

INSTITUTO FEDERAL
SÃO PAULO



Introdução ao Nuxt.js

Desenvolvimento de Sistemas Web (DSWI6)

Prof. Luiz Gustavo Diniz de Oliveira Vêras

E-mail: gustavo_veras@ifsp.edu.br



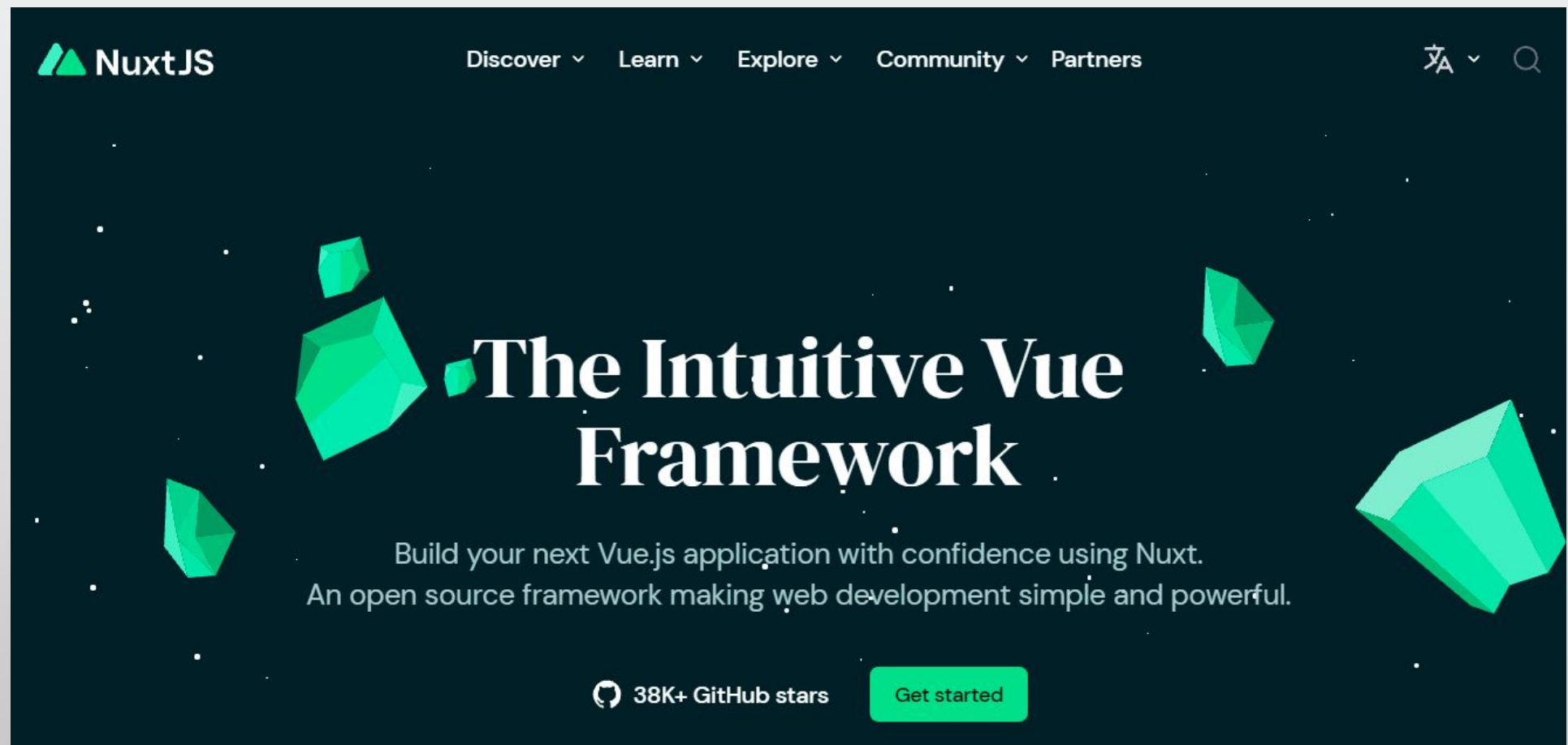
Roteiro da aula

- ✓ **Introdução ao Nuxt.js**
- ✓ **Requisitos**
 - ✓ **Instalando o Node**
 - ✓ **Plugin Vscode**
- ✓ **Criando um projeto Nuxt.js**
- ✓ **Entendendo a estrutura de um projeto Nuxt.js**
- ✓ **Adicionando pages e File system routing**
- ✓ **Roteamento e Páginas dinâmicas**



