Instituto Superior de Engenharia de Lisboa

Licenciatura em Engenharia Informática e de Computadores

Programação na Internet - Teste Global de Época Normal, Inverno de 2018/2019

---

1. [1] Um programador no desenvolvimento de um módulo NodeJs optou por implementar esse módulo com uma API síncrona. Justifique se existe algum cenário em que faça sentido esta opção.

2. [1] Uma aplicação Express configurada com o *middleware* dado por `passport.initialize()`, disponibiliza os métodos `login()` e `logout()` no objeto `Request`. Explique detalhadamente todos os passos executados pela chamada do método `logout()`.

3. [1] Um programador de uma aplicação Web (*Thoth*) para gestão de alunos (*student*) numa turma (*classroom*) definiu os seguintes caminhos para as operações de adição e remoção de aluno numa turma:
   - Adicionar - `/thoth/classroom/{id}/student/{student id}/add`
   - Remover - `/thoth/classroom/{id}/student/{student id}/remove`

   Justifique se esta opção é correcta, ou não?

4. [1] a) Justifique quantos pedidos HTTP resultam do carregamento do seguinte documento HTML no browser:

```
<html>
    <head>
        <link rel="stylesheet" href="bootstrap.css">
        <script src="handlebars.js"></script>
        <script src="bookSearch.js"></script>
    </head>
    <body>
                        <a href="https://www.gutenberg.org/" >
                            Project Gutenberg
                            <img  src="Gutenberg-logo.png">
                        </a>
```

4. [1] b) Justifique se a utilização do `webpack` pode ajudar a reduzir o número de pedidos HTTP de 4.a? Em caso afirmativo quais as modificações necessárias no *front-end* mantendo o mesmo comportamento de 4.a)

5. [2,5] Implemente a função `requestsToList(urls, listID)` que realiza um pedido HTTP para cada um dos caminhos presentes em `urls` e adiciona o corpo da resposta como `li` ao elemento HTML com id `listId`.
   Os elementos `li` devem ser adicionados:
   - pela mesma ordem do url correspondente em `urls`
   - à medida que são recebidas as respostas para cada url.

   Por exemplo, a resposta ao pedido do 3º url não pode ser inserida na lista antes da inserção da resposta do pedido do 2º url, mesmo que este último chegue mais tarde.
   A utilização de uma abordagem como a do `Promise.all` está errada. Pode usar o `fetch` ou `request-promise`.

6. [2,5] Implemente a função `profile(fn)` que retorna uma função com o mesmo comportamento de `fn`, mas que regista os tempos de execução de cada chamada a `fn`. Assuma que `fn` é assíncrona e retorna uma `Promise`.
   Além disso, a função retornada por `profile` tem ainda um método `avgDur` que retorna o tempo médio de execução de `fn` em milissegundos. Assuma a existência de um objecto global `performance` com um método `now()` que retorna o instante de tempo actual em milissegundos. Exemplo:

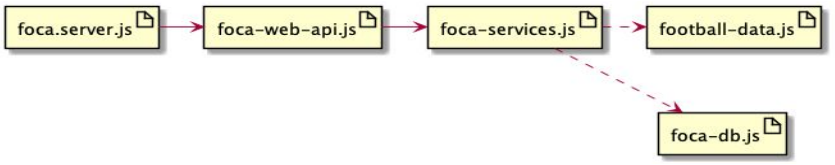
```
function foo(timeout) {
  return new Promise((resolve, reject)=> {
    setTimeout(
      () => resolve(`RESOLVED after ${timeout} ms`),
      timeout)
  })
}
```

```
const bar = profile(foo)
bar(2000)
    .then(console.log)
bar(3000)
    .then(console.log)
    .then(() => console.log(bar.avgDur()))
```

   Exemplo de uma utilização do código anterior:

```
RESOLVED after 2000 ms
RESOLVED after 3000 ms
2502.2250160006806
```

5. [2,5] A aplicação FOCA, desenvolvida na 1ª parte do trabalho prático, é constituída pelos módulos apresentados na figura seguinte, bem como as dependências entre cada um destes.



Para cada módulo, indique 2 funcionalidades que devam ser implementadas e 2 que não devam, porque são responsabilidade de um dos outros módulos.

6. [6] Pretende-se acrescentar à aplicação FOCA a possibilidade de ter grupos de grupos de equipas favoritas, designados de Grupos Compostos (GC). Um GC só pode ser constituído por outros grupos, que podem por sua vez, ser também grupos compostos. Deste modo, na página que apresenta a lista de jogos das equipas de um GC, constam os jogos de todas as equipas que pertencem a cada um dos grupos que o compõem, recursivamente. Note-se que um grupo, apenas pode constar uma única vez como subgrupo de um grupo composto.

   - [2,5] Defina os *endnpoints* HTTP para suportar as operações CRUD para ter a funcionalidade de GC na aplicação. Nessa definição inclua os *status codes* e o conteúdo da resposta, no caso da pedido ter sucesso e, caso se aplique, no caso de resposta com insucesso ao pedido.

   - [1,5] Para suportar esta funcionalidade, que módulos teria que alterar dis apresentados na questão anterior, e que alterações seriam essas, nomeadamente os métodos a adicionar a cada um dos módulos.

     NOTA: **Descreva apenas as alterações, sem as implementar**!

   - [2,5] A componente de cliente desta aplicação é uma Single Page Application (SPA). A implementação do módulo de *routing* dessa página está na listagem seguinte (`router.js`). Crie um módulo de cliente (`compound-groups.js`) que usa este módulo de routing para registar as suas rotas, de modo a incluir a funcionalidade de grupos compostos nessa SPA.

     NOTA: Assuma que o módulo `compound-groups.js` já tem: 1) numa variável global o identificador `mainContentId`; 2) implementadas todas as *views* e *scripts* de cliente necessários.

```
module.exports = function (mainContentId) {
 let mainContent = document.querySelector(mainContentId)
 window.addEventListener('hashchange', routeChanged);
 const routers = []
 routeChanged();

 return { add: routers.push }

 async function routeChanged() {
   let [routeName, ...params] = window.location.hash.split('/')
   routeName = routeName.substring(1)
   let route = routers.find(r => r[routeName])
   if (routeName) {
     mainContent.innerHTML = await route.view.apply(null, params)
     route.script.apply(null, params)
   } else {
     window.location.hash = '#welcome'
   }
 }
}
```

7. [1,5] Porque motivo as Single Page Applications apenas mudam o *hash* do URL?

Duração: 2 horas
ISEL, 17 de Janeiro de 2019