Ano letivo: 2022/2023

Curso: Lic. Engenharia De Redes E Sistemas De Computadores

Unidade Curricular	Programação Web

Lic.	Ano do curso	2º ano	2º semestre	ECTS	

NOME do ALUNO:

Prova Escrita

Versão: B

Duração: 100 minutos

Leia atentamente toda a prova antes de iniciar.

A prova é individual, não sendo permitido consultar os seus colegas. No entanto, pode consultar os apontamentos das aulas e a Internet.

O resultado final deve ser enviado para o moodle incluindo o Word da prova e PDF da prova (gravar como PDF) e os ficheiros HTML e JS desenvolvidos. Deve ser anexado o link para Github no tópico Avaliação.

No documento de resposta deve ser incluída a versão da prova.

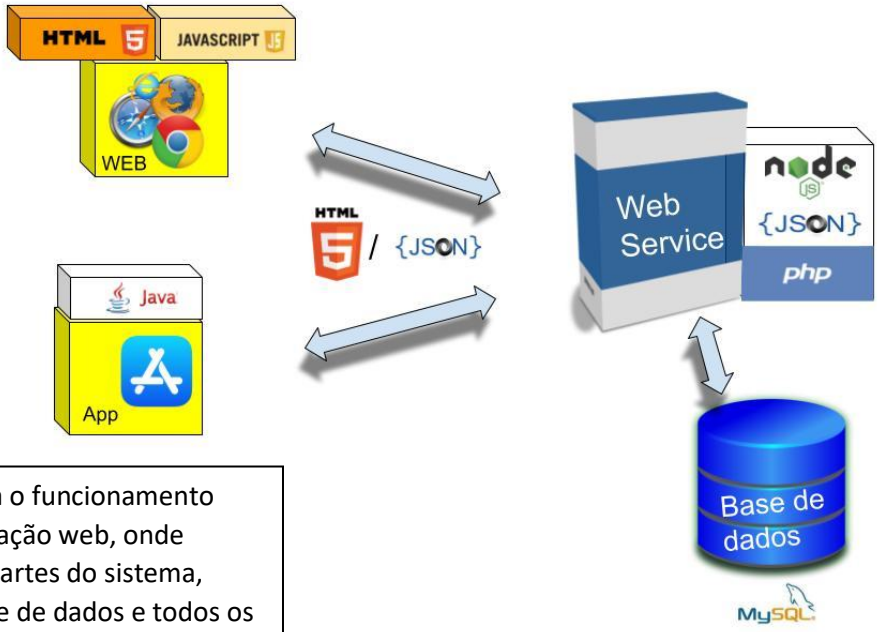
Durante a resolução deve ir gravando o trabalho para salvaguardar as alterações.

Parte I (25 valores)

1. À luz do que aprendeu na UC, comente a seguinte imagem.

Cofinanciado por:

POCH PORTUGAL 2020 UNIÃO EUROPEIA Fundo Social Europeu



A imagem representa o funcionamento básico de uma navegação web, onde vemos as principais partes do sistema, desde o cliente à base de dados e todos os seus atores.

Figura 1 - Estrutura do documento

2. Crie um protocolo para os alunos do IPVC para almoçar na cantina. Para que servem os protocolo e dê um exemplo

   -Em termos gerais, protocolos são conjuntos de regras e convenções que governam a comunicação entre dispositivos numa rede. Os protocolos estabelecem diretrizes para a formatação dos dados, controlo de erro, autenticação, gestão das ligações e outras funcionalidades necessárias para uma comunicação eficaz. Garantem que todos os dispositivos envolvidos na comunicação sigam as mesmas regras, o que permite a troca de informações de forma confiável e eficiente.

   Exemplo de protocolo para os alunos do IPVC almoçarem na cantina escolar:

   1 – Fazer escolha da senha para o respetivo dia e respetiva cantina.

   2 - Efetuar o pagamento da(s) senha(s) escolhida(s).

   3 – Dirigir-se e aguardar a sua vez nas filas de espera respetivas.

4 – Apresentar a sua identificação académica (cartão do aluno) ao ser atendido.

5 – Horário de atendimento da cantina está delimitado entre as 12:00 e as 14:00.

6 – Caso exista, efetuar a higienização das mãos e respeitar o distanciamento social exigido.