Nuxt.js

<https://nuxtjs.org/>



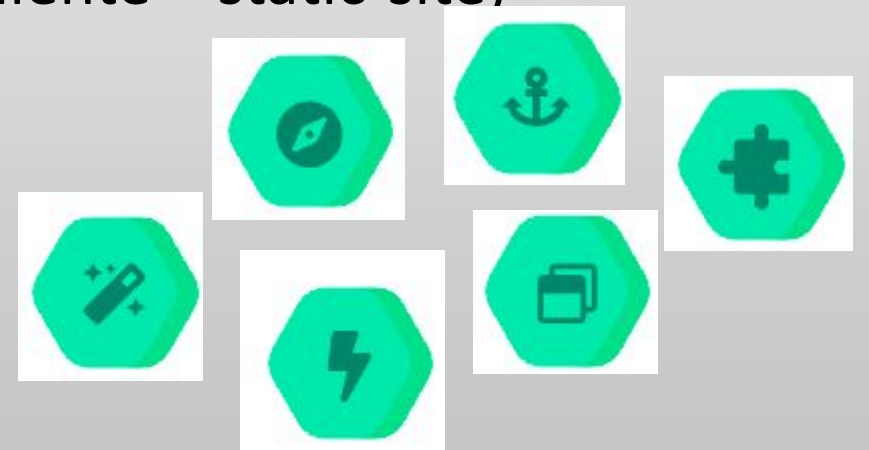


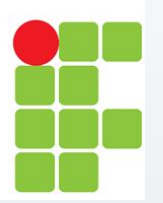
Nuxt.js

O **Nuxt.js** é um framework baseado em **Vue.js** que reúne ferramentas de desenvolvimento web comumente utilizadas, visando o rápido desenvolvimento e suporte ao programador.

Entre muitas das facilidades que ele oferece.

- Rápida configuração de aplicação web
- Roteamento baseado em sistema de arquivos (File-system Routing)
- Modos de renderização (servidor – on-demand; cliente – static-site)
- Auto-importação de componentes Vue.js
- Recursos de busca de dados em APIs
- Disponibilidade de diversos módulos





Requisitos

Para trabalharmos com Nuxt.js, iremos precisar instalar:

- Node.js
- Vscode
- Plugin Vscode para o Vue.js (Buscar por Volar)



Plugins Vscode

- Instale o plugin do Vue.js para suporte na programação com o Vscode.



Vue Language Features (Volar) v0.33.10

Johnson Chu |  440.439 |  (42)

Language support for Vue 3

[Desabilitar](#)  [Desinstalar](#)  [Alternar para a Versão de Pré-Lançamento](#) 

Esta extensão foi habilitada globalmente.



Criando um projeto com Nuxt.js

Abra o Vscode, escolha um diretório, abra o terminal (ctrl + ") e digite o comando abaixo:

```
npm init nuxt-app <project-name>
```

Substitua <Project-name> pelo nome que deseja para o projeto. No nosso exemplo vamos criar um chamado *exemplo-front*.

```
PROBLEMAS  SAÍDA  CONSOLE DE DEPURAÇÃO  TERMINAL
Microsoft Windows [versão 10.0.19041.1288]
(c) Microsoft Corporation. Todos os direitos reservados.

C:\Java Projects (VS)\SpringBoot\aulas_ltpi3_bim4>npm init nuxt-app exemplo-front
```



Criando um projeto com Nuxt.js

Depois de executar esse comando, o programa criador de projetor fará diversas perguntas sobre quais recursos de programação adicionar. Preencha como abaixo:

```
C:\Java Projects (VS)\SpringBoot\aulas_ltpi3_bim4>npm init nuxt-app exemplo-front

create-nuxt-app v3.7.1
✨ Generating Nuxt.js project in exemplo-front
? Project name: exemplo-front
? Programming language: JavaScript
? Package manager: Npm
? UI framework: Bootstrap Vue
? Nuxt.js modules: Axios - Promise based HTTP client
? Linting tools: ESLint, Prettier
? Testing framework: None
? Rendering mode: Universal (SSR / SSG)
? Deployment target: Server (Node.js hosting)
? Development tools: jsconfig.json (Recommended for VS Code if you're not using typescript)
? Continuous integration: None
? Version control system: Git
```




Criando um projeto com Nuxt.js

Aguarde a finalização da criação do projeto e veja a sua estrutura criada no explorador de arquivos do Vscode.

