Centro Federal de Educação Tecnológica - Unidade Nova Friburgo
Bacharelado em Sistemas de Informação
Prof. Thiago Delgado Pinto

Nota - até 10,0
3,1

PROGRAMAÇÃO DE CLIENTES WEB – PROVA 2 – 2022-2

Data: 05/01/23 Aluno(a): Brayan Mello

Atenção:
1. Questões entregues à lápis não têm direito à revisão.
2. Ao usar caneta esferográfica, usar tinta azul ou preta.
3. Usar EcmaScript >= 6 ou TypeScript, exceto se a questão solicitar usar algo diferente.
4. Usar EcmaScript Modules (ESM) para realizar importações e exportações de declarações.
5. Não usar frameworks ou bibliotecas de código externas.

0,2 1) [1,0] Analise cada alternativa a seguir e assinale "V" se considerá-la verdadeira e "F" se considerá-la falsa.
E a) [0,2] (V) Um arquivo que utilize *EcmaScript Modules* (ESM) pode exportar de forma `default` quantas declarações necessitar.
E b) [0,3] (F) Uma função com *async* encapsula em uma promessa (`Promise`) resolvida qualquer retorno que não seja uma promessa; e em uma promessa rejeitada qualquer exceção que seja lançada.
E c) [0,3] (V) O declarar uma *tag* `<script>` para um arquivo JavaScript, usar a propriedade *async* faz seu código ser executado sem bloquear a página.
✓ d) [0,2] (V) O declarar várias *tags* `<script>` sucessivas usando *defer*, para diferentes arquivos JavaScript, os arquivos podem ser carregados/baixados fora de ordem, mas são executados em ordem.

1,0 2) [4,0] Considere a existência de um servidor RESTful que utilize uma API em formato JSON – tal qual aquela criada pelo software JSON Server – e que esteja executando no endereço especificado no arquivo "`api.js`", abaixo:

```
// api.js
export const API = 'http://localhost:3000';
```

i) Considere que o navegador possua um objeto "`navigator`" que possua uma propriedade "`onLine`", que retorne `true` se o navegador estiver *online* e false caso esteja *offline*. Por exemplo:

```
console.log( navigator.onLine ? 'Você está online' : 'Você está offline' );
```

ii) Considere a existência da página HTML abaixo, assim como se encontra:

```
<!DOCTYPE html><html lang="pt"> <head> <meta charset="UTF-8" />
    <title>Cadastro de Game</title>
    <script src="src/cadastrar-game.js" type="module" defer ></script>
</head>
<body>
    <h1>Cadastro de Game</h1>
    <form>
        <label for="nome" >Nome:</label> <input type="text" id="nome" />
        <label for="fab" >Fabricante:</label> <input type="text" id="fab" />
        <label for="dl" >Nº de downloads:</label> <input type="text" id="dl" />
        <output></output> <!-- Mensagens devem ser exibidas no output -->
        <button type="button" id="salvar" >Salvar</button>
    </form>
</body>
</html>
```

Sem **necessitar** utilizar MVC, implemente o comportamento do cadastro de um game no recurso "`/games`" do servidor anteriormente indicado, cujo endereço deve ser obtido da constante API (vide código acima). Um game possui os atributos "nome" (*string* de 2 a 60 caracteres), "fabricante" (*string* de 2 a 30 carcteres) e "downloads" (número natural, ou seja,

inteiro >= 0), que devem ser validados antes do cadastro. Antes de cadastrar o game, ele deve ser acrescentado em um *array* (de games) no `localStorage`, em uma chave "games". Certifique-se de verificar se o *array* já existe, antes de acrescentá-lo. Então, caso o navegador se encontre *online* (conforme a propriedade indicada anteriormente, **todos os games** que porventura estiverem armazenados no `localStorage` devem ser cadastrados no servidor, um-a-um. Apenas se tudo ocorrer com sucesso, o *array* em `localStorage` deve ser esvaziado. Quaisquer erros devem ser devidamente apresentados ao usuário (em *output*). Utilize `async/await` para esta questão e **não utilize** os métodos `then()` e `catch()`.

0,5

3) [1,0] Converta o todo o código abaixo para **não utilizar** `async` e `await` e utilizar os métodos `then()` e `catch()`.

```
async function brindes() {
    if ( Math.random() <= 0.5 ) {
        return [ 'Chocolate', 'Paçoca' ];
    }
    throw new Error( 'Sem brindes hoje.' );
}
( async () => {
    try {
        console.log( await brindes() );
    } catch ( err ) {
        console.log( 'Erro: ', err.message );
    }
} )();
```

1,3

4) [2,0] Usando o **modelo MVC**, considere o arquivo HTML abaixo e apresente os games na tabela, considerando o servidor e arquivo "api.js" da questão 2. Considere também que cada game retornado do servidor possui um atributo "id", além dos demais.

```
<!DOCTYPE html><html lang="pt"> <head> <meta charset="UTF-8" /> <title>Games</title>
    <script src="src/controladora-listagem-game.js" type="module" defer ></script> </head>
<body>
    <h1>Listagem de Games</h1>
    <table>
        <thead> <tr><th>Id</th><th>Nome</th><th>Fabricante</th><th>Downloads</th></tr></thead>
        <tbody></tbody>
    </table>
</body>
</html>
```

