

# Treinamento Teknisa

Desenvolvendo projeto com VUE

Gustavo Silva

- I. Requisitos**
- II. Comandos iniciais**
- III. Configuração**
- IV. Estrutura do Projeto**
- V. Bibliotecas necessárias**
- VI. Desenvolvendo**

# I. Requisitos

# O que é necessário?

Para inicializar o projeto temos alguns pré-requisitos.

- Visual Studio Code
- Node.js
- npm ou Yarn
- Vue CLI

**PRÉ-REQUISITOS**



## II. Comandos de inicialização

# Preparando o ambiente

Antes de iniciarmos o projeto precisaremos instalar dependências para o npm, precisaremos dos seguintes comandos.

```
npm install -g @vue/cli  
npm i -g @vue/cli-init
```

No Visual Studio podemos instalar algumas extensões para ajudar no entendimento do código.

Vetur por Pine Wu  
Vue por jcbuisson

# Comandos

Comando para inicializar o projeto com Vue CLI  
"vue init webpack nomeprojeto"

caso seja apresentado algum erro, "npm i -g npm-upgrade"  
pode ajudar.

```
? Project name (teknisa) █
```

tecle enter ou altere o nome

```
Project description (A Vue.js project)
```

adicione uma descrição

```
Author (Gustavo Oliveira <gustavo.silva@teknisa.com>)
```

O projeto reconhece as credenciais do git, altere se desejar.

```
? Vue build (Use arrow keys)
> Runtime + Compiler: recommended for most users
  Runtime-only: about 6KB lighter min+gzip, but templates (or any Vue-specific
  HTML) are ONLY allowed in .vue files - render functions are required elsewhere
```

Selecione a primeira opção, "Runtime + Compiler"

# Comandos

```
? Vue build standalone
? Install vue-router? (Y/n) Y
```

mapear as rotas.

“ Y ” para instalar o router, responsável por

```
? Use ESLint to lint your code? (Y/n) Y
```

irá nos informar sobre erros e inconsistências no código em tempo real.

“ Y ” para adicionar ESLint, que

```
? Pick an ESLint preset (Use arrow keys)
> Standard (https://github.com/standard/standard)
  Airbnb (https://github.com/airbnb/javascript)
  none (configure it yourself)
```

Selecionaremos a primeira opção de preset para o ESLint, “ Standard ”.

```
? Set up unit tests (Y/n) n
```

“ n ” para os teste unitários.

```
? Setup e2e tests with Nightwatch? No
```

“ n ” para os teste automatizados.

```
? Should we run `npm install` for you after the project has been created?
(recommended) (Use arrow keys)
> Yes, use NPM
  Yes, use Yarn
  No, I will handle that myself
```

Deixe a opção “ Yes, use NPM ” para o “ npm install ” ser executado automaticamente ao terminarmos de inicializar a aplicação.

# Comandos

```
# Project initialization finished!  
# =====  
  
To get started:  
  
  cd teknisa  
  npm run dev  
  
Documentation can be found at https://vuejs-templates.github.io/webpack
```

Mensagem de sucesso para inicialização do projeto.

Na pasta criada para o projeto, execute “ npm run dev ” para compilar o projeto.

```
DONE Compiled successfully in 9022ms  
  
I Your application is running here: http://localhost:8080
```

Projeto já pode ser acessado no endereço informado pelo compilador.



# III. Configuração

## Personalizando as opções

Na raiz do projeto, o arquivo “ .eslintrc.js ” é responsável pela configuração do ESLint que é analisa o código em busca de erros e inconsistências, vamos adicionar em “rules” a opção:

```
  'indent': 'off',
```

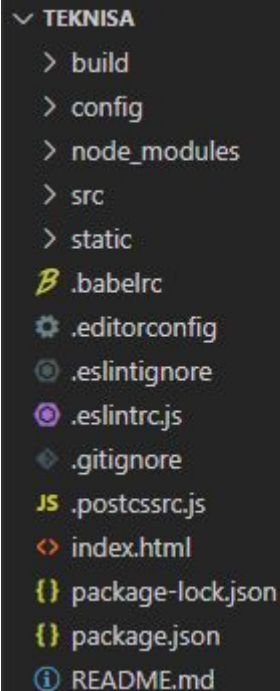
para desabilitar a opção de análise de indentação.

Ainda na raiz, no arquivo “ .editorconfig ” é possível fazer alterações nos padrões da aplicação, como charset, tipo de indentação, etc.

No readme é possível visualizar os comandos “npm” que ficam na aplicação por padrão.