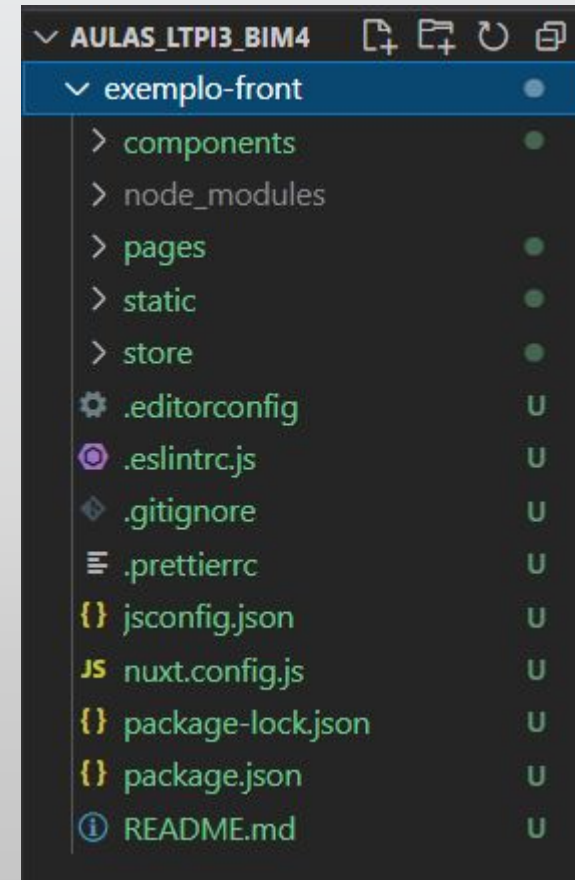
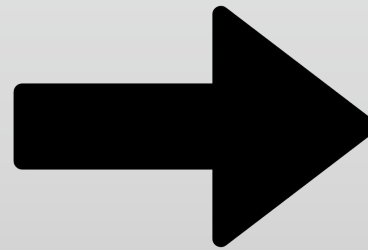
```
🎉 Successfully created project exemplo-front

To get started:

  cd exemplo-front
  npm run dev

To build & start for production:

  cd exemplo-front
  npm run build
  npm run start
```

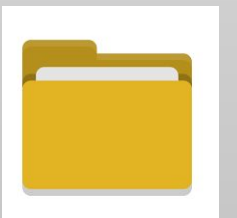


Entendendo a estrutura de um projeto Nuxt.js



Cada pasta de um projeto Nuxt.js habilita uma funcionalidade na aplicação web, por isso é importante entendermos o significado delas.

- **pages:** nelas ficam as *views* e *routes* da sua aplicação. Ex: Se um arquivo de nome `home.vue` está dentro dela, acessamos ela no browser com <http://localhost:3000/home>.
- **components:** É onde colocamos os componentes Vue que serão importados em cada uma das páginas.
- **static:** diretório onde ficam imagens, estilos e fontes que não sofrem alteração.
- **plugins (necessário criar):** onde definimos códigos javascript com injeção de funções e constantes, que são carregadas antes da aplicação Vue.js ser montada.
- **store:** Usado para ativar o armazenamento global de dados com Vuex. Vem desativado. Para ativar, basta criar um arquivo **index.js** vazio dentro da pasta.
- **package.json:** Arquivo com as dependências e scripts do projeto. Semelhantes ao `pom.xml` do Maven.
- **nuxt.config.js:** onde configuramos o Nuxt.js e seus módulos.