0,1

5) [2,0] Considere o código HTML abaixo e implemente o código JavaScript que faça com que o botão "Remover" remova o game selecionado pelo usuário do servidor RESTful indicado na questão 2, bem como a opção de dentro do `<select>`. Cada opção do `<select>` mantém o valor do "id" do game. Lembre-se que para remover um recurso em uma API RESTful, o id do recurso deve ser passado em sua URL. Apresente eventuais erros – ou uma mensagem de sucesso – para o usuário, sem usar o console.

```
<button>Remover</button>
<select id="game" >
    <option value="1" >Elder Ring</option>
    <option value="2" >The Witcher 3: Wild Hunt</option>
    <option value="3" >God of War: Ragnarok</option>
</select>
<script src="src/remover-game.js" type="module" defer ></script>
```

Sintaxe simplificada de algumas funções ou métodos que podem ser úteis – explicação não será fornecida:

```
navigator.onLine: boolean
JSON.parse( texto: string ): object
JSON.stringify( o: object ): string
Number.isNaN( valor: any ): boolean
fetch( url: string, opcoes: object ): Promise< Response >
```

Boa prova!

(continuação da 4)

async

const await try{

games = await service.todos();

Tabela = view.incluirNaTabela(games);

} catch(err){ ... }

}

}

Falta exibição de possíveis erros na visão

Controller = new gamesController();

controller.iniciar();

}

2-

import {API} from "./api.js";

async function cadastrar(){

const nome = document.getElementById('nome').value;

const fab = document.getElementById('fab').value;

const dl = document.getElementById('dl').value;

if (nome.lenght >=2 && nome.lenght <= 60 && typeof(nome) == 'string'){

continue;

} else {

Throw new Error("nome deve ter de 2 a 60 caracteres e ser uma string");

}

if (fab.lenght >= 2 && fab.lenght <= 30 && typeof(fab) == 'string') {

continue;

} else {

Throw new Error("fabricante deve ter de 2 a 30 caracteres e ser uma string");

}

if (isNaN(dl) == false && isInt(dl) == true){

continue;

} else {

Throw new Error('download deve ser um inteiro positivo');

(continuação da 2)

games = [];

game = {
"nome": nome,
"fabricante": fab,
"downloads": dl
}

€ .value

games.push(game);

localStorage.setItem(JSON.parse(games));

€ Precisa checar localStorage antes

try {
const response = await fetch(`${API}/games`),
{ method: 'Post',
headers: {
"content-type": "application/json"
},
body: game € JSON.stringify(game)
}
if (response.status >= 400)
throw new Error ("não foi possível cadastrar");
return response.json();
}

→ Quem está capturando? € a exibição da mensagem?

INCOMPLETO

5-

→ faltou importar API

function async remover() {
try {
const response = await fetch(`${API}/games` + id(),
{
method: 'DELETE'
})

De onde vem?

€ Ou use await no fetch, em função com async, ou use then()

if (response.status >= 400)
throw new Error ('não foi possível deletar o jogo');

return response.json(); € // Não retorna
}catch (err){
...
?
}

3- function trufas(){

```
    return new Promise((resolve, reject) => {
        if (math.random()<= 0.5)
            Promise.resolve(['chocolate','beijo']);
        else
            Promise.reject(new Error('sem trufas hoje'));
    });
}

trufas().then(resolve => {
    console.log(resolve)
}).catch( err => {
    console.log(throw new error("erro:", err.message));
});
```

4- (service)

```
import { API } from "./api.js";
export class gameService {
    async fetches() {
        const response = await fetch(`${API}/games`);
        if (response.status >= 400){
            throw new Error('erro ao obter games');
        } else {
            return response.json();
        }
        return response;
    }
}
```

(view)

```
import {gameService} from "./game.service.js";
export class gameView {

}
```

(continuação da 7)

Brayan

```
incluirNaTabela(games);

for (const g of games) {
const TdId = document.createElement('Td');
      TdId.innerText = g.id;
const TdNome = document.createElement('Td');
      TdNome.innerText = g.nome;
const TdFabricante = document.createElement('fabricante');
      TdFabricante.innerText = g.fabricante;
const TdDo = document.createElement('Td');
      TdDo.innerText = g.download;
const Tbody = document.querySelector('Tbody');  (ANTES)
const linha = document.createElement('tr');
      linha.append(TdId, TdNome, TdFabricante, TdDo);
      Tbody.append(linha);
  }
}
}
```

(controller)

```
import { gamesService } from "./games-service.js";
import { gamesView } from "./games-view.js";

export class gamesController {
      service = null;
      view = null;
      constructor() {
      this.service = new gamesService();
      this.controller = new gamesController();
   }
}
```