



Aplicação com Node.js e Vue.js/Quasar

Gestão de clientes (CRUD)

LoopBack

What is LoopBack?

LoopBack is a highly-extensible, open-source Node.js framework that enables you to:

- ▶ Create dynamic end-to-end REST APIs with little or no coding.
- ▶ Access data from major relational databases, MongoDB, SOAP and REST APIs.
- ▶ Incorporate model relationships and access controls for complex APIs.
- ▶ Separable components for file storage, third-party login, and OAuth 2.0.

Quasar

What is Quasar?

Quasar (pronounced /'kweɪ.zɑːr/) is an MIT licensed open-source framework (powered with Vue) that helps web developers create responsive++ websites/apps in many flavours:

- ▶ SPAs (Single Page App)
- ▶ SSR (Server-side Rendered App) (+ optional PWA client takeover)
- ▶ PWAs (Progressive Web App)
- ▶ Mobile Apps (Android, iOS, ...) through Apache Cordova
- ▶ Multi-platform Desktop Apps (using Electron)

Vue.js

O que é Vue.js?

Vue (pronuncia-se view, como em inglês) é um framework progressivo para a construção de interfaces de usuário. A biblioteca principal é focada exclusivamente na camada visual (*view layer*), sendo fácil adotar e integrar com outras bibliotecas ou projetos existentes.

Instalação dos pacotes/frameworks

NPM

- ▶ `npm install -g loopback-cli --no-optional vue-cli quasar-cli`
- ▶ Opcional para monitorar alterações no código fonte e automaticamente reiniciar o servidor: `npm install -g nodemon`

Criação do banco de dados

MySQL

Acessar no CMD o caminho `C:\Program Files\MySQL\MySQL Server 8.0\bin`, para executar o MySQL e criar o banco de dados:

- ▶ Acesso: `mysql -u root -p1@asdfg`
- ▶ Comando: `CREATE DATABASE cliente;`

Criação da API com LoopBack

Acessar ou criar a pasta onde ficará o projeto e executar o comando “lb” para iniciar o prompt com as opções disponíveis para a criação do projeto (nome do aplicativo, versão do LoopBack e tipo de API), conforme a imagem a seguir:

CA Prompt de Comando

```
C:\Dev\aula_dgaw>lb
? Qual o nome do seu aplicativo? aula_dgaw
? Qual versão de LoopBack você gostaria de usar? 3.x (Active Long Term Support)
? Que tipo de aplicativo você tem em mente? api-server (Um servidor de API LoopBack com autenticação do usuário local)
Gerando .yo-rc.json

I'm all done. Running npm install for you to install the required dependencies. If this fails, try running the command yourself.

Próximas etapas:

  Crie um modelo em seu aplicativo
    $ lb model

  Execute o aplicativo
    $ node .

A equipe do API Connect na IBM continua a desenvolver,
suportar e manter LoopBack, que está no núcleo do
API Connect. Quando suas APIs precisam de gerenciamento robusto e
opções de segurança, veja http://ibm.biz/tryAPIC

npm notice created a lockfile as package-lock.json. You should commit this file.
added 387 packages from 427 contributors in 36.417s
```

Criação da datasource

Após a criação da API, vamos criar a configuração da conexão com o banco de dados (datasource) criado anteriormente, executando o comando “lb datasource”, informando:

- ▶ Nome da datasource (cliente);
- ▶ Tipo de conector;
- ▶ IP do hospedeiro;
- ▶ Porta (conforme tipo de conector);
- ▶ Usuário que vai acessar o banco de dados;
- ▶ Senha de acesso;
- ▶ Nome do banco de dados;
- ▶ E se deseja instalar o conector do tipo selecionado para o LoopBack.

```
CA: Prompt de Comando
C:\Dev\aula_dgaw>lb datasource
? Insira o nome da origem de dados: clientes
? Seleccione o conector para clientes: MySQL (suportado por StrongLoop)
? Connection String url to override other settings (eg: mysql://user:pass@host/db):
? host: 127.0.0.1
? port: 3306
? user: jairo
? password: [hidden]
? database: clientes
? Instalar loopback-connector-mysql@5.3.0 Yes
npm WARN registry Using stale data from https://registry.npmjs.org/ because the host
npm WARN registry Using stale data from https://registry.npmjs.org/ due to a request
+ loopback-connector-mysql@5.3.1
added 6 packages from 14 contributors and audited 2484 packages in 21.005s
found 0 vulnerabilities
```