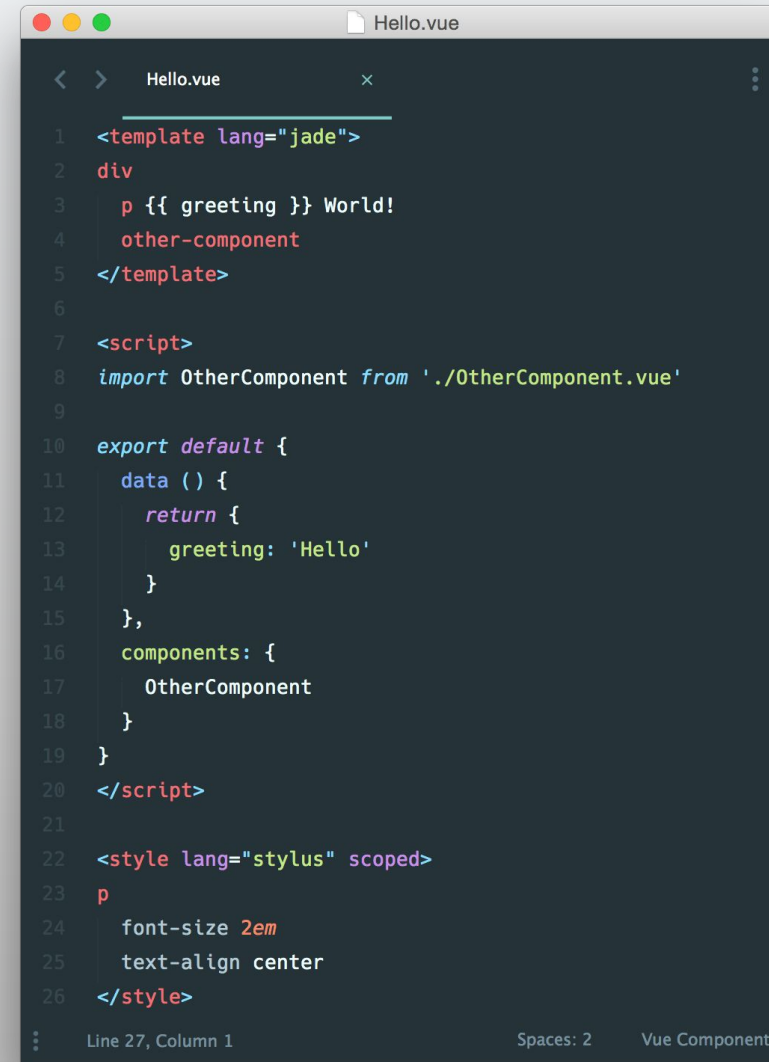
Adicionando pages e File system routing

Vamos criar nossa primeira página. As páginas tem extensão .vue, um formato chamado **Single-File Componente (SFC)** para ser usado em aplicação Vue que usa HTML, Javascript e CSS em um único arquivo. Dividido pelas tags:

`<template></template>`

`<script></script>`

`<style></style>`



```
1 <template lang="jade">
2   div
3     p {{ greeting }} World!
4     other-component
5 </template>
6
7 <script>
8   import OtherComponent from './OtherComponent.vue'
9
10  export default {
11    data () {
12      return {
13        greeting: 'Hello'
14      }
15    },
16    components: {
17      OtherComponent
18    }
19  }
20 </script>
21
22 <style lang="stylus" scoped>
23   p
24     font-size 2em
25     text-align center
26 </style>
```

Line 27, Column 1 Spaces: 2 Vue Component



Adicionando pages e File System Routing

- O projeto já vem com um arquivo chamado **index.vue** dentro de pages. Ela será o ponto de entrada do aplicativo.
- Programamos as páginas da mesma forma que especificamos um objeto Vue.
 - Dentro da tag <script> podemos usar as mesmas propriedades data, methods, mounted ou quaisquer outras funcionalidades do Vue.
 - Dentro da tag <template> podemos usar o padrão {{ data }} para acessar dados e funções definidas no script.
 - Se tiver CSS a ser definido, faça-o na tag <style>. Caso contrário nem precisa criá-la.



Adicionando pages e File System Routing

index.vue

```
1 <template>
2   <div class="container">
3     <h1>{{titulo}}</h1>
4     <ul>
5       <li :key="item.id" v-for = "item in pessoas">
6         {{item.nome}} - Idade: {{item.idade}} anos
7       </li>
8     </ul>
9     <button @click="digaOla">Diga olá!</button>
10  </div>
11 </template>
```

```
34 <style scoped>
35   1 reference
36   .container{
37     margin: 20px
38   }
39 </style>
```

```
13 <script>
14 export default {
15   data: function(){
16     return {
17       titulo: "Lista de pessoas e idades",
18       pessoas: [
19         {id: 1, nome: "Marcio", idade: "33"},
20         {id: 2, nome: "Diego", idade: "44"},
21         {id: 3, nome: "Luana", idade: "50"},
22         {id: 4, nome: "Tamara", idade: "23"},
23       ]
24     }
25   },
26   methods:{
27     digaOla: function(){
28       alert("Olá Mundo do Nuxt.js!");
29     }
30   }
31 }
32 </script>
```




Adicionan

Routing

index.vue

ATENÇÃO:

- Ao usar v-for, a ferramenta exige a indicação de uma key, que é o id do item no v-for.
- :key é abreviação para v-bind:key

```
1 <template>
2   <div class="container">
3     <h1>{{titulo}}</h1>
4     <ul>
5       <li :key="item.id" v-for = "item in pessoas">
6         {{item.nome}} - Idade: {{item.idade}} anos
7       </li>
8     </ul>
9     <button @click="digaOla">Diga olá!</button>
10  </div>
11 </template>
```

ATENÇÃO:

- Você pode definir funções javascript na propriedade methods e chama-las a qualquer momento na programação do view.

```
12 <script>
13   {id: 1, nome: "Diego", idade: "44"},
14   {id: 2, nome: "Diego", idade: "44"},
15   {id: 3, nome: "Luana", idade: "50"},
16   {id: 4, nome: "Tamara", idade: "23"},
17 ]
18 },
19 ],
20 {id: 2, nome: "Diego", idade: "44"},
21 {id: 3, nome: "Luana", idade: "50"},
22 {id: 4, nome: "Tamara", idade: "23"},
23 ],
24 },
25 },
26 methods: {
27   digaOla: function() {
28     alert("Olá Mundo do Nuxt.js!");
29   }
30 },
31 },
32 </script>
```

ATENÇÃO:

- @click é abreviação de v-on:click

```
34
35
36
37
38 </style>
```



Adicionando pages e File System Routing

- Para executar o projeto, execute **npm run dev** na linha de comando. Certifique-se de estar no diretório do projeto.

```
C:\Java Projects (VS)\SpringBoot\aulas_ltpi3_bim4\exemplo-front>npm run dev  
  
> exemplo-front@1.0.0 dev C:\Java Projects (VS)\SpringBoot\aulas_ltpi3_bim4\exemplo-front  
> nuxt
```

