

Parte I	1
Parte II	1
Parte III	3
Parte IV	3
Parte V	3
Parte VI	4

Parte I

1. Na Web consegue interpretar o conteúdo HTML5, JavaScript (e css), podemos ter aplicações desenvolvidas em Java, temos o Web Service que é a parte servidor onde recebe os pedidos vindo os clientes (Web e App), dando respostas em HTML5, JSON, XML etc. O Web Service é o responsável por tratar da informação, sendo este o único que fica ligado à Base de Dados.
2. O protocolo é uma série de instruções que devem ser seguidas para que algo aconteça.
 - a. O aluno possui um identificador (Número de Aluno)
 - b. O aluno se dirige à cantina ou a alguma aplicação que venda a senha
 - c. O aluno aguarda pela sua vez
 - d. O aluno escolhe a sua refeição
 - e. O aluno paga a senha
 - f. O aluno aguarda na fila para a cantina
 - g. O aluno mostra a sua senha ou passa o cartão
 - h. O aluno recebe a refeição
 - i. O aluno almoça

Parte II

1. Porque o ID é único por página e os outros (TagName e ClassName) pode haver múltiplos.

```
<div id="id_example"></div> (Apenas vai ler este)  
<div id="id_example"></div> (Este será Ignorado)
```

```
console.log(document.getElementById("id_example");  
(Apenas o primeiro que encontrar na página com esse id será lido)
```

```
<div class="id_example"></div>  
<div class="id_example"></div>  
console.log(document.getElementsByClassName("id_example");  
(Ambos vão ser lidos, o comando irá devolver um Array com ambos)
```

```
<div></div>  
<div></div>  
console.log(document.getElementsByTagName("div");  
(Ambos vão ser lidos, o comando irá devolver um Array com ambos)
```

2. .

JSON:

```
{  
  "atores": [  
    {  
      "nome": "Tom Cruise",  
      "nascimento": "03/07/1962",  
      "filmes": [  
        {  
          "titulo": "Missão Impossível",  
          "ano": 1996  
        },  
        {  
          "titulo": "Top Gun",  
          "ano": 1986  
        }  
      ]  
    },  
    {  
      "nome": "Brad Pitt",  
      "nascimento": "18/12/1963",  
      "filmes": [  
        {  
          "titulo": "Clube da Luta",  
          "ano": 1999  
        },  
        {  
          "titulo": "Sr. e Sra. Smith",  
          "ano": 2005  
        }  
      ]  
    }  
  ]  
}
```

```
}  
]  
}
```

XML:

```
<?xml version="1.0" encoding="UTF-8" ?>  
<root>  
  <atores>  
    <nome>Tom Cruise</nome>  
    <nascimento>03/07/1962</nascimento>  
    <filmes>  
      <titulo>Missão Impossível</titulo>  
      <ano>1996</ano>  
    </filmes>  
    <filmes>  
      <titulo>Top Gun</titulo>  
      <ano>1986</ano>  
    </filmes>  
  </atores>  
  <atores>  
    <nome>Brad Pitt</nome>  
    <nascimento>18/12/1963</nascimento>  
    <filmes>  
      <titulo>Clube da Luta</titulo>  
      <ano>1999</ano>  
    </filmes>  
    <filmes>  
      <titulo>Sr. e Sra. Smith</titulo>  
      <ano>2005</ano>  
    </filmes>  
  </atores>  
</root>
```

Parte III

1. <p> -> Parágrafo, avança uma linha, texto com espaços formatados.
<pre> -> Texto Pré Formatado, mostra o texto exatamente como está no código
2. Identificar que a página está a utilizar UTF-8 (Identificar os caracteres, exemplo: ç)

Parte IV

1. Github - parte_iv/index.html

Parte V

1. Github - parte_v/index.html
2. Github - parte_v/index.html

Parte VI

1. .
 - a.

```
productsRouter.get("/", controller.getAll);
productsRouter.get("/:id", controller.getById);
productsRouter.post("/", authMiddleware, controller.create);
productsRouter.put("/:id", authMiddleware, controller.update);
productsRouter.delete("/:id", authMiddleware, controller.delete);
```
 - b.

```
const productsRouter = require("express").Router(); // Importar o Router da
biblioteca express
const controller = require("../controllers/products"); // Importar os controllers
(Funções que fazem a lógica de cada rota)
const authMiddleware = require("../middleware/auth/auth"); // Importar o
middleware de autenticação (Função intermediária para verificar sempre que
houver pedido para a rota)
```



```
productsRouter.get("/", controller.getAll); // Rota para buscar todos os produtos
productsRouter.get("/:id", controller.getById); // Rota para buscar um produto
específico pelo id
productsRouter.post("/", authMiddleware, controller.create); // Rota para criar um
novo produto (Necessário estar autenticado)
productsRouter.put("/:id", authMiddleware, controller.update); // Rota para
atualizar um produto pelo id (Necessário estar autenticado)
productsRouter.delete("/:id", authMiddleware, controller.delete); // Rota para
apagar um produto pelo id(Necessário estar autenticado)
```



```
module.exports = productsRouter; // Exportar o router para ser utilizado no
ficheiro principal ou em outro router
```

- c. Github - parte_vi/controller.js
(Possui o servidor testes em parte_vi/exp_server)
(Utilizado *pnpm run dev* para executar)

- 2. <https://github.com/diogocastrodev/teste-pw>