# IV. Estrutura do Projeto

# Pastas

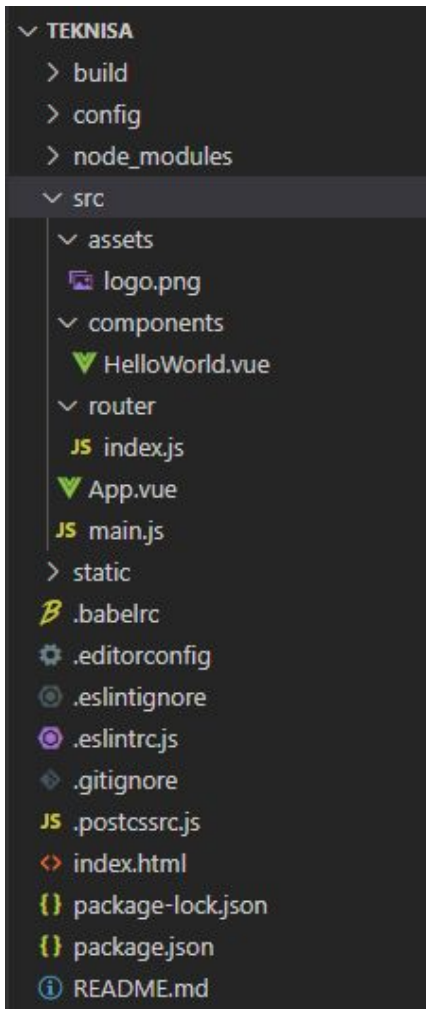


```
▼ TEKNISA
  > build
  > config
  > node_modules
  > src
  > static
  B .babelrc
  ⚙ .editorconfig
  ⦿ .eslintignore
  ⦿ .eslintrc.js
  ⦿ .gitignore
  JS .postcssrc.js
  <> index.html
  {} package-lock.json
  {} package.json
  ⓘ README.md
```

Após a inicialização do projeto vamos nos deparar com a seguinte estrutura de pastas e arquivos.

Na pasta build é onde ficam os arquivos compilados, em config ficam arquivos para configuração em produção e desenvolvimento, node\_modules são as dependências do projeto, src é onde nosso sistema será desenvolvido, static ficarão nossos assets.

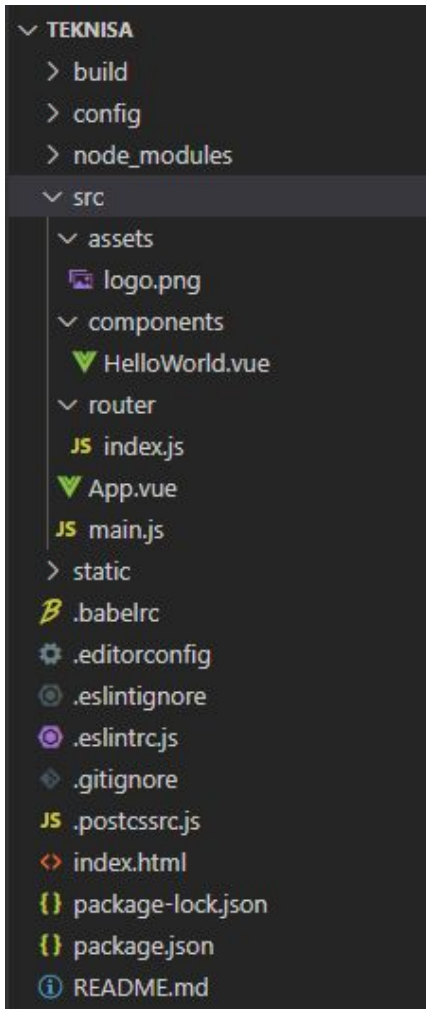
# Pastas



Dentro pasta “src/components” é onde colocaremos as partes do nosso sistema que funcionará de maneira modular.

Já na pasta “src/router” com arquivo “index.js” é onde configuramos as urls da nossa aplicação.

# Arquivos



Em “src/components” ficaram nossos arquivos com extensão “.vue”, por padrão temos o arquivo “HelloWorld.vue” que é a tela que aparece ao acessar o endereço web da nossa aplicação.

No arquivo “src/router/index.js” ficam as rotas da url.

“App.vue” é o index do nosso sistema, onde chamaremos nossos components, bibliotecas, etc.

“main.js” é onde importaremos as bibliotecas que usaremos em todos os components.

# V. Bibliotecas necessárias

# Dependências do projeto

“vue-axios” será usado para comunicarmos com a api .

```
npm install vue-axios
```

```
npm install --save axios
```

“vue-js-modal” será usado para apresentarmos o modal dos cards.

```
npm install vue-js-modal
```

```
npm install --save vue-js-modal
```



# V. Desenvolvendo

# Codando

Começaremos criando um novo componente ou alterando o existente, HelloWorld.

```
JS index.js  X
src > router > JS index.js > [?] default
1  import Vue from 'vue'
2  import Router from 'vue-router'
3  import HelloWorld from '@components/HelloWorld'
4
5  Vue.use(Router)
6
7  export default new Router({
8    routes: [
9      {
10       path: '/',
11       name: 'HelloWorld',
12       component: HelloWorld
13     }
14   ]
15 })
16
```

# Codando

▼ App.vue ×

src > ▼ App.vue > {} "App.vue"

```
1 <template>
2   <div id="app">
3     <router-view/>
4   </div>
5 </template>
6
7 <script>
8   export default {
9     name: 'App'
10  }
11 </script>
12
13 <style>
14 #app {
15   font-family: 'Avenir', Helvetica, Arial, sans-serif;
16   -webkit-font-smoothing: antialiased;
17   -moz-osx-font-smoothing: grayscale;
18   text-align: center;
19   color: #2c3e50;
20   margin-top: 60px;
21 }
22 </style>
```

# Codando

O endereço da api que será usado na aplicação é:

<https://bernardosantos.zeedhi.com/workfolder/dev.php>

O exemplo que seguiremos para nossa aplicação será o seguinte:

<https://www.figma.com/file/h4z36ouFG2dR0TIIBI3S3t/Treinamento?node-id=0%3A1>

# Codando

O projeto feito durante o treinamento ficará disponível no github no seguinte endereço:

[gustavo491/listagem-vue](#)

<https://github.com/gustavo491/listagem-vue>