Nuxt @ v2.15.8

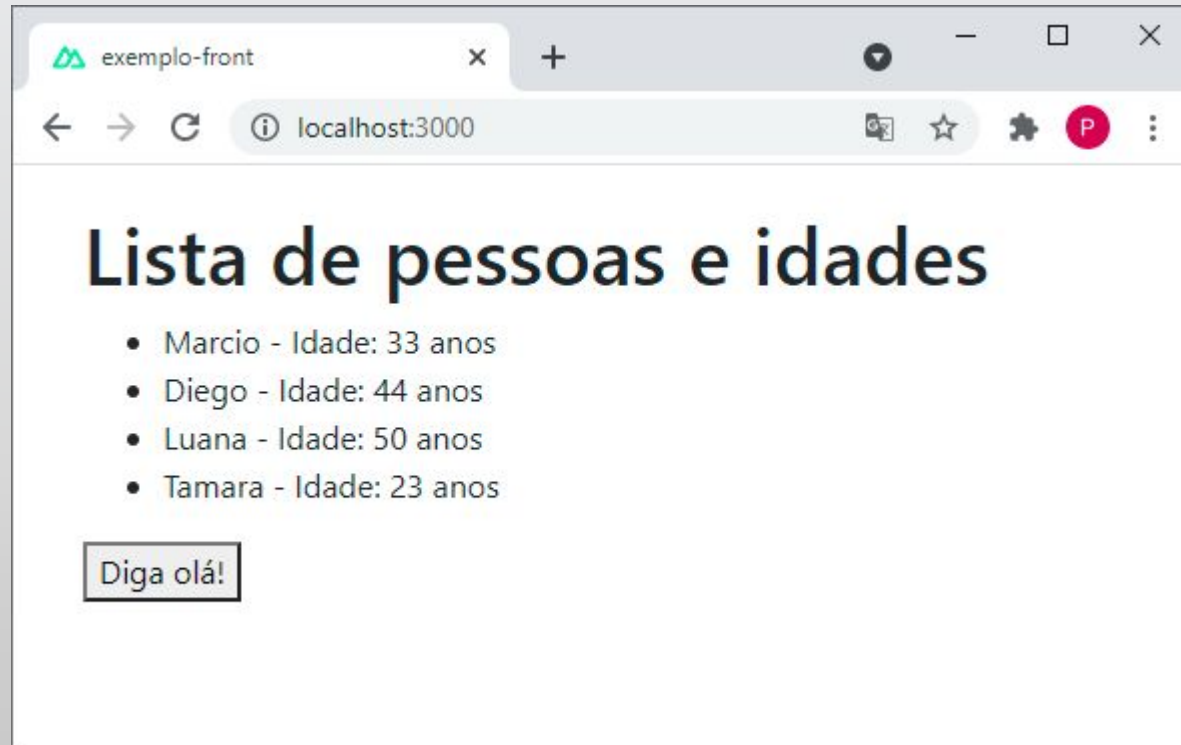
‣ **Environment:** development
‣ **Rendering:** server-side
‣ **Target:** server

Listening: <http://localhost:3000/>



Adicionando pages e File System Routing

- Digite <http://localhost:3000> no browser para acessar a página.



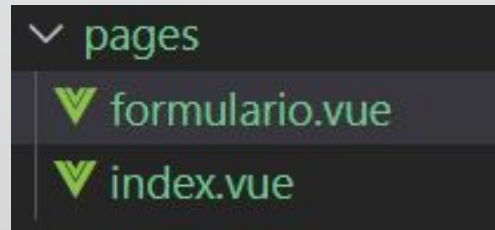


Adicionando pages e File System Routing

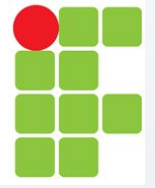
Podemos criar mais páginas utilizando o File System Routing.

Se criarmos um arquivo `.vue`, o nome dele será o path na url para acessá-lo.

- Crie o arquivo de nome **formulário.vue** dentro da pasta `pages`.



- Insira o código descrito a seguir e salve as alterações.



