

MAC0475 - Laboratório de Sistemas Computacionais Complexos

Exercício-Programa 2

Contexto

Nas últimas aulas, tivemos o segundo bloco de tecnologia do curso, focado em ferramentas para desenvolvimento front-end: HTML, CSS, JavaScript, VueJS, NuxtJS, Axios e TailwindCSS. Agora é hora de aplicá-las em conjunto!

Neste exercício-programa, você deve utilizar as tecnologias acima para implementar um único site. A especificação, apresentada a seguir, deverá ser seguida à risca.

Especificação

A tarefa consiste em implementar o fluxo de autenticação e autorização (a.k.a. *Auth*) pela perspectiva do *front-end*. Para facilitar o trabalho, vocês utilizarão uma ferramenta para representar um *back-end* chamada [json-server-auth](#), que adere às técnicas já abordadas anteriormente e que possui uma documentação clara.

Neste EP, espera-se que você implemente páginas web para dar suporte às seguintes funcionalidades do fluxo de *auth*:

1. Cadastro de Usuário (`/sign-up`)
2. Acesso de Usuário (`/sign-in`)
3. Perfil de Usuário (`/users/:id`)

As instruções sobre os elementos que devem ser apresentados estão descritas abaixo.

Seu repositório deve ser gerado usando a ferramenta [create-nuxt-app](#), conforme mostrado em aula. O detalhamento das respostas às perguntas feitas pela ferramenta segue conforme o seguinte:

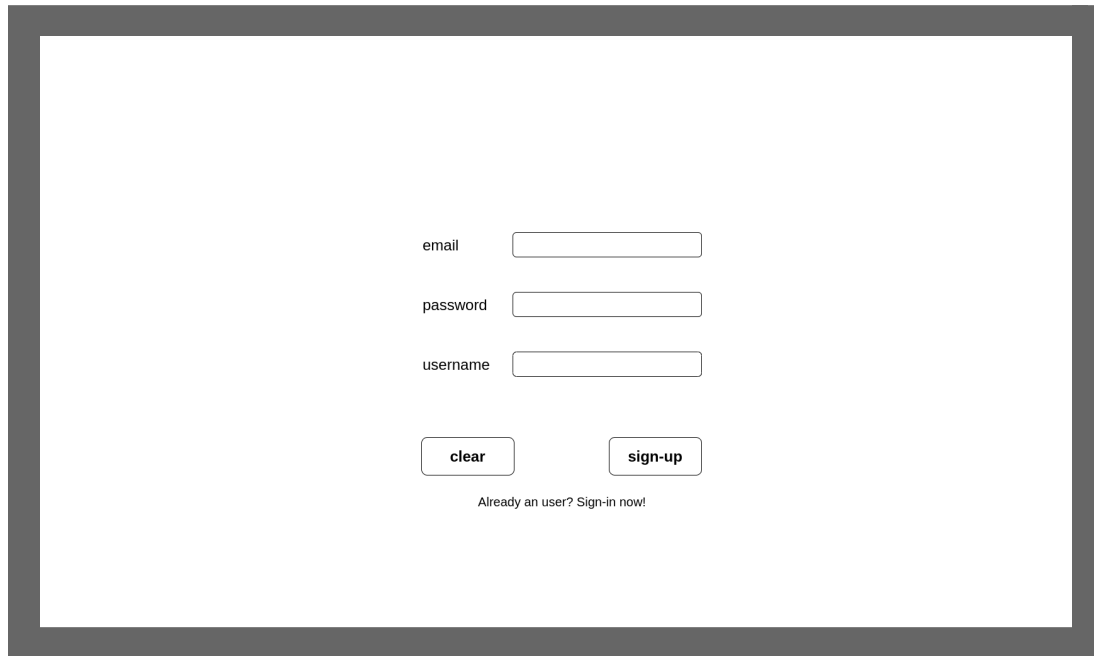
```
create-nuxt-app v3.6.0
✨ Generating Nuxt.js project in pingr
? Project name: pingr
? Programming language: JavaScript
? Package manager: Npm
? UI framework: Tailwind CSS
? Nuxt.js modules: Axios - Promise based HTTP client
? Linting tools: ESLint, Prettier, Lint staged files, StyleLint, Commitlint
? Testing framework: None
? Rendering mode: Universal (SSR / SSG)
? Deployment target: Static (Static/Jamstack hosting)
? Development tools: jsconfig.json (Recommended for VS Code if you're not using typescript)
? Continuous integration: None
? Version control system: Git
```