Criação da model

Após a criação da datasource, vamos criar uma model chamada **cliente**, executando o comando “**lb model**”, informando:

- ▶ Nome da model;
- ▶ Selecionar a datasource (cliente) criada anteriormente;
- ▶ Selecionar a classe base para a model (neste caso PersistedModel);
- ▶ Selecionar se deseja exibir esta model na interface da API;
- ▶ Nome no plural para a model (opcional);
- ▶ Tipo de model;
- ▶ Incluir as propriedades de model:
 - ▶ Nome;
 - ▶ Tipo;
 - ▶ Se é obrigatório;
 - ▶ Valor padrão (opcional).

Git Prompt de Comando

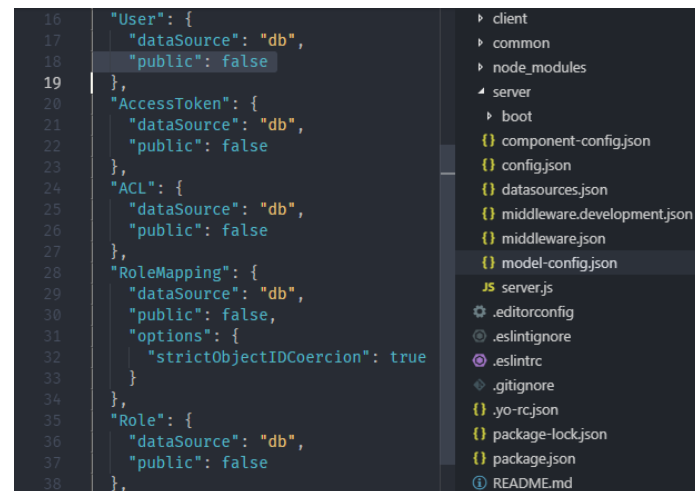
```
C:\Dev\aula_dgaw>lb model
? Insira o nome do modelo: cliente
? Seleccione a origem de dados para a qual conectar o cliente: clientes (mysql)
? Seleccione a classe base do modelo PersistedModel
? Expor cliente por meio da API REST? Yes
? Formulário plural customizado (usado para construir REST URL): clientes
? Modelo comum ou apenas servidor? comum
Vamos incluir algumas propriedades cliente agora.

Insira um nome de propriedade em branco ao concluir.
? Nome da Propriedade: nome
? Tipo de propriedade: string
? Necessário? Yes
? Valor padrão [deixe em branco para nenhum]:
```

Configurações adicionais

Após a criação da model, vamos efetuar mais algumas configurações no nosso código fonte:

- ▶ Por padrão, quando criamos uma model usando a classe base “PersistedModel”, o LoopBack exhibe na interface da API a model “User”. Como neste projeto não vamos usar esta model, vamos ocultá-la da API:
 - ▶ Na pasta do projeto, acesse “server/model-config.json” e na linha 16, onde temos a model “User”, vamos adicionar a propriedade “public”: false.
- ▶ Logo após, vamos utilizar a datasource criada nas nossas models:
 - ▶ No mesmo arquivo “model-config.json”, vamos editar o nome da datasource em cada model, mudando de “db” para “cliente”.



```
16  "User": {
17    "dataSource": "db",
18    "public": false
19  },
20  "AccessToken": {
21    "dataSource": "db",
22    "public": false
23  },
24  "ACL": {
25    "dataSource": "db",
26    "public": false
27  },
28  "RoleMapping": {
29    "dataSource": "db",
30    "public": false,
31    "options": {
32      "strictObjectIDCoercion": true
33    }
34  },
35  "Role": {
36    "dataSource": "db",
37    "public": false
38  },
```