Adicionando pages e File System Routing

formulario.vue

```
1  <template>
2    <div class="container">
3      <h3>Insira seus dados</h3>
4      <form @submit.prevent="digaOla">
5        <span ref="nome">Nome: </span>
6        <input id="nome" v-model="form.nome" type="text" /> <br><br>
7        <span ref="email">Email: </span>
8        <input id="email" v-model="form.email" type="email" /> <br><br>
9        <button type="submit">Ok</button>
10     </form>
11
12     <div v-show="form.nome">
13       <h4 v-if = "form.nome && form.email">
14         Seus já podem ser enviados!
15       </h4>
16       <span v-else style="color: ■ red;">Dados incompletos!</span>
17     </div>
18   </div>
19 </template>
```



Adicionando pages e File System

formulario.vue

```
1 <template>
2   <div class="container">
3     <h3>Insira seus dados</h3>
4     <form @submit.prevent="liga01a">
5       <span ref="nome">Nome: </span>
6       <input type="text" v-model="form.nome" type="text" /> <br><br>
```

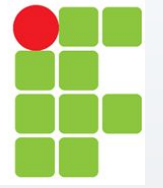
ATENÇÃO:

- Se v-show receber um valor de form.nome, o seu conteúdo será mostrado na tela.

ATENÇÃO:

- V-if e v-else funcionam em par. Se a primeira for falsa, o segundo elemento será mostrado.

```
11   <div v-show="form.nome">
12     <h4 v-if = "form.nome && form.email">
13       Seus já podem ser enviados!
14     </h4>
15     <span v-else style="color: red;">Dados incompletos!</span>
16   </div>
17 </div>
18 </div>
19 </template>
```



Adicionando pages e File System Routing

formulario.vue

```
21 <script>
22 export default {
23   data: function () {
24     return {
25       form: {
26         nome: '',
27         email: '',
28       },
29     }
30   },
31   methods: {
32     digaOla: function () {
33       alert(JSON.stringify(this.form))
34     },
35   },
36 }
37 </script>
```

```
34 <style scoped>
    1 reference
35   .container{
36     margin: 20px
37   }
38 </style>
```



Adicionando pages e File System Routing

- Digite <http://localhost:3000/formulario> no browser para acessar a página.

The screenshot shows a web browser window with the address bar displaying `localhost:3000/formulario`. The page content includes a form titled "Insira seus dados" with two input fields: "Nome:" containing "Marcelos marcos" and "Email:" containing "marcos@ifsp.edu.br". Below the form is a button labeled "Ok". A message "Seus dados foram enviados!" is displayed at the bottom. A modal dialog box is open in the center, showing the text "localhost:3000 diz" and a JSON object `{"nome": "Marcelos marcos", "email": "marcos@ifsp.edu.br"}`, with an "OK" button.



Roteamento e Páginas dinâmicas

O Nuxt.js já vem acoplado com um sistema de roteamento, que permite navegar de uma página para a outra, além de passar parâmetros para elas. Os módulos que permite o roteamento são:

vue-route: módulo de roteamento oficial do Vue.js, que é utilizado pelo Nuxt.js

NuxtLink: Componente fornecido pelo Nuxt.js para realizar roteamento.

```
<template>
  <NuxtLink to="/">Home page</NuxtLink>
</template>
```



Roteamento e Páginas dinâmicas

O roteamento também é bastante útil com o uso de **páginas dinâmicas**. Essa funcionalidade nos permite definir o caminho das páginas baseado nos dados passados durante o roteamento.

Para utilizar o páginas dinâmicas, basta iniciar o nome de cada arquivo de página com sobrescrito (_). Podemos utilizar nomes de pastas como o mesmo esquema para definir subníveis.

Ex:

_dados.vue

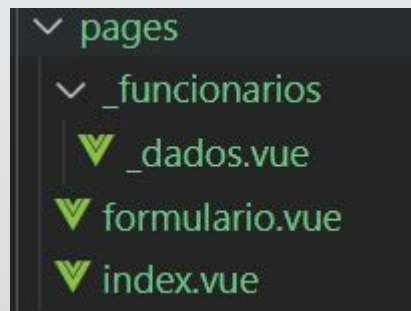
_funcionario/_dados.vue



Roteamento e Páginas dinâmicas

Crie o seguinte esquema de diretório dentro da pasta **pages**.

`_funcionário/_dados.vue`



- Na página de formulário, ao invés do botão use um **NuxtLink**.



Roteamento e Páginas dinâmicas

formulario.vue

```
1  <template>
2    <div class="container">
3      <h3>Insira seus dados</h3>
4      <form @submit.prevent="digaOla">
5        <span ref="nome">Nome: </span>
6        <input id="nome" v-model="form.nome" type="text" /> <br><br>
7        <span ref="email">Email: </span>
8        <input id="email" v-model="form.email" type="email" /> <br><br>
9        <!-- <button type="submit">Ok</button> -->
10     </form>
11
12     <div v-show="form.nome">
13       <h4 v-if = "form.nome && form.email">
14         <NuxtLink :to="`/${form.nome}/${form.email}`">Seus já podem ser enviados!</NuxtLink>
15       </h4>
16       <span v-else style="color: ■ red;">Dados incompletos!</span>
17     </div>
18   </div>
19 </template>
```



Roteamento e Páginas dinâmicas

formulario.vue

ATENÇÃO:

- Os valores recebidos pelo **to** são rotas relacionadas a aplicação. Se uma rota não existir a aplicação irá gerar um erro.