Atenção: As opções acima destacadas em negrito aparecerão para serem selecionadas interativamente durante a execução do create-nuxt-app. **Utilize as mesmas opções especificadas acima.**

Página Cadastro de Usuário (/sign-up)

Wireframe

Utilize o seguinte [wireframe](#) como referência para sua implementação:



The wireframe shows a registration form with three input fields: 'email', 'password', and 'username'. Below these fields are two buttons: 'clear' and 'sign-up'. At the bottom, there is a link that says 'Already an user? Sign-in now!'.

```
email   
password   
username   
  
   
  
Already an user? Sign-in now!
```

Dados

A página de criação de usuário deverá possibilitar ao usuário preencher três campos:

1. **email**: texto obrigatório, aderente ao formato de um email convencional e único dentre todos os registros conhecidos.
2. **password**: texto alfanumérico, de comprimento entre 8 e 32 (inclusive), permitindo letras maiúsculas e minúsculas, sem qualquer obrigatoriedade entre as escolhas.
3. **username**: texto alfanumérico obrigatório e não vazio que não pode ser iniciado por número.

Ao iniciar o desenvolvimento, você deverá inicializar o [json-server-auth](#) para a criação da API local, conforme instruído pela documentação na ferramenta.

Validação

Ao preencher os campos do formulário, você deverá verificar se os campos atendem às **regras de validação** acima especificados.

Até a validação ser cumprida, você deverá exibir alguma **resposta textual + visual** instruindo o usuário. Por exemplo, você poderá alterar a cor da borda para vermelho indicando que a condição não foi cumprida e exibir uma legenda abaixo do campo explicando qual regra não foi atendida até o momento.

Formulário

Os dados do formulário poderão ser salvos na [Vuex Store](#) via [integração da Vuex com o Nuxt](#) para permitir que eles sejam acessados posteriormente durante o envio do formulário.

Quando o usuário clicar no **botão sign-up**, você deverá utilizar o [plugin do Nuxt para Axios](#) para chamar a rota POST /register do back-end enviando os dados coletados no formulário.

Caso o usuário exista, você receberá uma resposta com código [400 \(BAD REQUEST\)](#) contendo uma string que indica o motivo do erro (e.g., "E-mail already exists").

Nesse caso, você deverá exibir a string que indica o motivo do erro para o usuário, solicitando que ele tente novamente.

Caso o usuário NÃO exista, você receberá uma resposta com código [201 \(CREATED\)](#) contendo um [token de acesso](#) no formato [JSON Web Token \(JWT\)](#) que representará o usuário logado.

Nesse caso, você deverá salvar o token de acesso na [Vuex Store](#) via [integração da Vuex com o Nuxt](#) para poder acessá-lo em outras páginas.

Ao final da requisição, você deverá utilizar o [Vue Router](#) via [integração do Vue Router com o Nuxt](#) para redirecionar para a página de perfil.

