Número: Nome:

Sistemas Distribuídos

Faculdade de Ciências e Tecnologia
2017/18
Teste 1
sem consulta - 90 minutos

Questão 1

Pretende-se desenvolver um serviço web para gerir uma lista de contactos que será implementado por recurso a tecnologias REST e SOAP. As operações a disponibilizar são: a) criar um novo contacto; b) modificar um contacto existente, dado o seu nome; c) obter a informação de um contacto, dado o seu nome; d) eliminar um contacto, dado o seu nome; e) obter todos os contactos cujo nome começa com um dado prefixo. As interfaces incompletas do serviço, com estas operações, encontram-se nos anexos A e B. Um contacto é modelado pela classe `Contact`, omitida.

Onde aplicável, as operações descritas devem retornar um erro para o cliente caso o contacto não exista. Em REST, o erro será assinalado com o código HTTP `404 NOT FOUND`, em SOAP com a exceção `ContactDoesNotExist` (Anexo B). Ainda em SOAP, se qualquer dos argumentos for `null`, deve ser produzida a exceção `InvalidArgumentException`.

a) Complete o Anexo A, como achar necessário, sabendo que as operações listadas acima utilizam os seguintes URLs para um servidor a executar na máquina local no porto 8080.

i) `http://localhost:8080/contacts/`

ii) `http://localhost:8080/contacts/{name}`

iii) `http://localhost:8080/contacts/{name}`

iv) `http://localhost:8080/contacts/{name}`

v) `http://localhost:8080/contacts/list`

b) Complete o Anexo B, como achar necessário.

c) O anexo C contém o código (incorreto) de um cliente REST do serviço para criar um novo contacto. Após lançar o servidor na máquina local, como resultado da execução do cliente é obtido um traço de uma exceção (Anexo C). Indique, sucintamente, a razão do erro observado e como este pode ser corrigido, assumindo que o servidor está correto.

d) O anexo D contém parte do código do servidor REST e de um cliente que pretende remover um contacto da lista de contactos. Suponha que existem duas máquinas onde podem ser lançados o servidor e o cliente, em qualquer combinação. Estas máquinas possuem, respetivamente, os IPs 192.168.1.1 e 192.168.1.2, além das interfaces com endereço 127.0.0.1 (em ambas máquinas). Para cada uma das combinações (A) e (B), da tabela abaixo, depois de aplicadas no código do cliente e do servidor (Anexo D), identifique o resultado da execução do cliente de acordo com a chave abaixo.

R - **Sucede** mesmo que o cliente seja **remoto** relativamente ao servidor;

L - **Sucede** apenas se o cliente for **local** relativamente ao servidor;

F - **Falha** independentemente da localização do cliente face ao servidor.

A: 127.0.0.1	B: 127.0.0.1		A: 192.168.1.1	B: 127.0.0.1	
A: 127.0.0.1	B: 192.168.1.1		A: 192.168.1.1	B: 192.168.1.1	
A: 127.0.0.1	B: 192.168.1.2		A: 192.168.1.1	B: 192.168.1.2	
A: 0.0.0.0	B: 127.0.0.1		A: 192.168.1.2	B: 127.0.0.1	
A: 0.0.0.0	B: 192.168.1.1		A: 192.168.1.2	B: 192.168.1.1	
A: 0.0.0.0	B: 192.168.1.2		A: 192.168.1.2	B: 192.168.1.2	

Número: Nome:

Questão 2

Para cada uma das seguintes afirmações indique se é `[V]`erdadeira ou `[F]`alsa. Em caso de dúvida, justifique a resposta na página em branco no final do teste. (nota: respostas incorretas não justificadas descontam.)

a) —— Um sistema distribuído consiste num conjunto de componentes de software e hardware que se coordenam entre si através da partilha de memória.

b) —— De acordo com a definição, dois componentes a executar **na mesma máquina**, em alguns casos poderão ser considerados um sistema distribuído, noutros não.

c) —— A distribuição de carga e a tolerância a falhas são motivações para conceber uma solução distribuída. Porém, são mutuamente incompatíveis, obrigando a optar por uma ou pela outra.

d) —— Um sistema de *middleware* pode ser usado no contexto de sistemas distribuídos para endereçar desafios de heterogeneidade de diversa ordem.

e) —— Normalmente desejável, a transparência em excesso pode revelar-se um defeito num sistema distribuído.

f) —— Caso exista a necessidade de alterar a implementação de um sistema distribuído isso pode significar, por exemplo, indicar que o sistema não é escalável.