ATENÇÃO:

- Aqui é usada a funcionalidade do javascript chamada String template, que mistura texto com código javascript. Tudo que estiver dentro de `${}` será executado como javascript e devolvido como texto.

```
7      <span ref="email">Email: </span>
8      <input id="email" v-model="form.email" type="text" />
9      <!-- <button type="submit">Ok</button> -->
10     </form>
11
12     <div v-show="form.nome">
13       <h4 v-if = "form.nome & form.email">
14         <NuxtLink :to="`/${form.nome}/${form.email}`">Seus já podem ser enviados!</NuxtLink>
15       </h4>
16       <span v-else style="color: red;">Dados incompletos!</span>
17     </div>
18   </div>
19 </template>
```



Roteamento e Páginas dinâmicas

`_funcionarios/_dados.vue`

```
1 <template>
2   <div class="container">
3     <h1>Dados do funcionário</h1>
4     <h4>Nome: {{ funcionario.nome }}</h4>
5     <h4>Email: {{ funcionario.email }}</h4>
6   </div>
7 </template>
```

```
22 <style scoped>
    1 reference
23   .container {
24     margin: 20px;
25   }
26 </style>
```

```
9 <script>
10 export default {
11   asyncData({ params }) {
12     const funcionario = {
13       nome: params.funcionarios, // recebe o valor do primeiro nível do path do link
14       email: params.dados // recebe o valor do segundo nível do path do link
15     };
16     console.log(funcionario); // Para Debug
17     return { funcionario };
18   },
19 }
20 </script>
```



Roteamento e Páginas dinâmicas

`_funcionarios/_dados.vue`

ATENÇÃO:

- `asyncData` é uma função padrão executada pelo Nuxt.js antes de gerar a página Vue. Será bastante usada para carregar plugins, dados de APIs e, como no exemplo abaixo, carregar parâmetros vindos pelo router.

```
10 export default {
11   asyncData({ params }) {
12     const funcionario = {
13       nome: params.funcionarios, // recebe
14       email: params.dados // recebe o valor
15     };
16     console.log(funcionario); // Para Debug
17     return { funcionario };
18   },
19 }
20 </script>
```

ATENÇÃO:

- Os nomes das propriedades são os nomes das pastas e arquivos dinâmicos. Definimos uma pasta de nome `_funcionarios` e um arquivo de nome `_dados.vue` dentro dela. Logo as propriedades dos parâmetros serão `params.funcionarios` e `params.dados`.
- Os serão recebidos na ordem que estiverem especificados no `:to` do `NuxtLink`.
- Ex: `/ifsp/ifsp@edu.com.br` irá resultar no objeto `"params: { funcionarios: ifsp, dados: ifsp@edu.com.br}"`



Roteamento e Páginas dinâmicas

`_funcionarios/_dados.vue`

```
1 <template>
2   <div class="container">
3     <h1>Dados do funcionário</h1>
4     <h4>Nome: {{ funcionario.nome }}</h4>
5     <h4>Email: {{ funcionario.email }}</h4>
6   </div>
7 </template>
```

```
22 <style scoped>
    1 reference
23   .container {
24     margin: 20px;
25   }
26 </style>
```

```
9 <script>
10 export default {
11   asyncData({ params }) {
12     const funcionario = {
13       nome: params.funcionarios, // recebe o valor
14       email: params.dados // recebe o valor
15     };
16     console.log(funcionario); // Para Debug
17     return { funcionario };
18   },
19 }
20 </script>
```

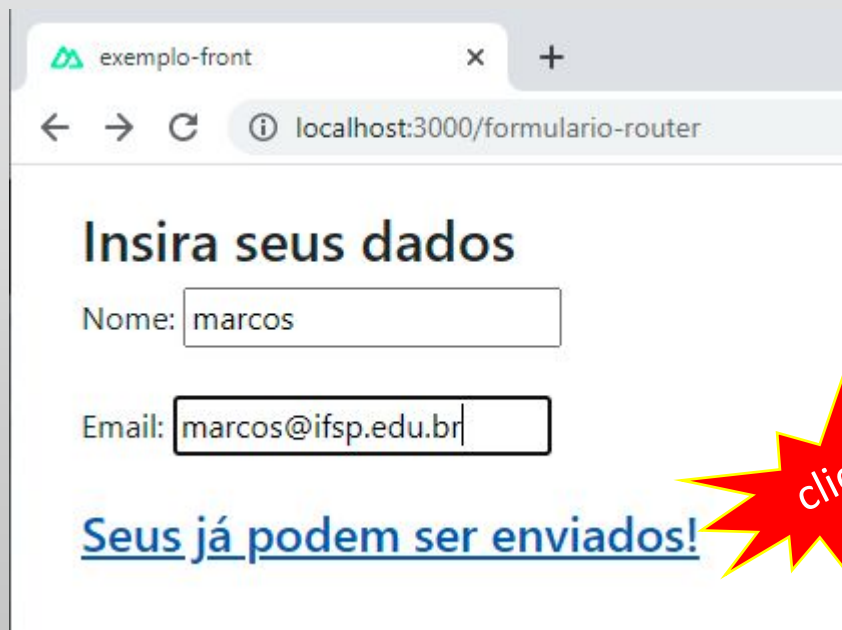
ATENÇÃO:

- Os dados retornados pela função `asyncData` são adicionados ao data do componente vue criado nessa page. Atenção ao uso de `{}`. Caso contrário os dados não serão incorporados ao estado do componente Vue.



