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

Nota - até 10,0

PROGRAMAÇÃO DE CLIENTES WEB – PROVA 2 – 2021-2

Data: _____ Aluno(a):_____

Para esta prova:

- Use EcmaScript 6 ou superior, exceto quando explicitamente solicitado outra versão.
- Use EcmaScript Modules (ESM) para realizar importações e exportações de declarações.
- Não use frameworks ou bibliotecas de código externas.

Considere um servidor web com uma API RESTful, que utilize dados em formato JSON, tanto para receber quanto para retornar requisições HTTP. Esse servidor pode ser simulado através do JSON Server, que pode ser configurado pelos parâmetros abaixo:

Instalação:

```
npm init --yes
npm i -D json-server
```

Execução do json-server com o arquivo db.json, listado adiante:

```
npx json-server --watch db.json
```

Arquivo db.json:

```
{
    "games": [
        { "id": 1, "nome": "Minecraft",             "ano": 2011, "fabricante": "Mojang Studios"    },
        { "id": 2, "nome": "Grand Theft Auto V",    "ano": 2013, "fabricante": "Rockstar North"    },
        { "id": 3, "nome": "PUBG: Battlegrounds",   "ano": 2017, "fabricante": "PUBG Corporation"  },
        { "id": 4, "nome": "Red Dead Redemption 2", "ano": 2018, "fabricante": "Rockstart Studios" }
    ],

    "generos": [
        { "id": 1, "descricao": "Sobrevivência" },
        { "id": 2, "descricao": "Ação/Aventura" },
        { "id": 3, "descricao": "Esportes" },
        { "id": 4, "descricao": "Battle Royale" },
        { "id": 5, "descricao": "Plataforma" },
        { "id": 6, "descricao": "Corrida" },
        { "id": 7, "descricao": "RPG" }
    ]
}
```

Ao executar o JSON Server, é comum que sua API fique disponível em `http://localhost:3000/` – assim, considere essa URL.

Esse servidor disponibiliza os seguintes recursos:

- "`/games`", que contém alguns jogos eletrônicos conhecidos;
- "`/generos`", que contém alguns gêneros de jogos eletrônicos.

1) [1,0] Usando `fetch`, `async`, `await` e a classe `Promise` – sem usar os *métodos* `then` ou `catch` – crie uma página `q1.html`, que contenha um script que consulte (via GET) concorrentemente todos os recursos disponíveis no servidor e informe qual conseguiu responder mais rápido. Não exiba mensagens no console.

2) [2,0] Usando `fetch`, sem usar `async` e `await`, crie uma página `q2.html` que consulte os recursos (via GET) de "`/games`" e preencha dinamicamente uma tabela HTML com seus dados. Em caso de erro, exiba uma mensagem em uma `div`, contendo o código de status HTTP retornado. Não utilize a propriedade `innerHTML` na solução. Não exiba mensagens no console.

3) Usando a *Location API*, Expressões Regulares (para tratar URLs) e o modelo de separação de responsabilidades *Model-View-Controller* (MVC), crie uma pequena *Single-Page Application* (SPA) que:

   a) [2,0] Contenha os seguintes arquivos e comportamentos (que devem ser criados):
      - `index.html`, que é a **página inicial** e contém em seu corpo uma *tag* `main`, cujo conteúdo deve ser dinamicamente trocado, de acordo com a URL;
      - `welcome.html` (fornecido com a prova), que é uma **página de boas-vindas**, carregada por padrão ao acessar o endereço da página inicial ("`/`"), ou o endereço "`/welcome`";
      - `404.html` (fornecido com a prova), que é uma página que indica um **endereço não encontrado**, carregada quando for acessado *qualquer endereço não tratado pela aplicação*, ou o endereço "`/404`". Nessa página, **adicione** um botão "Voltar" que use a *History API* para voltar à página anterior.

   Considere que qualquer URL pode ser acessada em letras maiúsculas ou minúsculas e possa terminar com uma barra.

   Lembrete: Conteúdos em JavaScript necessitam ser carregados dinamicamente, após o carregamento do conteúdo HTML.

   b) [3,5] Ao acessar, pelo navegador, o endereço "`/games/:id`", em que "`:id`" deve ser um número natural qualquer (ex. "2"), carrega o arquivo `form.html` (fornecido com a prova) e, sem seguida, carrega o respectivo recurso do servidor RESTful (ou seja, o recurso com o id indicado) e o exibe no formulário. Caso o recurso não exista, deve ser exibida uma mensagem indicando o fato e ser feito o redirecionamento da navegação para "`/welcome`".

   Ao clicar no botão "Alterar", os dados do formulário devem ser enviados para alteração no servidor, em formato JSON. Em caso de sucesso ou erro, uma mensagem de alerta deve ser exibida, indicando os detalhes do ocorrido. Restrições para validação, antes do envio:
      a. O **nome** deve ter entre 2 e 80 caracteres;
      b. O **ano** deve ser um valor numérico compreendido entre 1970 e 2022. Seu valor deve ser convertido para número;
      c. O **fabricante** deve ter entre 2 e 60 caracteres.

   Lembrete: Regras de validação fazem parte do domínio do negócio.

   c) [1,5] Adicione, ao formulário da questão anterior, um botão "Remover" que permita remover o respectivo recurso do servidor. Em caso de sucesso ou erro, uma mensagem de alerta deve ser exibida, indicando os detalhes do ocorrido. Após uma mensagem de sucesso, deve-se **redirecionar a navegação** do usuário para a página inicial da aplicação.

Boa prova!