g) —— As primitivas de comunicação *anycast* não sendo por definição fiáveis, significa que quando uma mensagem é enviada, a recepção da mesma pode ocorrer entre entre 0 e n dos potenciais n destinatários.

h) —— Um sistema distribuído pode ser geo-replicado e particionado ao mesmo tempo.

i) —— É correcto dizer-se que o particionamento tem como objetivo secundário melhorar a disponibilidade de um sistema.

j) —— Devido à maior proximidade com o utilizador/cliente e sendo a latência menor, a coordenação entre réplicas, num sistema distribuído geo-replicado é mais eficiente.

k) —— Um *proxy* colocado junto ao servidor permite mascarar ao cliente falhas de conectividade e que este trabalhe em modo desligado/*offline*.

l) —— Um sistema P2P cuja topologia forma uma árvore ternária, onde cada participante tem uma ligação para 3 outros parceiros, pode ser um sistema P2P não-estruturado.

m) —— Existem alguns tipos de pesquisas que mesmo num sistema P2P estruturado podem ter de ser realizadas por inundação ou outro método pouco eficiente.

n) —— Num *middleware* de comunicação em grupo que implemente a ordem FIFO, no processo emissor, a entrega da mensagem pode ocorrer imediatamente após ao seu envio, sem ter que esperar por outras mensagens.

o) —— Num *middleware* de comunicação em grupo que implemente a ordem causal, a entrega de uma mensagem pode ser feita assim que ocorrer a sua recepção, em qualquer processo.

Questão 3

No contexto da invocação remota, indique para cada um das seguintes afirmações se é `[V]`erdadeira ou `[F]`alsa. Em caso de dúvida, justifique a resposta na página em branco no final do teste. (nota: respostas incorretas não justificadas descontam.)

a) —— Em REST, por omissão, a semântica de invocação é "pelo menos uma vez"(*at least once*).

b) —— No .NET Remoting é possível aceder a atributos (variáveis) do servidor.

c) —— No Java RMI utiliza-se a própria linguagem Java como IDL.

d) —— Em SOAP, a passagem de dados envolve a codificação dos mesmos no formato JSON.

e) —— ProtoBuf é um sistema de invocação alternativo ao Java RMI.

f) —— No REST, para alterar um recurso através do método PUT, o conteúdo do recurso pode ser codificado no URL do pedido HTTP.

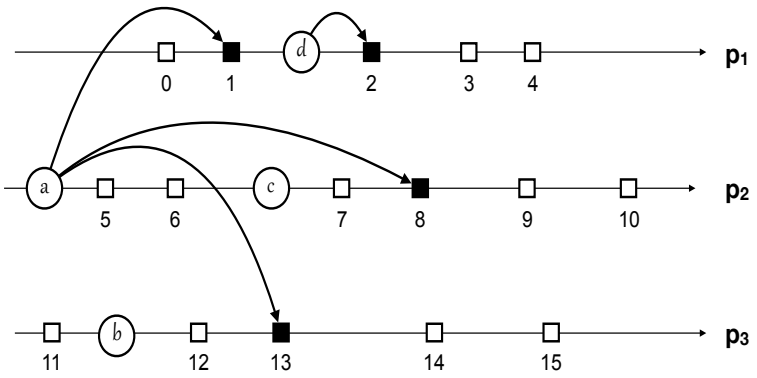
g) —— Nos web services SOAP, WSDL refere-se a um documento XML com uma parte abstrata e uma parte concreta. A parte concreta aponta para uma instância do serviço.

h) —— Nos web services SOAP, em nenhum caso, o WSDL obtido de um dado servidor pode ser usado para invocar outro servidor.

Número: Nome:

Questão 4

Considere o seguinte diagrama que ilustra um padrão de comunicação em grupo, envolvendo 3 processos: p_1, p_2 e p_3. No total, são enviadas 4 mensagens: a, b, c e d. Cada seta indica o momento em que ocorre a entrega dessa mensagem num dado processo. Para a mensagem a esse momento está fixado nos 3 processos; para a mensagem d, apenas para o processo p_1. Para as demais situações, o momento de entrega está em aberto (e poderá ser identificado por um número). A comunicação é fiável e todos os processos deverão receber cada uma das mensagens.



a) Considerando apenas o par de mensagens (a, c) e o momento de entrega da mensagem c em cada processo, indique um padrão de entrega que respeite a ordem FIFO.

$a \rightarrow [1, 8, 13]$ $c \rightarrow [$ $]$

b) Considerando todas as mensagens (a, c, d), indique um padrão de entrega que respeite a ordem total.

