PROGRAMAÇÃO DISTRIBUÍDA

Exame da Época Normal – Parte I

4 de fevereiro de 2022

Sem consulta / Duração: 40 minutos / 4 valores

---

1. *[2 valores]* O paradigma de programação baseado em objetos/serviços remotos pressupõe a existência de um tipo de objeto genericamente designado *proxy*, *stub* ou referência remota.

a. Explique o objetivo principal de este tipo de objeto;

R: Os objetivos principais de um stub (proxy/referência remota) são representar um objeto remoto localmente e abstrair a comunicação (sockets, mensagens trocadas, etc...) necessária à invocação de métodos nesse objeto remoto. Habitualmente o stub implementa a interface que define os métodos passíveis de invocação remota (métodos do serviço-alvo) mas implementa os seus métodos com recurso a comunicação com o servidor que disponibiliza o serviço (pedido único para cada método e posterior espera por uma resposta do servidor).

b. Indique, justificando, dois membros (atributos) que devem obrigatoriamente fazer parte de este tipo de objeto;

R: Dois membros que devem obrigatoriamente fazer parte de um stub são a referência do objeto remoto e o endereço de rede no qual o objeto está alojado. Isto porque sem uma referência do objeto, é impossível encontrar o serviço que deve responder aos pedidos do cliente num sistema em que existam vários serviços/servidores a correr num só endereço de rede e sem o endereço de rede, seria impossível localizar o objeto remoto.

c. Indique qual é a característica que este tipo de objeto tem em comum com os respetivos objetos/serviços remotos que se pretende efetivamente aceder.

R: A característica que o stub tem em comum com o serviço/objeto remoto que se pretende aceder é a assinatura dos seus métodos (stub deve ter os mesmos métodos do serviço – nome, parâmetros, tipo devolvido).

d. Explique por que razão, quando se invocam métodos em serviços remotos Java RMI, não é possível passar como argumento nem devolver como resultado objetos do tipo `Tarefa`.

```
public class Tarefa {
      private int id;
     private String description;

     . . .
}
```

R: Tarefa não é serializável (não implementa a interface Serializable – o que significa que também não é "Marshallable"). Objetos passados por argumento e devolvidos em métodos de um objeto remoto têm que ser instâncias de classes serializáveis para permitir que possam ser transformados em bytes e re-transformados em objetos durante a comunicação.

2. *[2 valores]* Considere uma API REST que inclui, entre outros, os seguintes pedidos:

	Método HTP	**URI / caminho**	**Parâmetros opcionais**
1	GET	/tarefas/urgentes	id_max id_min
2	GET	/tarefas/{id}	
3	PUT	/tarefas/secundarias/{id}	
4	DELETE	/tarefas/obtem/{id}	
5	POST	/tarefas/secundarias	

a. Indique um possível objetivo para cada pedido.

R:

1 Devolver as tarefas urgentes (se forem passados parâmetros na URI – que tenham id superior ou igual a id_min e inferior ou igual a id_max) registados no sistema.
2 Devolver os dados referentes à tarefa identificada pelo parâmetro id (da URI).
3 Substituir a tarefa secundária identificada pelo parâmetro id (da URI) com os dados passados no corpo do pedido.
4 Remover a tarefa identificada pelo id (da URI).
5 Adicionar uma tarefa secundária nova com os dados passados no corpo do pedido.

b. Identifique, justificando, eventuais URI malformadas.

1 Está correto de acordo com os princípios e normas HTTP reenforçados no REST (devolve uma coleção de tarefas urgentes).
2 Está correto de acordo com os princípios e normas HTTP reenforçados no REST (devolve uma tarefa identificada pelo id – singleton).
3 Está correto de acordo com os princípios e normas HTTP reenforçados no REST (substitui uma tarefa secundária identificada pelo id).

4 Está incorreto de acordo com os princípios e normas HTTP reenforçados no REST porque na uri verificamos o verbo "obtem" que não encaixa de todo num pedido com o método DELETE (seria mais esperado num pedido com o método GET ou no pior dos casos HEAD).

5 Está correto de acordo com os princípios e normas HTTP reenforçados no REST (publica uma nova tarefa secundária).

c. Indique, justificando, quais são os pedidos que requerem a existência de dados no corpo das mensagens HTTP correspondentes. Assumindo que as tarefas são caracterizadas por um identificador numérico e uma descrição (texto), dê um exemplo concreto de conteúdo (corpo) para cada um dos pedidos identificados (em JSON ou XML).

Os pedidos que requerem a existência de dados no corpo da mensagem são principalmente os pedidos que recorrem aos métodos/verbos PUT, o POST e o PATCH.

PUT e POST:

{

"id": 1,

"descrição": "Descrição exemplo"

}

PATCH:

{

"op": "replace",

"path": "/descrição",

"value": "Descrição exemplo"

}

d. Complete o pedido HTTP seguinte assumindo que se pretende obter a lista de tarefas urgentes com identificadores compreendidos entre 1000 e 2000 e que é fornecido o *token* de autorização "abcd":

```
GET /tarefas/urgentes?id_min=1000&id_max=2000 HTTP/1.1
Host: pd.isec.pt
User-Agent: curl/7.55.1
Accept: */*
Authorization: Bearer abcd
```