Instituto Superior de Engenharia de Lisboa
Licenciatura em Engenharia Informática e de Computadores
Programação na Internet
Teste Global de Época Recurso, Verão de 2018/2019

GRUPO 1

1. [3] Considere o código seguinte de um *middleware* Express, cujo objetivo é realizar a autorização, deixando apenas prosseguir pedidos de utilizadores autenticados.

```
module.exports = function(redirectUrl) {
 return function(req, rsp, next) {
   if(req.isAuthenticated()) {
     return next();
   }
   rsp.set({'Location': redirectUrl}).end()
 }
}
```

   a. [1] Indique os pre-requisitos deste *middleware*, nomeadamente que middleware(s) têm que ser instalados antes deste, justificando.
   b. [2] A implementação deste módulo está incorreta. Indique os problemas estes podem provocar nas aplicações cliente e corrija a implementação de modo a eliminar esses problemas.

2. [2,5] Para a aplicação B4 desenvolvida durante as aulas, definiu-se que não são suportados grupos (*bundles*) com o mesmo nome. Tendo em conta este requisito a as características inerentes aos diferentes tipos de pedido do protocolo HTTP, identifique quais as possibilidades de endpoints para criação de um grupo, em termos do tipo de pedido HTTP e correspondente URI e por qual optaria, justificando.

3. [2,5] A função seguinte usa a API `fetch()` para obter os títulos de todos os livros pertencentes a todos os grupos de livros (bundles) existentes na base de dados ElasticSearch da aplicação B4, desenvolvida durante as aulas.

```
async function getAllBundlesBookTitles() {
 const bookUrlBase = 'http://localhost:9200/books/book/'
 return fetch('http://localhost:9200/b4/bundle/_search')
                .then(response =>  response.json())
                .then(bundles =>
                  bundles.hits.hits
                    .flatMap(bundle => bundle._source.books)
                    .map(b => fetch(bookUrlBase + b.id)))
                .then(requests => _____________________)        // 1
                .then(responses => responses.map(resp => resp.json()))
                .then(responseBodies => _____________________)  // 2
                .then(books => books.map(b => b._source.title))
}
```

   a. [1] Complete os espaços em branco nas linhas que marcadas com os comentários //1 e //2.
   b. [1,5] Reescreva a função de modo a ter o mesmo comportamento, mas usando sempre que possível a keyword `await`.

4. [3] Implemente o módulo **express-server**, que suporta a utilização exemplificada no código seguinte. Este módulo usa o módulo **express** para suportar a sua funcionalidade e, quando for chamado o método **start**, inicia uma nova instância de aplicação Express, no *host* e porto recebidos como parâmetro.

```
const HOST = "localhost"
const PORT = 8080

require('./express-server')()
 .use(mw1)
 .get('/somePath', mw2)
 .get(mw3)
 .put('/somePath', mw4)
 .delete('/somePath', mw5)
 .post('/somePath', mw6)
 .start(HOST, PORT);
```

NOTAS: m1, m2, m3, m4, m5 e m6 são *middlewares* express definidos anteriormente. A listagem anterior é meramente exemplificativa. O módulo deve suportar os mesmos métodos e parâmetros que os métodos correspondentes numa aplicação Express para cada um dos tipos de pedidos suportados: get, put, post e delete.

GRUPO 2

5. [9] Pretende-se adicionar à YAMA, desenvolvida no trabalho prático, a possibilidade de adicionar várias vezes a mesma música a uma *playlist*. No entanto, esta é uma característica da playlist, ou seja, há *playlists* que suportam músicas repetidas e outras que não, dependendo da opção com que cada uma foi criada, ou posteriormente editada.

   a. [2] Especifique o *endpoint* que está disponível na componente servidora da aplicação, para suportar esta funcionalidade. A resposta a este pedido deve ser sempre no formato Json, quer seja concluído com sucesso ou não.

   b. [1,5] Seguindo a arquitetura de módulos da componente servidora definida para o trabalho prático, defina o método do módulo yama-web-api, que implementa o endpoint definido na alínea anterior. Este módulo deve usar um método do módulo **yama-service**, presente na variável **yamaService** global ao módulo, que será implementado na alínea seguinte. Deste modo, a sua utilização deve ser coerente com a implementação a realizar.

   c. [3,5] Implemente o método do módulo **yama-service** usado na alínea anterior, com a sintaxe e semântica que resultam da sua utilização. Na implementação desta método, assuma que tem disponível uma instância do módulo **yama-db** na variável **yamaDb**, global ao módulo. Assuma que o módulo **yama-db** tem as funções necessárias à implementação desta funcionalidade. Descreva o que fazem todas funções de **yama-db** utilizadas, sem as implementar.

      NOTA: Na implementação deste método, não esquecer que nem todas as playlists suportam a adição da mesma música mais que uma vez.

   d. [2] Assumindo esta nova funcionalidade que possibilita a existência de várias músicas numa *playlist* se esta assim o suportar, descreva as alterações teriam que ser realizadas ao modelo de dados que representa uma *playlist*, e que alterações teriam que ser realizadas ao método de **yama-service** que edita os dados de uma playlist.

   NOTA: Nesta alínea, pede-se apenas que descreva as alterações, não que as implemente.

Duração: 2 horas
ISEL, 2 de Julho de 2019