$a \rightarrow [1, 8, 13]$ $c \rightarrow [$ $]$ $d \rightarrow [2,$ $]$

c) Considerando todas as mensagens (a, b, c, d), indique um padrão de entrega que respeite a ordem total mas não a ordem causal.

$a \rightarrow [1, 8, 13]$ $b \rightarrow [$ $]$ $c \rightarrow [$ $]$ $d \rightarrow [2,$ $]$

Questão 5

Considere o Youtube que permite o carregamento de vídeos, produzidos pelos próprios utilizadores, e sua, posterior visualização por uma audiência global. Neste sistema, vídeos de autores populares podem acumular milhões de visualizações em pouco tempo, enquanto uma enorme massa de vídeos, não passa de umas poucas centenas ao final de vários anos. A visualização dos vídeos gera ingressos ao dono da plataforma através da publicidade que é embebida nos vídeos durante a sua reprodução. Estes lucros são partilhados com os autores que, para tal, são incentivados a produzir conteúdo original, com regularidade, sob pena de caírem no esquecimento e verem o seu rendimento desaparecer. É frequente os utilizadores "profissionais"da plataforma apelarem à audiência para que subscreva o seu canal, deixe comentários, e melhor ainda, peça para ser notificada quando existe novo conteúdo. Para além de conseguir criar um vídeo que se torna viral, esta é a forma sustentada de um canal conseguir criar uma audiência regular e começar a aparecer nas listas de vídeos recomendados que a plataforma apresenta aos utilizadores. Hoje em dia, os conteúdos do Youtube podem ser consumidos numa grande variedade de equipamentos, desde computadores pessoais, dispositivos móveis e televisões inteligentes, espalhados por todo o planeta.

a) Das enunciadas, indique uma característica que motive para neste sistema se utilizar uma arquitectura baseada num servidor *particionado*.

b) Das características enunciadas, indique pelo menos uma que motive para neste sistema se utilizar uma arquitectura baseada num servidor *replicado*.

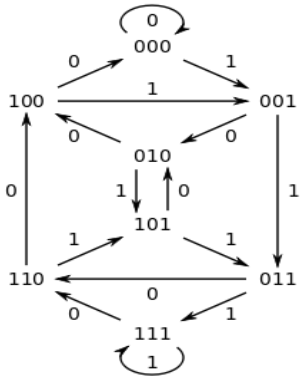
c) Das características enunciadas, indique uma que motive para neste sistema se utilizar uma arquitectura baseada num servidor *geo-replicado*.

d) Das características enunciadas, indique uma que motive para neste sistema se utilizar um sistema editor/assinante.

Questão 6

Num grafo *de Bruijn*, de d dimensões, existem 2^d vértices. Cada vértice tem um identificador único de d bits. Nestes grafos, um vértice com identificador i está ligado aos vértices: $2i \ mod \ 2^d$ e $2i \ mod \ 2^d \ + \ 1$. A figura, abaixo, mostra o grafo *de Bruijn* de dimensão 3.

Considere um sistema P2P em que os nós participantes estão organizados num grafo *de Bruijn*, com 32 dimensões. Por conseguinte, a sua capacidade máxima deste sistema será de 2^{32} *participantes*, que estarão ligados uns aos outros *peers* de acordo com a regra descrita anteriormente.



a) Indique, justificando, se este sistema P2P deveria ser classificado como estruturado ou não-estruturado.

b) Apresente de forma resumida um esboço do processo de encaminhamento neste sistema P2P. Qual será o comprimento do caminho mais longo entre quaisquer dois nós participantes?

Anexo A - Interface do Servidor REST

```
package sd.teste1.rest.api;

import ...
import sd.teste1.Contact;


public interface ContactListResource {




   void createContact(                        Contact c);




   Contact changeContact(                        String name,                        Contact c);




   Contact obtainContact(                        String name);




   void removeContact(                        String name);




   Contact[] listContacts(                         String prefix);
}
```

Anotações JAX-RS

```
@Path()
@GET
@PUT
@POST
@DELETE
@Consumes()
@Produces()
@PathParam()
@QueryParam()

MediaType.APPLICATION_JSON
MediaType.APPLICATION_OCTET_STREAM
```