Detalhes Adicionais

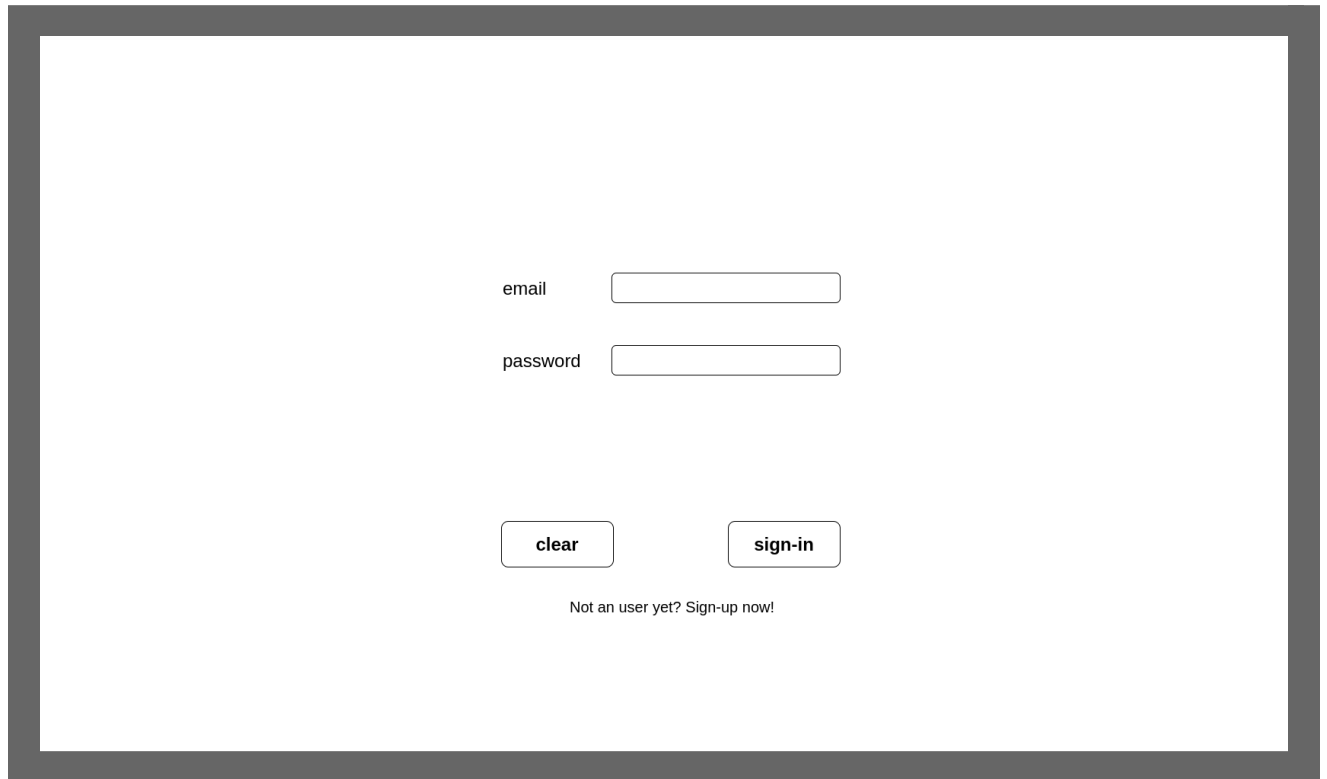
No **botão clear**, você deverá limpar o conteúdo dos campos para que o usuário comece do zero.

No texto "Sign-in now!", você deverá colocar um link para redirecionar para a página /sign-in.

Página Acesso de Usuário (/sign-in)

Wireframe

Utilize o seguinte [wireframe](#) como referência para sua implementação:



The wireframe shows a sign-in page layout. It features two input fields: one for 'email' and one for 'password'. Below these fields are two buttons: 'clear' and 'sign-in'. At the bottom, there is a link that says 'Not an user yet? Sign-up now!'.

email

password

[Not an user yet? Sign-up now!](#)

Dados

A página de criação de usuário deverá possibilitar ao usuário preencher dois campos:

1. **email**: texto obrigatório, aderente ao formato de um email convencional e único dentre todos os registros conhecidos.
2. **password**: texto alfanumérico, de comprimento entre 8 e 32 (inclusive), permitindo letras maiúsculas e minúsculas, sem qualquer obrigatoriedade entre as escolhas.

Ao iniciar o desenvolvimento, você deverá inicializar o [json-server-auth](#) para a criação da API local, conforme instruído pela documentação na ferramenta.

Validação

Ao preencher os campos do formulário, você deverá verificar se os campos atendem às **regras de validação** acima especificados.

