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

---

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

   1.1. A realização de um pedido de método GET para `https://example.com/games/create`, deve ser interpretado por um intermediário como sendo:

   i. Um pedido não *safe* e não idempotente.
   ii. Um pedido *safe* e não idempotente.
   iii. Um pedido não *safe* e idempotente.
   iv. Um pedido *safe* e idempotente.

   1.2. Uma mensagem de resposta HTTP com *status code* igual a 200 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 representacão.
   iv. Nenhuma das anteriores.

   1.3. O campo `rel` presente num *header* `Link` representa:

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

   1.4. No contexto da utilização da biblioteca Spring MVC, a execução da função `doFilter` pertencente à interface `HttpFilter`:

   i. Ocorre sempre no contexto da mesma *thread*.
   ii. Ocorre sempre no contexto da *thread* associada à instância sobre a qual é chamado o método.
   iii. Ocorre sempre no contexto da *thread* associada ao pedido HTTP que resultou nesta chamada.
   iv. Ocorre sempre no contexto da *thread* associada ao *handler* que vai processar o pedido HTTP.

   1.5. Considere o seguinte componente para a biblioteca React

```
function Counter() {
  const [value, setValue] = useState(0)
  useEffect(() => {
    const tid = setInterval(() => setValue(value + 1), 1000)
    return () => {clearInterval(tid)}
  }, [])
  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 milisegundos.
   iii. Na apresentação de um valor numérico, incrementado a cada 1000 milisegundos.
   iv. Nenhuma das anteriores.

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

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

2. (2) No contexto da utilização de *hypermedia* no âmbito de APIs HTTP, em que situações deve ser usada o *link relation* `self`?

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

4. (4) Realize um ou mais componentes para uso com a biblioteca Spring MVC de forma a que, para cada pedido HTTP, seja emitida uma mensagem de *log* com: método HTTP; URI do recurso acedido; *status code* da resposta; tempo de processamento; identificador do *handler* que processou o pedido, caso o *handler* seja do tipo `HandlerMethod`. Valorizam-se soluções onde o cálculo do tempo de processamento tem em conta mais etapas desse processamento. Use o método `getShortLogMessage` para obter o identificador dum `HandlerMethod`.

5. (4) Realize um componente para a biblioteca React que recebe a propriedade `f` do tipo `()=>Promise<string>` e que apresenta o *fullfilment value* ou a *rejection reason* da *promise* resultante da avaliação de `f`. Enquanto esta *promise* estiver pendente, deve ser apresentado um contador incrementado a cada 100 milisegundos. O componente deve ser sensível a mudanças na propriedade `f`.

6. (2) Realize a função

```
useInput(initial: string): [
  currentValue: string,
  changeHandler: React.ChangeEvent<HTMLInputElement> => void
]
```

   para ser usada como *hook* em componentes para a biblioteca React, tal como ilustrado no seguinte exemplo

```
function Example() {
  const [value, handler] = useInput("")
  return (
    <input type="text" value={value} onChange={handler} />
  );
}
```

Filipe Freitas,
Pedro Félix,
4 de fevereiro de 2023