Instituto Superior de Engenharia de Lisboa
Licenciatura em Engenharia Informática e de Computadores
Desenvolvimento de Aplicações Web
Teste Final de Época Especial, Semestre de Inverno, 22/23
Duração: 2 horas

---

1. (6) Para cada uma das seguintes questões, indique qual a resposta correta. Cada resposta incorreta subtrai 1/3 pontos à classificação total do conjunto de questões deste grupo.

   1.1. No protocolo HTTP, um pedido de método GET para `https://example.com/games/create`:
   - i. Solicita a criação do recurso com URI `https://example.com/games/create`.
   - ii. Solicita a criação de um recurso com URI `https://example.com/games/{id}`, onde o valor de `id` é determinado pelo servidor.
   - iii. Solicita uma representação do recurso com URI `https://example.com/games/create`.
   - iv. Deve resultar sempre numa resposta com *status code* 405 (*method not allowed*).

   1.2. Uma mensagem de resposta HTTP com *status code* igual a 401 e `Content-Type` igual a `application/problem+json` deve ser interpretada por um intermediário como sendo:
   - i. Uma resposta de sucesso.
   - ii. Uma resposta de não sucesso.
   - iii. Uma resposta de sucesso ou de não sucesso, dependendo do valor do campo `type` presente na representação.
   - iv. Nenhuma das anteriores.

   1.3. O campo `rel` presente num *header* `Link` representa:
   - i. O identificador do recurso alvo do *link*.
   - ii. O *media-type* potencialmente recebido na resposta a um pedido ao recurso alvo do *link*.
   - iii. Um valor booleano que indica se o URI para o destino é absoluto ou relativo.
   - iv. Nenhuma das anteriores.

   1.4. Na biblioteca *Spring MVC* e na configuração por omissão:
   - i. É usada apenas uma instância por cada tipo de *controller*.
   - ii. É criada sempre uma nova instância dum *controller* por cada pedido.
   - iii. Apenas são criadas novas instâncias de *controller* quando todas as instâncias existentes estiverem a ser usadas.
   - iv. Não são criadas instâncias de *controller* porque todos os métodos têm de pertencer ao *companion object*.

   1.5. Considere o seguinte componente para a biblioteca React

```
function Counter() {
  const [value, setValue] = useState(0)
  useEffect(() => {
    const tid = setInterval(() => setValue((x) => x + 1), 1000)
    return () => {clearInterval(tid)}
  }, [setValue])
  return (
    <div>{value}</div>
  );
}
```

   A colocação deste componente resulta:
   - i. Na apresentação constante do valor 0.
   - ii. Na apresentação do valor 0, seguida do valor 1 após 1000 milissegundos.
   - iii. Na apresentação de um valor numérico, incrementado a cada 1000 milissegundos.
   - iv. Nenhuma das anteriores.

1.6. Quando uma *single page application* suporta *deep linking* e o utilizador introduz diretamente o URL `https://example.com/games?id=123` (e.g. ativando um *bookmark*), o *browser* faz sempre um pedido HTTP

   i. de método GET, usando o URL `https://example.com/index.html`.
   ii. de método GET, usando o URL `https://example.com/`.
   iii. de método GET, usando o URL `https://example.com/games`.
   iv. de método GET, usando o URL `https://example.com/games?id=123`.

2. (2) No desenho de APIs HTTP, quais as vantagens da utilização de métodos idempotentes?

3. (2) Tendo em conta que os *browsers* modernos já suportam o sistema de módulos ECMAScript Modules (ESM), qual a relevância de se ainda usar uma ferramenta como o webpack?

4. (5) Realize um ou mais componentes para a plataforma Spring MVC, de forma a que um recurso seja exposto no caminho `/errors`. Um pedido de método `GET` para esse recurso deve retornar uma mensagem com uma representação contendo um objeto JSON. Esse objeto deve representar os pedidos processados nos últimos 10 minutos e que resultaram numa resposta com *status code* igual a 500 A representação de cada pedido deve incluir: o momento em que o pedido foi recebido, o método do pedido, o URI do pedido, e o nome do controlador e do método responsável pelo processamento desse pedido (caso o processamento tenha sido realizado por um controlador).

5. (5) Realize um componente para a biblioteca React que receba as propriedade `f` do tipo `()=>Promise<string>` e que apresenta o *fulfillment value* ou a *rejection reason* da *promise* resultante da avaliação de `f`. O componente deve também apresentar um botão que promove a reavaliação da função `f` sempre que premido. Este botão deve estar desabilitado enquanto a promessa retornada pela última avaliação não se tiver completado. O componente deve ser sensível a mudanças na propriedade `f`.

Filipe Freitas,
Pedro Félix,
5 de setembro de 2023