Até a validação ser cumprida, você deverá exibir alguma **resposta textual + visual** instruindo o usuário. Por exemplo, você poderá alterar a cor da borda para vermelho indicando que a condição não foi cumprida e exibir uma legenda abaixo do campo explicando qual regra não foi atendida até o momento.

Formulário

Os dados do formulário poderão ser salvos na [Vuex Store](#) via [integração da Vuex com o Nuxt](#) para permitir que eles sejam acessados posteriormente durante o envio do formulário.

Quando o usuário clicar no **botão sign-in**, você deverá utilizar o [plugin do Nuxt para Axios](#) para chamar a rota POST /login do back-end enviando os dados coletados no formulário.

Caso o usuário NÃO exista, você receberá uma resposta com código [400 \(BAD REQUEST\)](#) contendo uma string que indica o motivo do erro (e.g., "Cannot find user").

Nesse caso, você deverá exibir a string que indica o motivo do erro para o usuário, solicitando que ele tente novamente.

Caso o usuário exista, você receberá uma resposta com código [200 \(OK\)](#) contendo um [token de acesso](#) no formato [JSON Web Token \(JWT\)](#) que representará o usuário logado.

Nesse caso, você deverá salvar o [token de acesso](#) na [Vuex Store](#) via [integração da Vuex com o Nuxt](#) para poder acessá-lo em outras páginas.

Ao final da requisição, você deverá utilizar o [Vue Router](#) via [integração do Vue Router com o Nuxt](#) para redirecionar para a página de perfil.

Detalhes Adicionais

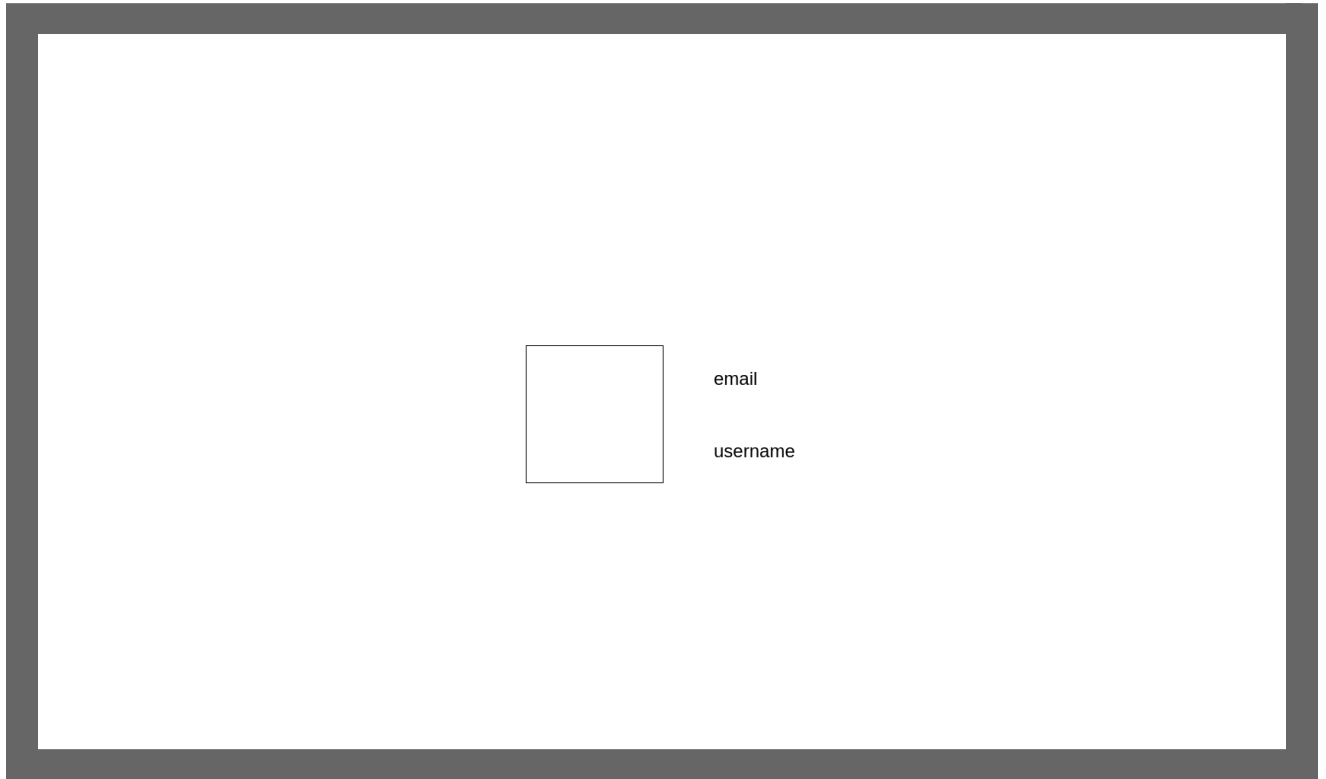
No **botão clear**, você deverá limpar o conteúdo dos campos para que o usuário comece do zero.

