Instituto Superior de Engenharia de Lisboa
Licenciatura em Engenharia Informática e de Computadores
Programação na Internet
Inverno de 2017/2018
2ª Época (*2 de Fevereiro de 2018)* – Duração: 2h30

Grupo 1 [6 valores]

1. [3] Considere a funcionalidade de criação de um recurso (exº uma nova lista de filmes) numa aplicação web.

a. [1,5] Para a definição do endpoint que cria o recurso, podem ser usadas duas abordagens: uma baseada no método POST e outra no método PUT. Qual o URI usado em cada abordagem e, caso alguma delas imponha limitações, identifique-as.

b. [1,5] Na implementação desse endpoint qual o código HTTP que escolheria retornar no caso em que a operação de criação é concluída com sucesso? Justifique.

2. [3] O seguinte código em JavaScript está incluído num documento HTML. Os *endpoints* HTTP a que este código acede estão implementados com tecnologia Node.js e quando acedidos retornam sempre uma resposta. Tendo em consideração que os ambientes de execução do *browser* e Node são JavaScript e, como tal, apenas têm um fio de execução, **justifique** a sequência de mensagens apresentadas na consola do *browser*.

```
1  <script>
2     function makeRequest(method, uri, data, cb, msg) {
3         console.log(`${msg} -> Send request`)
4         let req = new XMLHttpRequest()
5         req.open(method, uri)
6         req.onreadystatechange = () => {
7             if(req.readyState == XMLHttpRequest.DONE) {
8                 cb(req.status, req.responseText, msg)
9             }
10        }
11        req.send(data)
12    }
13
14    function processResponse(status, response, msg) {
15        console.log(`${msg} -> Reply received. status: ${status} - ${response}`)
16    }
17    makeRequest('GET', '/foo', null, processResponse, "request1")
18    makeRequest('PUT', '/foo', null, processResponse, "request2")
19 </script>
```

Grupo 2 [6 valores]

3. [2,5] Considere o seguinte código fonte de um documento HTML.

```
<!DOCTYPE html>
<html>
<head><title>Page</title></head>
<body>
   <form method="POST" action="/name">
       <label for="name">Name</label><input type="text" name="" id="name" />
       <input type="submit" value="Send" id="submit" disabled>
   </form>
</body>
<script>
   function nameEmpty() {
      if(document.getElementById("submit").disabled =
                                          !document.getElementById("name").value)
          nameEmpty()
   }
   nameEmpty()
</script>
</html>
```

Pretende-se que o botão de submit do formulário apenas esteja ativo quando a caixa de texto `name` tiver algum conteúdo. Comente a implementação e apresente uma alternativa se considerar que esta pode ser melhorada.

4. [3,5] Implemente em Node.js o módulo `obj-mw` que retorna uma função geradora de *middlewares*, que recebe como argumento um objeto. Esse objeto deve ter propriedades cujo nome corresponde aos métodos HTTP. Os valores são um objetos em que os nomes das propriedades correspondem às *paths* suportadas. Por sua vez, o valor de cada um dessas propriedades é uma função com a assinatura de um *middleware* express, que atenderá o pedido. A listagem seguinte apresenta um exemplo de utilização do módulo:

```
const express = require('express');
const objmw = require("./obj-mw")
const app = express();
let obj = {
   get: {
       p1: (req, rsp) => rsp.end("get_p1"),
       p2: (req, rsp) => rsp.end("get_p2")
   },
   put: {
       p1: (req, rsp) => rsp.end("put_p1"),
       p2: (req, rsp) => rsp.end("put_p2")
   },
   post:{
       p1: (req, rsp) => rsp.end("post_p1")
   }
}
app.use('/foo', objmw(obj))
```

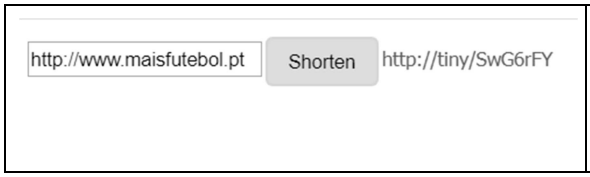
Para este exemplo a aplicação suportaria os seguintes *endpoints*:

- `GET /foo/p1`
- `GET /foo/p2`
- `PUT /foo/p1`
- `PUT /foo/p2`
- `POST /foo/p1`

NOTA: Na implementação de `obj-mw`, não pode ser usado qualquer módulo externo, nomeadamente o Express.

Grupo 3 [8 valores]

A aplicação Web (tiny.pt) gere versões curtas de URLs. O utilizador insere um URL e obtém um endereço correspondente no domínio tiny.pt. Seguindo um endereço obtido de `tiny.pt`, será reencaminhado para o endereço original.
Se inserir um endereço inválido (e.g. `http://www.ljkdsf.pt/`) é apresentad a mensagem "Invalid URL!" conforme o exemplo.
Se inserir um endereço (e.g. `http://www.maisfutebol.pt`) que é redireccionado para um outro endereço, a aplicação `tiny.pt` deve armazenar o endereço final para onde foi reencaminhado (e.g. http://www.maisfutebol**.iol.**pt).
A aplicação deve seguir **todos** os redireccionamentos subsequentemente **até que obtenha o endereço final válido a armazenar**.



A aplicação tiny.pt aceita pedidos nas seguintes paths:

- **`/`** – retorna a página principal correspondente à vista HTML seguinte
- **`/shorten`** – recebe um endereço no corpo do pedido HTTP via AJAX e retorna um resultado de acordo com a descrição acima. **Armazena em memória a relação entre os dois endereço**s: o curto e o original, resolvido se tiver reencaminhamentos.
- **`/{tinyURL}`** – recebe um pedido para um endereço curto e redirecciona para o endereço original.

```
<html>
 <head>
    <script type='text/javascript'
            src='/javascripts/tiny.js'>
  </script>
 </head>
 <body>
  <input id="srcUrl" type="text">
  <button id="btShorten" class="btn">
    Shorten
  </button>
  <span id="tinyUrl"></span>
 </body>
</html>
```

1. [2] Complete código do ficheiro tiny.js nos pontos 1 a 7.
2. [4] Sendo app a instância da aplicação express que implementa a aplicação tiny.pt, **adicione** a app e **implemente** o *middleware* corresponde ao *endpoint* `/shorten`. (admita a existência de uma função auxiliar `shortString` que produz uma versão curta da `String` recebida por parâmetro).
3. [2] A**dicione** a app e **implemente** o *middleware* corresponde ao *endpoint* `/{tinyURL}`.

```
window.onload = function() {
  const srcUrl = document.getElementById('srcUrl')
  const tinyUrl = document.getElementById('tinyUrl')
  document
    .getElementById('btShorten')
    .onclick = ________ // *1*
       httpRequest(___/*2*/, ___/*3*/, ___/*4*/, tinyResult)

  function tinyResult(err, data){
     /*5*/
  }
  function httpRequest(method, path, data, cb) {
    const xhr = new XMLHttpRequest()
    xhr.open(method, path, true)
    ________ //*6*/
    xhr.onreadystatechange = function() {
      if(xhr.readyState == XMLHttpRequest.DONE) {
        if(xhr.status == 200)
            cb(null, xhr.responseText)
        else
            ________ //*7*/
      }
    }
    xhr.send(data)
  }
}
```

Os Docentes,
Luís Falcão, Miguel Carvalho e Paulo Pereira