos mínimos (max: 9.5 valores)

- **API REST** - Serviços de *backend* versão REST funcionais quando considerados isoladamente.
- **Controlo de concorrência** - Os serviços atendem vários clientes em simultâneo de forma correta.

Requisitos base (max: 14 valores)

- **Auto-configuração** - A descoberta por multicast funciona corretamente, permitindo ao *Proxy Load Balancer* encontrar os servidores;
- **Funcionalidade completa** - Serviços de backend REST funcionais e integrados, implementados em servidores separados.

  A percepção dos utilizadores da aplicação Microgram é a de um sistema coerente e integrado. As ações dos utilizadores repercutem-se no sistema de acordo com as expetativas, incluindo na evolução das estatísticas associadas aos dados (*Posts* e *Profiles*)

Nota: Os requisitos base têm como pressupostos os requisitos mínimos;

Elementos valorativos (max: 20 valores)

- **WebServices Soap** - O sistema funciona misturando versões REST e SOAP dos serviços Posts e Profiles (€);
- **Garbage-Collection** - o sistema é capaz de proativamente eliminar informação obsoleta. Por exemplo, caso um utilizador seja removido do sistema, as suas publicações serão igualmente removidas e as estatísticas dos outros utilizadores atualizadas (€).

[**Adenda:** A funcionalidade de garbage-collection não é referente à remoção de conteúdos noutros servidores como parte da execução de uma operação (e.g., o facto da remoção de um profile levar a que se remova também todos os posts desse profile (entre outras coisas) não é uma operação de garbage-collection). O Garbage-collection pretende resolver problemas que emergem, por exemplo, devido à falta de atomicidade entre operações sob diferentes serviços. E.g., um utilizador pretende fazer um post, guarda uma imagem no servidor de Media, mas falha antes de conseguir criar o Post que refere a imagem já adicionada. Nesta situação a imagem adicionada pelo utilizador não têm nenhuma referência em nenhum Post, e consequentemente o sistema pode removê-la automaticamente.]

- **Distribuição** - Existe uma estratégia adequada para melhorar a escalabilidade da plataforma, com base na distribuição/descentralização de um dos serviços Posts/Profiles (à escolha):
  - A cotação máxima só é possível com uma solução adequada neste âmbito (€€€ €);
  - A solução pode assumir que o número de instâncias do serviço alvo é fixo e conhecido à priori.

[**Adenda**: O número de servidores de cada tipo são comunicados a cada serviço através de argumentos (opcionais) passados à função main na linha de comandos. Em particular para indicar o número de servidores de Profiles é usada a notação -profiles n (onde n é o número total de servidores de Profiles), sendo que para indicar o número de servidores de Posts é usada a notação -posts n. A ordem pela qual os argumentos são passados é variavel.]

Nota: Os elementos valorativos não serão considerados em pleno caso os requisitos base não sejam atingidos de forma satisfatória. Não é necessário realizar todos os elementos valorativos para atingir a cotação máxima.

Fatores depreciativos

- O código entregue deverá seguir boas práticas de programação. A repetição desnecessária de código, inclusão de constantes mágicas, o uso de estruturas de dados inadequadas, etc., poderá incorrer numa penalização.
  (max: 2 valores)
- Falta de robustez e comportamento erráticos da solução são motivo para penalização.
  (max: variável)
  - A solução deve contemplar as falhas temporárias dos canais de comunicação.

Execução

O trabalho pode ser executado em grupo, de 1 ou 2 alunos. Os alunos do mesmo grupo não precisam de pertencer ao mesmo turno prático.

Avaliação

A avaliação do trabalho terá em conta os seguintes critérios:

- Funcionalidades desenvolvidas e a sua conformidade com a especificação, tendo como base os resultados da bateria de testes automáticos;
- Qualidade da solução, podendo incluir uma avaliação de desempenho;
- Qualidade do código desenvolvido.

A classificação final do aluno é individual e será menor ou igual à classificação do trabalho, em função dos resultados obtidos na discussão individual, a realizar no final do semestre.

Bateria de testes

Oportunamente será disponibilizada uma bateria de testes destinada a verificar a conformidade da solução com a especificação.

A solução deverá ser compatível com Java 8 (1.8), sob pena de poder falhar os testes.

Aplicação Microgram + Proxy Load Balancer

Estes componentes não são necessários para resolver o trabalho em pleno.

Devido à política restritiva de gestão de tráfego da rede WIFI dos laboratórios, a aplicação android Microgram e o Proxy Load-balancer só serão disponibilizados por pedido expresso do grupo.

Material de apoio

A realização do trabalho poderá recorrer ao material de apoio disponível neste repositório. A utilização deste código no todo ou em parte é opcional.

O repositório contém um esqueleto da solução, incluindo código que:

- Implementa algumas das operações sobre os dados da plataforma;
- Ajuda a fatorizar as partes em comum entre as versões REST e SOAP dos serviços;
- Mostra como evitar código repetido no tratamento das falhas de comunicação.
- O código fornecido recorre a tipos de dados genéricos, interfaces funcionais, expressões lambda e Java streams. Este código será também a base da solução de referência que os alunos poderão adaptar (se assim entenderem) para realizarem o segundo trabalho.

O trabalho pode recorrer a outros componentes e bibliotecas já existentes e evitar re-inventar a roda. Por exemplo, se uma cache valorizar algum aspecto do trabalho, pode-se recorrer à biblioteca Google Guava para o efeito. Codificar de raiz essa cache não será valorizado.

Ambiente de desenvolvimento

Todo o material de apoio fornecido pressupõe que o desenvolvimento será em ambiente Linux e Java 8. A validação do trabalho por via da bateria de testes automática terá uso de tecnologia Docker.

Na falta de um ambiente Linux nativo, recomenda-se o recurso a uma solução de virtualização, em particular aos utilizadores Windows. Chama-se à atenção que os alunos têm acesso gratuito aos produtos VMWare. Em alternativa, poderão utilizar software opensource VirtualBox.

Entrega

A entrega do trabalho será feita através do preenchimento de um formulário Google Forms, dentro do prazo previsto.

O código da solução deverá ser depositado num repositório **privado** na plataforma GitHub, indicando no formulário de entrega o identificador do comando da versão final.

Será ainda necessário adicionar como colaborador do repositório, o utilizador **smduarte**.

Formulário de entrega

Experiência de utilização

O vídeo abaixo captura um exemplo da experiência do utilizador da aplicação Microgram.