

Questão 1

IMPORTANTE: Não faça o commit da solução no repositório do Projeto 2. Faça o commit da solução deste simulado em uma branch separada.

Nesta avaliação você deverá implementar uma nova funcionalidade no seu Projeto 2. A funcionalidade consiste em mostrar na página inicial um email de usuário obtido através de uma API.

Utilizaremos a API do servidor de exercícios JS para obter o email de usuário. Mais detalhes dos passos necessários serão descritos abaixo.

Para obter o email de usuário será necessário seguir os passos a seguir:

- **[2 pts]** Obter um token de acesso para o seu usuário na API indicada;
 - **[4 pts]** Fazer uma requisição GET solicitando a lista de exercícios;
 - **[1 pts]** Obter o email de usuário contido nas informações do exercício `'nome-do-usuario'`
 - **[3 pts]** Adicionar o nome de usuário obtido na página principal do projeto
-

Restrições:

- As requisições devem ser feitas através do código (não utilizando alguma ferramenta externa como o Postman. Utilize somente para teste);
 - Você deve implementar essa funcionalidade usando o seu Projeto 2 como base e as funcionalidades novas devem ser apresentadas na página inicial. Ou seja, você **não** deve remover funcionalidades existentes.
 - Se você acredita que existe um motivo razoável para não seguir algumas das restrições acima, converse com a professora.
 - **Importante:** Programas que não puderem ser executados (ex: erros de sintaxe ou que impeçam o servidor de executar) receberão nota zero. O mesmo vale se você não utilizar o código do projeto 2.
-

Instruções

Abaixo explicamos com mais detalhes cada um dos passos.

- **Obtendo o token**

O token pode ser obtido realizando uma requisição POST ao endpoint

<https://tecweb-js.insper-comp.com.br/token>

A requisição deve enviar no corpo (*body*) o seguinte conteúdo no formato *JSON*:

`{"username": "NOME DE USUÁRIO"}`. Onde "NOME DE USUÁRIO" é o seu usuário Insper.

Além disso, a requisição também deve definir o headers da requisição: `{ headers: { "Content-Type": "application/json", "Accept": "application/json" } }`

A resposta será um objeto *JSON* contendo o token de acesso no seguinte formato

`{"accessToken": "TOKEN GERADO"}`.

Faça um `console.log` com o token obtido (mantenha o `console.log` na solução para que eu possa verificar).

- **Solicitando a lista de exercícios**

Faça uma requisição *GET* para o endpoint

<https://tecweb-js.insper-comp.com.br/exercicio>

Você deve obrigatoriamente enviar o *token* obtido na etapa anterior na chave headers da requisição. O headers do *JSON* enviado deve seguir o formato

`{ headers: { "Content-Type": "application/json", "Accept": "application/json", "Authorization": "Bearer SEUTOKEN" } }`

(substituindo `SEUTOKEN` pelo token obtido na etapa anterior).

Em caso de sucesso, essa requisição devolve uma lista de exercícios.

Faça um `console.log` com a lista de exercícios (mantenha o `console.log` na solução para que eu possa verificar).

- **Obtendo o email de usuário**

Agora que temos a lista de exercícios, queremos obter o email de usuário contido no exercício chamado `nome-do-usuario`.

Vasculhe a lista de exercícios que foi impressa no console do navegador e descubra como acessar o email do usuário oferecido pelo exercício.

Faça um `console.log` com o email do usuário (mantenha o `console.log` na solução para que eu possa verificar).

- **Adicionar o nome de usuário na página principal**

Agora que você conseguiu acessar o email do usuário oferecido no exercício, adicione este email em qualquer lugar da página principal do seu projeto 2. (Não é necessário implementar nenhum estilo CSS)

Sempre que a página da aplicação for atualizada, uma nova requisição deve ser realizada apresentando um novo email.

Questão 2

Nesta questão você deverá implementar duas novas rotas no seu Projeto 2 Django REST.

Queremos implementar novas endpoints em nossa API para criar, listar e deletar *Fun Facts*.

Fun Facts são pequenos textos contendo fatos interessantes de qualquer assunto.

Atenção

Para esta questão só estamos interessados em implementar a *endpoint*. Não será necessário utilizar esta funcionalidade no *Frontend*.

Implemente uma nova *view* do Django REST Framework para a rota `api/funfact/`. Esta rota deve ser capaz de receber requisições GET e POST.

- Primeiramente, precisamos criar um modelo novo chamado **FunFact**. Este modelo deve conter apenas um campo texto para armazenar o texto com o fato curioso (**Obs.:** O modelo também deve ter um ID, porém não é necessário especificar o ID).
- Ao receber uma requisição POST, a endpoint deve salvar no banco a informação da FunFact.
Esta requisição será enviada da seguinte forma:
 - requisição: POST
 - endpoint: `api/funfact/`
 - body: `{"fact": "ALGUMA FUNFACT"}`
- Ao receber uma requisição GET, a endpoint deve retornar a lista de todos os dados de Fun Facts registradas no banco de dados.
 - Veja um exemplo abaixo

```
[
  {
    "id": 1,
    "fact": "Um crocodilo não consegue colocar sua língua para fora."
  },
  {
    "id": 2,
    "fact": "O músculo mais forte do corpo humano é a mandíbula."
  }
]
```

- Implemente uma segunda endpoint `api/funfact/ID` (Onde `ID` representa o ID de uma Fun Fact). Esta endpoint deve aceitar apenas requisições DELETE.

Ao receber uma requisição DELETE a API deve deletar do banco de dados a Fun Fact com o ID passado e retornar o código 204 caso a operação seja realizada com sucesso.

Pontuação:

- **[1 pts]** Criar o modelo **FunFact**;
 - **[4 pts]** Implementar a funcionalidade para a requisição POST salvar as informações recebidas conforme descrito no enunciado;
 - **[2 pts]** Implementar a funcionalidade para a requisição GET **FunFact** conforme descrito no enunciado;
 - **[2 pts]** Implementar a funcionalidade para a requisição DELETE **FunFact** conforme descrito no enunciado;
 - **[1 pts]** Retornar o código 204 caso a requisição DELETE seja realizada com sucesso;
-