Parte II (25 valores)

1. Considera os seguintes exemplos de objetos DOM.
   - document.getElementById(id)
   - document.getElementsByTagName(tagName)
   - document.getElementsByClassName(className)

   Porque no primeiro caso temos getElement e nos dois seguintes getElements? Dê um exemplo de utilização para cada exemplo

O motivo de os dois últimos serem chamados como “getElement**s**” e apenas o primeiro ser chamado como “getElement” é devido ao facto das diferentes naturezas do que esta função vai buscar, ou seja, enquanto o primeiro(getElement) vai buscar o elemento com um id especifico, que por sua vez existe um e um só em todo o código, os seguintes (getElements) vão buscar, respetivamente, um conjunto de elementos, o primeiro(getElementsByTagName) vai buscar todos os elementos com a tag pedida, podendo ser <p> , <h1>, etc… e o segundo(getElementsByClassName) vai buscar todos os elementos pedidos que tenha associada a class pedida, ex: <class=”bg-color”>, <class=”titulos”>, etc…

No código HTML:

```
<div id="primeiro">
 <h1 class="titulos">
 <p>Hello, world!</p>
  <ul>
    <li id="segundo">Item 1</li>
    <li>Item 2</li>
    <li>Item 3</li>
  </ul>
</div>
```

No código JavaScript:

```
const myIDS = document.getElementById('segundo');

const myCLASSES = document.getElementsByClassName('titulos');

const myTAGS = document.getElementsByTagName('p');
```

2. Cria uma estrutura em JSON para registar Atores e Filmes. Faz um XML para a mesma estrutura. Comenta os resultados

No código JSON:

```
{
 "atores": [
  {
   "id": 1,
   "nome": "Leonardo DiCaprio",
   "filmes": [1, 2]
  },
  {
   "id": 2,
   "nome": "Kate Winslet",
   "filmes": [1]
  }
 ],
 "filmes": [
  {
   "id": 1,
   "titulo": "Titanic",
```

```
      "dataDeLançamento": 1997
    },
    {
      "id": 2,
      "titulo": "Inception",
      " dataDeLançamento ": 2010
    }
  ]
}
```

No código XML:

```
<database>
 <atores>
  <ator>
   <id>1</id>
   <nome>Leonardo DiCaprio</nome>
   <filmes>
    <filme>1</filme>
    <filme>2</filme>
   </filmes>
  </ator>
  <ator>
   <id>2</id>
   <nome>Kate Winslet</nome>
   <filmes>
    <filme>1</filme>
```

```
    </filmes>
  </ator>
 </atores>
 <filmes>
  <filme>
   <id>1</id>
   <titulo>Titanic</titulo>
   <dataDeLançamento>1997</dataDeLançamento>
  </filme>
  <filme>
   <id>2</id>
   <titulo>Inception</titulo>
   <dataDeLançamento>2010</dataDeLançamento>
  </filme>
 </filmes>
</database>
```

Os dois blocos de código tem o mesmo output, no entanto percebemos facilmente que o código JSON é de muito mais fácil leitura relativamente ao código XML.

Parte III (20 valores)

1. Qual a diferença entre <p> e <pre>

   A diferença que existe entre estas duas tags, é visível na apresentação da página, onde as tags <p> dão-nos uma quebra de linha com espaçamento, e as tags <pre> dão-nos a tanto quebras de linhas como espaçamentos diferenciando-se do <p> por manter a formatação do texto tal e qual como é escrito no ficheiro HTML.

Cofinanciado por:

2. Para que server

   <meta charset="utf-8">

   Serve para especificar a codificação de carateres do documento. Esta codificação determina como os caracteres do texto são representados. Garantimos assim que o navegador interprete corretamente o conjunto completo de carateres. UTF-8 (Unicode Transformation Format - 8 bits).

Parte IV (30 valores)

1. Prepara uma página com uma tabela 2x2 com estilos CSS que permitam apresentar 4 marcas de produtos de rede. Usa cores de fundo e cores de escrita e o logotipo de cada marca.

   Resolução na pasta “PART4” .

Parte V (50 valores)

1. Usando o Bootstrap, construa uma página com cards que mostre 6 monumentos e atrações turísticas do seu local de residência.
2. Cada card tem de ter um botão “ver mais” para ver mais detalhes.

   Resolução na pasta “PART5” .

Parte VI (50 valores)

Considere as imagens seguintes.

routes > JS products.js > ...

```
1   const productsRouter = require('express').Router();
2   const controller = require('../controllers/products');
3   const authMiddleware = require('../middlewares/auth/auth');
4
5
6
7
8
9
10
11
12  module.exports = productsRouter;
```

Figura 2 - Rotas

controllers > JS products.js > ...

```
1   const apiResponse = require('../utils/response/apiResponse');
2   const Products = require('../data/entities/products');
3
4   exports.getAll = async (req, res) => {···
15  }
16
17  exports.getById = async (req, res) => {···
30  }
31
32  exports.create = async (req, res) => {···
49  }
50
51  exports.update = async (req, res) => {···
72  }
73
74  exports.delete = async (req, res) => {···
92  }
```

Figura 3 - Controller Produtos

1.1 - Complete o ficheiro de rotas dos produtos.

1.2 - Explique cada uma das linhas do ficheiro anterior

1.3 - Desenvolva um ficheiro JSON que permita guardar a informação dos produtos e escreva o código para cada um dos métodos do controller products.

2. O Resultado final da prova escrita deve ser colocada no github sendo partilhado o link como resposta à prova

Bom trabalho!

António Lira Fernandes

Cofinanciado por: