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: A

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 Envio da Prova Escrita.

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.

   - A imagem a seguir representada é a base de dados e linguagem de programação que corre no servidor.

   A imagem descrever os diferentes componentes e tecnologias envolvidas no desenvolvimento de aplicações web, tanto no lado do cliente (front-end) quanto no lado do servidor (back-end).

Cofinanciado por:

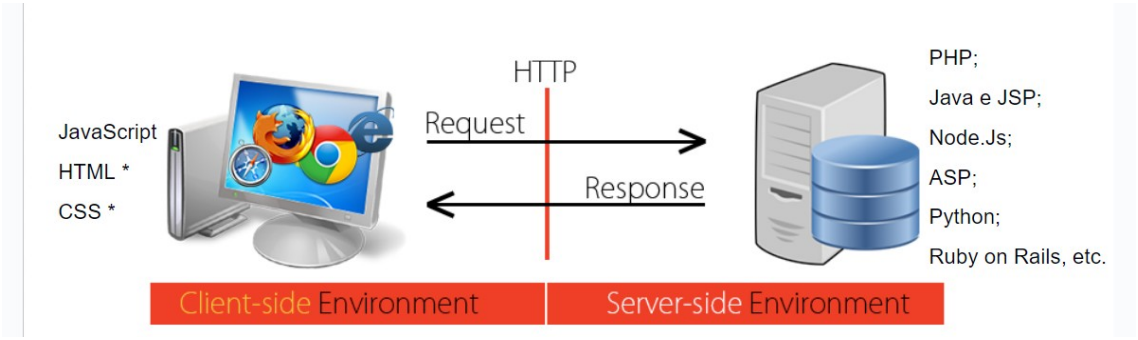

Figura 1 - Imagem a ser comentada

2. Crie um protocolo para os alunos do IPVC para a realização de Exames. Para que servem os protocolo e dê um exemplo

   - Um protocolo estabelece diretrizes para a realização do mesmo, em que se organize por etapas o que se vai fazer.

   1- Preparação Antes do Exame

   2- Dia do Exame

   3- Realização do Exame

   4. Após Exame

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

   No primeiro caso temos getElement e nos dois seguintes getElements pois tem haver com a quantidade de elementos que cada método pode retornar:

Cofinanciado por:

document.getElementById(id): Retorna um único elemento que possui o ID especificado.

document.getElementsByTagName(tagName): Retorna uma coleção (HTMLCollection) de todos os elementos que possuem o nome da tag especificada.

document.getElementsByClassName(className):Retorna um coleção (HTMLCollection) de todos os elementos que possuem a classe especificada.

Exemplos:

document.getElementById(id):

```
<!DOCTYPE html>
<html>
<head>
  <title>Exemplo getElementById</title>
</head>
<body>
  <div id="main-content">Este é o conteúdo principal.</div>
  <script>
    var element = document.getElementById("main-content");
    element.style.color = "red";
  </script>
</body>
</html>
```

document.getElementsByTagName(tagName):

```
<!DOCTYPE html>
<html>
<head>
  <title>Exemplo getElementsByTagName</title>
```

```
</head>
<body>
  <p>Primeiro parágrafo.</p>
  <p>Segundo parágrafo.</p>
  <script>
    var elements = document.getElementsByTagName("p");
    for (var i = 0; i < elements.length; i++) {
      elements[i].style.fontWeight = "bold";
    }
  </script>
</body>
</html>
```

document.getElementsByClassName(className):

```
<!DOCTYPE html>
<html>
<head>
  <title>Exemplo getElementsByClassName</title>
</head>
<body>
  <div class="highlight">Texto destacado 1</div>
  <div class="highlight">Texto destacado 2</div>
  <script>
    var elements = document.getElementsByClassName("highlight");
    for (var i = 0; i < elements.length; i++) {
```

```
            elements[i].style.backgroundColor = "green";
        }
    </script>
</body>
</html>
```

2. Cria uma estrutura em JSON para registar UC e docentes do teu curso. Faz um XML para a mesma estrutura. Comenta os resultados.

JSON

```
{
  "curso": "Engenharia Redes e Sistemas Computadores",
  "unidadesCurriculares": [
    {
      "nome": "Programação Web",
      "codigo": "TI101",
      "docentes": [
        {
          "nome": "Dr. Pedro Silva",
          "email": "pedro.silva@ipvc.pt"
        },
        {
          "nome": "Prof. Ana Costa",
          "email": "ana.costa@ipvc.pt"
        }
```

```
    ]
  },
  {
    "nome": "Sistemas Operativos",
    "codigo": "TI102",
    "docentes": [
      {
        "nome": "Dr. Miguel Sousa",
        "email": "miguel.sousa@ipvc.pt"
      }
    ]
  },
  {
    "nome": "Redes de Computadores",
    "codigo": "TI103",
    "docentes": [
      {
        "nome": "Prof. Rita Pereira",
        "email": "ritapereira@ipvc.pt"
      },
      {
        "nome": "Dr. Manuel Antonio",
        "email": "antoniomny@ipvc.pt"
      }
    ]
```

```
      }
    ]
}
```

XML

```
<curso>
  <nome>Engenharia Redes e Sistemas de Computadores</nome>
  <unidadesCurriculares>
    <unidadeCurricular>
      <nome>Programação Web</nome>
      <codigo>TI101</codigo>
      <docentes>
        <docente>
          <nome>Dr. Pedro Silva</nome>
          <email>pedro.silva@ipvc.pt</email>
        </docente>
        <docente>
          <nome>Prof. Ana Costa</nome>
          <email>ana.costa@ipvc.pt</email>
        </docente>
      </docentes>
    </unidadeCurricular>
    <unidadeCurricular>
      <nome>Sistemas Operativos</nome>
      <codigo>TI102</codigo>
```

```
        <docentes>
          <docente>
            <nome>Dr. Miguel Sousa</nome>
            <email>miguel.sousa@ipvc.pt</email>
          </docente>
        </docentes>
      </unidadeCurricular>
      <unidadeCurricular>
        <nome>Redes de Computadores</nome>
        <codigo>TI103</codigo>
        <docentes>
          <docente>
            <nome>Prof. Rita Pereira</nome>
            <email>ritapereira@ipvc.pt</email>
          </docente>
          <docente>
            <nome>Dr. Manuel Antonio</nome>
            <email>antoniomnys@ipvc.pt</email>
          </docente>
        </docentes>
      </unidadeCurricular>
    </unidadesCurriculares>
</curso>
```

Parte III (20 valores)

1. Qual a diferença entre:

```
<img
src="https://www.ipvc.pt/wp-content/uploads/2020/11/logo
_ipvc_svg.svg" alt="logo da IPVC">
```

e

```
<img src="/images/logo_ipvc_svg.svg" alt="logo da esm">
```

- A diferença entre as duas tags <img> apresentadas tem haver como os caminhos (src) para as imagens são especificados. No primeiro exemplo é um URL completo que especifica a localização exata do recurso da web enquanto já no segundo especifica um caminho relativo ao diretório raiz do site.

1. Para que serve o atributo

```
<html lang="pt">
```

- Serve para especificar o idioma principal do conteúdo que está na página web.

Parte IV (30 valores)

1. Considere os seguintes estilos.

```
1. p {
2.     text-align: center;
3.     color: red;
```

```
}
#para1 {
  text-align: center;
  color: blue;
}
.center {
  text-align: center;
}
p.center {
  text-align: center;
  color: black;
}
```

Construa uma página html que use os estilos. Apresente o html e o resultado final

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.
1. Cada card tem de ter um botão “ver mais” para ver mais detalhes.