No texto "Sign-up now!", você deverá colocar um link para redirecionar para a página /sign-up.

Página Perfil de Usuário (/users/:id)

Wireframe

Utilize o seguinte [wireframe](#) como referência para sua implementação:



Dados

A página de perfil de usuário deverá possibilitar ao usuário visualizar dois campos:

3. **email**: texto obrigatório, aderente ao formato de um email convencional e único dentre todos os registros conhecidos.
4. **username**: texto alfanumérico obrigatório e não vazio que não pode ser iniciado por número.

Adicionalmente, você deverá exibir uma imagem placeholder gerada com o uso do website <https://getavataaars.com>

Acesso

A página do perfil privado deverá ser exibida acessando a rota `/users/:id`

Para recuperar os dados dessa página, você deverá utilizar o [plugin do Nuxt para Axios](#) para chamar a rota `POST /600/users/:id` do back-end enviando os dados coletados no formulário. Para essa requisição, você deverá utilizar o [token de acesso](#) armazenado na [Vuex Store](#) via [integração da Vuex com o Nuxt](#).

Caso não haja token de acesso disponível, você deverá redirecionar para a página `/sign-up` para que o usuário possa fazer seu cadastro.

Caso o ID de usuário NÃO corresponda ao token de acesso, você receberá uma resposta com código [403 \(FORBIDDEN\)](#) contendo uma string que indica o motivo do erro (e.g., "Private resource access: entity must have a reference to the owner id").

Nesse caso, você deverá exibir a string que indica o motivo do erro para o usuário, solicitando que ele tente novamente.

Caso o ID de usuário NÃO exista, você receberá uma resposta com código [401 \(UNAUTHORIZED\)](#) contendo uma string que indica o motivo do erro (e.g., "Cannot read property 'id' of undefined").

Nesse caso, você deverá exibir a string que indica o motivo do erro para o usuário, solicitando que ele tente novamente.

Caso o ID de usuário corresponda ao token de acesso, você receberá uma resposta com código [200 \(OK\)](#) contendo os dados do usuário para renderização.

Nesse caso, você poderá renderizar a página corretamente.

Entrega

A entrega deverá ser feita via [GitLab](#). Você deve criar o seu repositório pessoal, sendo que sua entrega consistirá em enviar a URL do seu repositório. A versão considerada será o último *commit* feito antes do horário limite da entrega.

A data limite para entrega é 05/07 (segunda-feira), até às 23h55, com submissão via e-disciplinas do nosso curso.

Avaliação

Desta vez, a avaliação será feita **manualmente**. Isto é, cada repositório será clonado e avaliaremos a sua execução. Verificaremos que as **validações** estão de acordo com a especificação acima. Além disso, inspecionaremos se as **requisições** para trocas de dados entre *back-end* e *front-end* também acontecem conforme especificado.

EXTRA -- Não existe especificação para o visual das páginas. Porém, você é encorajado a utilizar o [TailwindCSS](#) conforme visto em aula para estilizar seus EPs. Para estimulá-los, escolheremos o Top-3 front-ends para ganharem cupons do iFood! Para participar, você deverá enviar um *print-screen* das telas renderizadas no navegador.