Anexo B - Interface do Servidor SOAP

```
package sd.teste1.soap.api;

import ...

import sd.teste1.Contact;



public interface ContactListApi {

    static final String NAME = "ContactList";
    static final String NAMESPACE = "http://sd2018";
    static final String INTERFACE = "_____________________________________________";




    public void createContact(Contact c) throws InvalidArgumentException;




    public Contact changeContact(String name, Contact c) throws InvalidArgumentException, ContactDoesNotExist;




    public Contact obtainContact(String name) throws InvalidArgumentException, ContactDoesNotExist;




    public void removeContact(String name) throws InvalidArgumentException, ContactDoesNotExist;




    public Contact[] listContacts(String prefix) throws InvalidArgumentException;



    class InvalidArgumentException extends Exception {
        private static final long serialVersionUID = 1L;

        public InvalidArgumentException() {
            super("");
        }
        public InvalidArgumentException(String msg) {
             super(msg);
        }
    }



    class ContactDoesNotExist extends Exception {
        private static final long serialVersionUID = 1L;

        public ContactDoesNotExist() {
            super("");
        }
        public ContactDoesNotExist(String msg) {
            super(msg);
        }
     }
}
```

Anexo C - Cliente REST

```
package sd.teste1.rest;

import ...
import sd.teste1.Contact;

public class ContactListCreateClient {

    public static void main (String[] args) {

        ClientConfig config = new ClientConfig();
        Client client = ClientBuilder.newClient(config);

        URI baseURI = UriBuilder.fromUri("http://localhost:8080/").build();
        WebTarget target = client.target( baseURI );

        Response response = target.path("contacts/").request()
              .post( Entity.entity( new Contact("João Leitão", "FCT-UNL", "10765"),
                                                                  MediaType.APPLICATION_OCTET_STREAM));

        if( response.getStatus() == 204)
            System.out.println("OK: Contact created");
        else
            System.out.println("ERROR: Contact already exists");
    }
}
```

Execution Output...

```
Apr 26, 2018 3:05:27 PM org.glassfish.jersey.message.internal.WriterInterceptorExecutor$TerminalWriterInterceptor aroundWriteTo
SEVERE: MessageBodyWriter not found for media type=application/octet-stream, type=class sd.teste1.Contact, genericType=class sd.teste1.Contact.
Exception in thread "main" org.glassfish.jersey.message.internal.MessageBodyProviderNotFoundException: MessageBodyWriter
not found for media type=application/octet-stream, type=class sd.teste1.Contact, genericType=class sd.teste1.Contact.
        at ...
        at org.glassfish.jersey.message.internal.WriterInterceptorExecutor.proceed(WriterInterceptorExecutor.java:162)
        at org.glassfish.jersey.message.internal.MessageBodyFactory.writeTo(MessageBodyFactory.java:1130)
        at org.glassfish.jersey.client.ClientRequest.doWriteEntity(ClientRequest.java:517)
        at org.glassfish.jersey.client.ClientRequest.writeEntity(ClientRequest.java:499)
        at org.glassfish.jersey.client.internal.HttpUrlConnector._apply(HttpUrlConnector.java:393)
        at org.glassfish.jersey.client.internal.HttpUrlConnector.apply(HttpUrlConnector.java:285)
        at org.glassfish.jersey.client.ClientRuntime.invoke(ClientRuntime.java:252)
        ...
        at org.glassfish.jersey.process.internal.RequestScope.runInScope(RequestScope.java:444)
        at org.glassfish.jersey.client.JerseyInvocation.invoke(JerseyInvocation.java:681)
        at org.glassfish.jersey.client.JerseyInvocation$Builder.method(JerseyInvocation.java:437)
        at org.glassfish.jersey.client.JerseyInvocation$Builder.post(JerseyInvocation.java:343)
        at sd.teste1.rest.ContactListCreateClient.main(ContactListCreateClient.java:29)
```

Anexo D - Servidor

```
package sd.teste1.rest.impl;

import ...
import sd.teste1.Contact;
import sd.teste1.rest.api.ContactListResource;

public class ContactListImplementation implements ContactListResource {
   private Map<String, Contact> contacts = new ConcurrentHashMap<>();

   // Server methods omitted...
   public static void main( String[] args ) {
     String URI_BASE = "http://         (A)         :8080/";

     ResourceConfig config = new ResourceConfig();
     config.register( new ContactListImplementation() );
     JdkHttpServerFactory.createHttpServer( URI.create(URI_BASE), config);
     System.err.println("Server ready....");
  }
}
```

Anexo D - Cliente

```
package sd.teste1.rest;

import ...

public class ContactListDeleteClient {
   public static void main (String[] args) {
      ClientConfig config = new ClientConfig();
      Client client = ClientBuilder.newClient(config);

      URI baseURI = UriBuilder.fromUri("http://         (B)         :8080/").build();

      WebTarget target = client.target( baseURI );
      Response response = target.path("contacts/" + "João Leitão").request().delete();
      if( response.getStatus() == 204)
         System.out.println("OK: Contact deleted");
      else
         System.out.println("ERROR: Contact does not exist");
   }
}
```