Parte VI (50 valores)

1. Considere as imagens seguintes.

Cofinanciado por:

routes > 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 > 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

Cofinanciado por:

1.1 - Complete o ficheiro de rotas dos produtos.

```
const productsRouter = require('express').Router();
const controller = require('../controllers/products');
const authMiddleware = require('../middlewares/auth/auth');

productsRouter.get('/', controller.getAllProducts);

productsRouter.get('/:id', controller.getProductById);

productsRouter.post('/', authMiddleware.authenticate, controller.createProduct);

productsRouter.put('/:id', authMiddleware.authenticate, controller.updateProductById);

productsRouter.delete('/:id',authMiddleware.authenticate, controller.deleteProductById);

module.exports = productsRouter;
```

1.2 - Explique cada uma das linhas do ficheiro anterior

```
const productsRouter = require('express').Router();
const controller = require('../controllers/products');
const authMiddleware = require('../middlewares/auth/auth');

// Rota para obter todos os produtos
productsRouter.get('/', controller.getAllProducts);
```

```
// Rota para obter um produto pelo ID
productsRouter.get('/:id', controller.getProductById);

// Rota para criar um novo produto
productsRouter.post('/', authMiddleware.authenticate, controller.createProduct);

// Rota para atualizar um produto pelo ID
productsRouter.put('/:id',authMiddleware.authenticate,
controller.updateProductById);

// Rota para deletar um produto pelo ID
productsRouter.delete('/:id',authMiddleware.authenticate,
controller.deleteProductById);

module.exports = productsRouter;
```

1.3 - Escreva o código para cada um dos métodos do controller products. Fica a seu critério definir a tabela produtos e atende a que a base de dados é em postgresql e deve ser usado prisma para executar as consultas.

```
const apiResponse = require('../utils/response/apiResponse');
const { PrismaClient } = require('@prisma/client');

const prisma = new PrismaClient();

exports.getAll = async (req, res) => {
  try {
    const products = await prisma.products.findMany();
```

```
        return apiResponse.successResponse(res, "Lista de produtos que foi recuperada com sucesso", products);
    } catch (error) {
        console.error("Erro ao recuperar lista de produtos:", error);
        return apiResponse.errorResponse(res, "Erro ao recuperar lista de produtos");
    }
};

exports.getById = async (req, res) => {
    const { id } = req.params;
    try {
        const product = await prisma.products.findUnique({
            where: {
                id: parseInt(id)
            }
        });
        if (!product) {
            return apiResponse.notFoundResponse(res, "Produto não encontrado");
        }
        return apiResponse.successResponse(res, "Detalhes do produto recuperados com sucesso", product);
    } catch (error) {
        console.error("Erro ao recuperar detalhes do produto:", error);
        return apiResponse.errorResponse(res, "Erro ao recuperar detalhes do produto");
    }
};
```

```
exports.create = async (req, res) => {
  const { name, price } = req.body;
  try {
    const newProduct = await prisma.products.create({
      data: {
        name,
        price
      }
    });
        return apiResponse.successResponse(res, "Produto foi criado com sucesso", newProduct);
  } catch (error) {
    console.error("Erro ao criar produto:", error);
    return apiResponse.errorResponse(res, "Erro ao criar produto");
  }
};

exports.update = async (req, res) => {
  const { id } = req.params;
  const { name, price } = req.body;
  try {
    const updatedProduct = await prisma.products.update({
      where: {
        id: parseInt(id)
      },
```

```
            data: {
              name,
              price
            }
        });
            return apiResponse.successResponse(res, "Produto atualizado com sucesso", updatedProduct);
    } catch (error) {
        console.error("Erro ao atualizar produto:", error);
        return apiResponse.errorResponse(res, "Erro ao atualizar produto");
    }
};

exports.delete = async (req, res) => {
    const { id } = req.params;
    try {
        await prisma.products.delete({
            where: {
                id: parseInt(id)
            }
        });
        return apiResponse.successResponse(res, "Produto foi excluído com sucesso");
    } catch (error) {
        console.error("Erro ao excluir produto:", error);
        return apiResponse.errorResponse(res, "Erro ao excluir produto");
    }
```

};

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