Roteamento e Páginas dinâmicas

Agora o texto “Seus dados já podem ser enviados” é um link. Clicando nele podemos enviar os dados para a página dinâmica que tem o padrão **_funcionarios/_dados**. Na URL, o caminho é substituído pelos dados passado na propriedade **:to** do NuxtLink.



exemplo-front x +

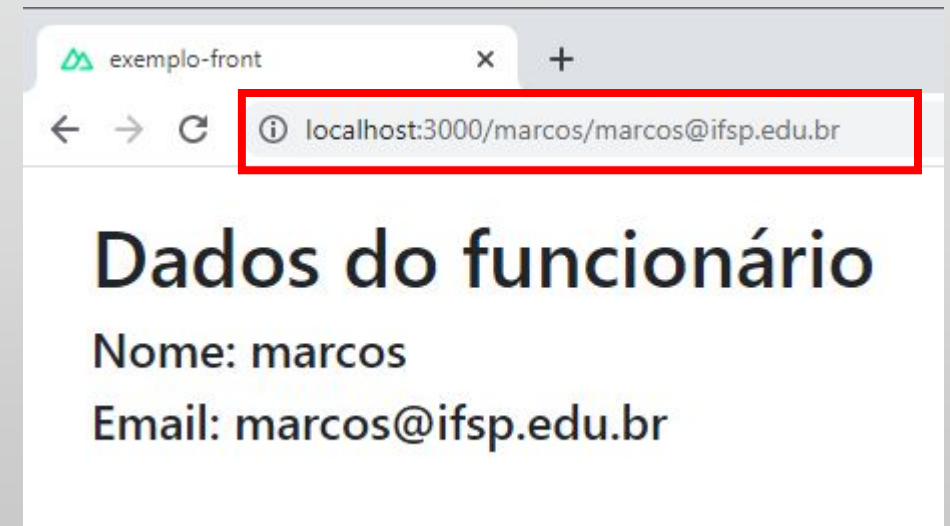
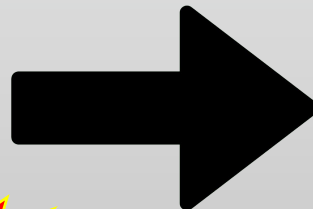
localhost:3000/formulario-router

Insira seus dados

Nome:

Email:

[Seus dados já podem ser enviados!](#)



exemplo-front x +

localhost:3000/marcos/marcos@ifsp.edu.br

Dados do funcionário

Nome: marcos

Email: marcos@ifsp.edu.br





Algumas dicas

- Se ao criar um novo arquivo .vue no projeto Nuxt.js e ela não for carregada de imediato, pare o servidor Nuxt.js pressionando **ctrl+C** com o terminal selecionado e execute novamente **npm run dev**.
- Para ver o que está acontecendo dentro dos métodos, imprima os dados com `console.log()` e veja a saída no **inspetor de elemento** do seu browser, na aba console. Se tiver interesse, pode usar a ferramenta de Debug. Ver como configurar em <https://medium.com/js-dojo/debugging-nuxt-js-with-vs-code-60a1a9e75cf6>



Algumas dicas

- Para não repetir o mesmo estilo css em cada arquivo, você pode criar a pasta **layouts** e criar o arquivo **default.vue**. Não esqueça de registrá-lo no arquivo de configuração do nuxt.



nuxt.config.js

```
18
19 // Global CSS: https://go.nuxtjs.dev/config-css
20 css: [
21   '@/assets/css/style.css'
22 ],
23
```




Links e documentação do Nuxt.js

Documentação (necessário traduzir)

- <https://nuxtjs.org/docs/get-started/installation>

Estrutura de diretórios (necessário traduzir)

- <https://nuxtjs.org/docs/get-started/directory-structure>

NuxtLink

- <https://nuxtjs.org/docs/features/data-fetching/>

AsyncData

- <https://nuxtjs.org/docs/features/data-fetching/>

Exemplos

- <https://nuxtjs.org/examples/routing/hello-world>

Configuração do Debug

- <https://medium.com/js-dojo/debugging-nuxt-js-with-vs-code-60a1a9e75cf6>