Configurações adicionais

- ▶ Para facilitar, vamos criar um script para execução da API:
 - ▶ Na pasta do projeto, edite o arquivo “package.json”, adicionando na linha 8, em “scripts”, o seguinte código: “dev”: “nodemon .”, caso você tenha instalado este pacote nas etapas anteriores.

```
8   "scripts": {  
9     "lint": "eslint .",  
10    "start": "node .",  
11    "posttest": "npm run lint",  
12    "dev": "nodemon ."  
13  },
```

Configurações adicionais

- ▶ Para facilitar nossa vida, vamos criar um script para criar nossas tabelas automaticamente, utilizando as configurações das nossas models.
- ▶ Crie o arquivo `server/boot/00-migration.js`, com o seguinte código:

```
'use strict';

module.exports = function(app, cb) {
  let lbTables = {
    db: [
      'ACL',
      'RoleMapping',
      'Role',
      'AccessToken',
      'cliente',
    ],
  };

  console.log('\x1b[35m\n%s\n\x1b[0m', 'Atualizando o banco de dados...');

  app.dataSources.cliente.autoupdate(lbTables.db, err => {
    if (err) throw err;
  });

  process.nextTick(cb);
};
```

Teste da API

Agora vamos testar nossa API, executando o comando “`npm run dev`” no cmd.

Se estiver tudo funcionando, no console vai aparecer as mensagens:

```
Web server listening at: http://localhost:3000  
Browse your REST API at http://localhost:3000/explorer
```

Indicando que o servidor está aguardando requisições na porta 3000 e que você pode acessar a interface da API em `http://localhost:3000/explorer`.

Criação da aplicação com Quasar

Acessar a pasta onde ficará a aplicação e executar o comando “`quasar init nome_do_projeto`” no cmd, para iniciar o prompt com as opções disponíveis para a criação da aplicação:

- ▶ Nome do projeto;
- ▶ Nome do produto;
- ▶ Descrição do projeto (opcional);
- ▶ Autor;
- ▶ Plugins;
- ▶ Padrão do Linter;
- ▶ Id do Cordova (opcional);
- ▶ Gerenciador de pacotes para o projeto.

```
Windows PowerShell
PS C:\Users\jairo.ortiz.CETIL\Downloads> quasar init TESTE
Running command: vue init 'quasarframework/quasar-starter-kit' TESTE

? Project name (internal usage for dev) teste
? Project product name (official name) Quasar App
? Project description A Quasar Framework app
? Author Jairo Ortiz <jairo.ortiz@govbr.com.br>
? Check the features needed for your project: ESLint, Axios
? Pick an ESLint preset Standard
? Cordova id (disregard if not building mobile apps) org.cordova.quasar.app
? Should we run `npm install` for you after the project has been created? (recommended) NPM

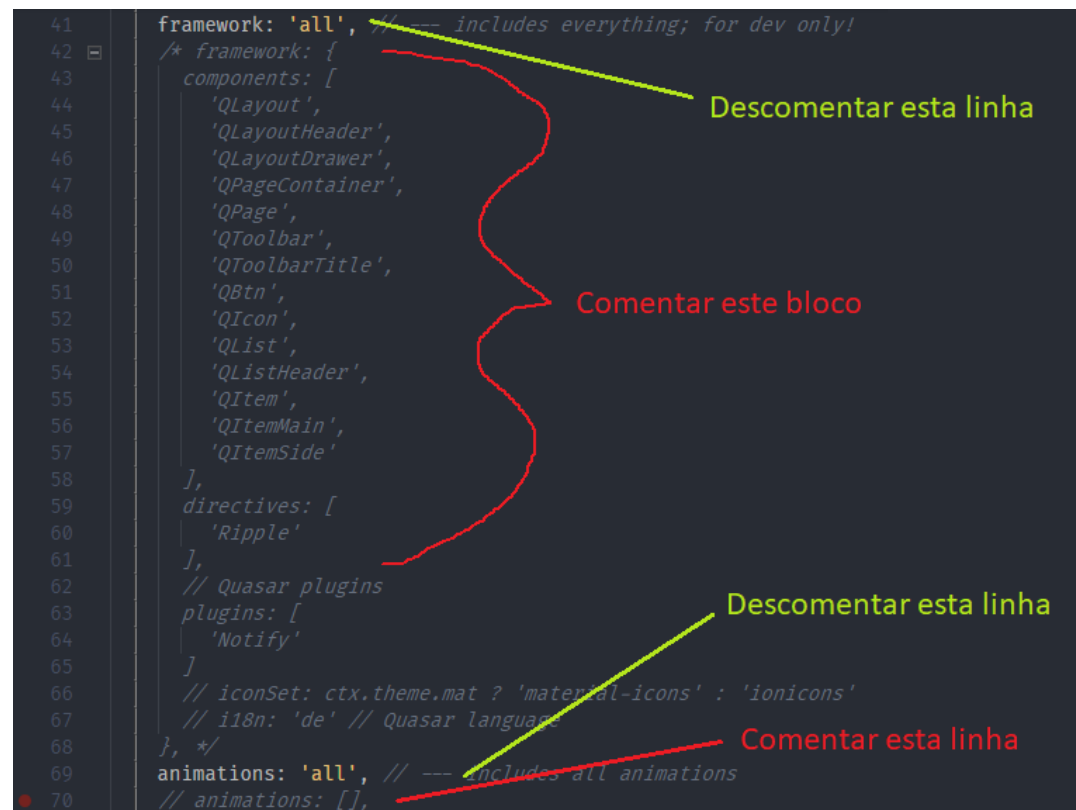
vue-cli - Generated "TESTE".

[*] Installing project dependencies ...
```

Configuração inicial

Primeiramente, como este é um projeto de testes, vamos configurar o framework para utilizar todos os componentes e animações disponíveis, para evitar ter que adicionar um a um quando for necessário.

Para isso, vamos editar o arquivo “[quasar.conf.js](#)”, conforme a imagem a seguir:



The image shows a code editor with the `quasar.conf.js` file. The code is as follows:

```
41 framework: 'all', // -- includes everything; for dev only!
42 /* framework: {
43   components: [
44     'QLayout',
45     'QLayoutHeader',
46     'QLayoutDrawer',
47     'QPageContainer',
48     'QPage',
49     'QToolbar',
50     'QToolbarTitle',
51     'QBtn',
52     'QIcon',
53     'QList',
54     'QListHeader',
55     'QItem',
56     'QItemMain',
57     'QItemSide'
58   ],
59   directives: [
60     'Ripple'
61   ],
62   // Quasar plugins
63   plugins: [
64     'Notify'
65   ]
66   // iconSet: ctx.theme.mat ? 'material-icons' : 'ionicons'
67   // i18n: 'de' // Quasar language
68 }, */
69 animations: 'all', // -- includes all animations
70 // animations: [],
```

Annotations on the image:

- A green arrow points to line 41 with the text "Descomentar esta linha".
- A red bracket on the right side of the `components` array (lines 44-57) is labeled "Comentar este bloco".
- A green arrow points to line 63 with the text "Descomentar esta linha".
- A red arrow points to line 68 with the text "Comentar esta linha".

Configuração inicial

Para facilitar, vamos criar um script para execução da aplicação:

- ▶ Na pasta do projeto, edite o arquivo “package.json”, adicionando na linha 9, em “scripts”, o seguinte código: “dev”: “quasar dev”.

```
9   "scripts": {  
10     "lint": "eslint --ext .js,.vue src",  
11     "test": "echo \"No test specified\" && exit 0",  
12     "dev": "quasar dev"  
13   },
```

Também vamos configurar o endpoint da API na aplicação:

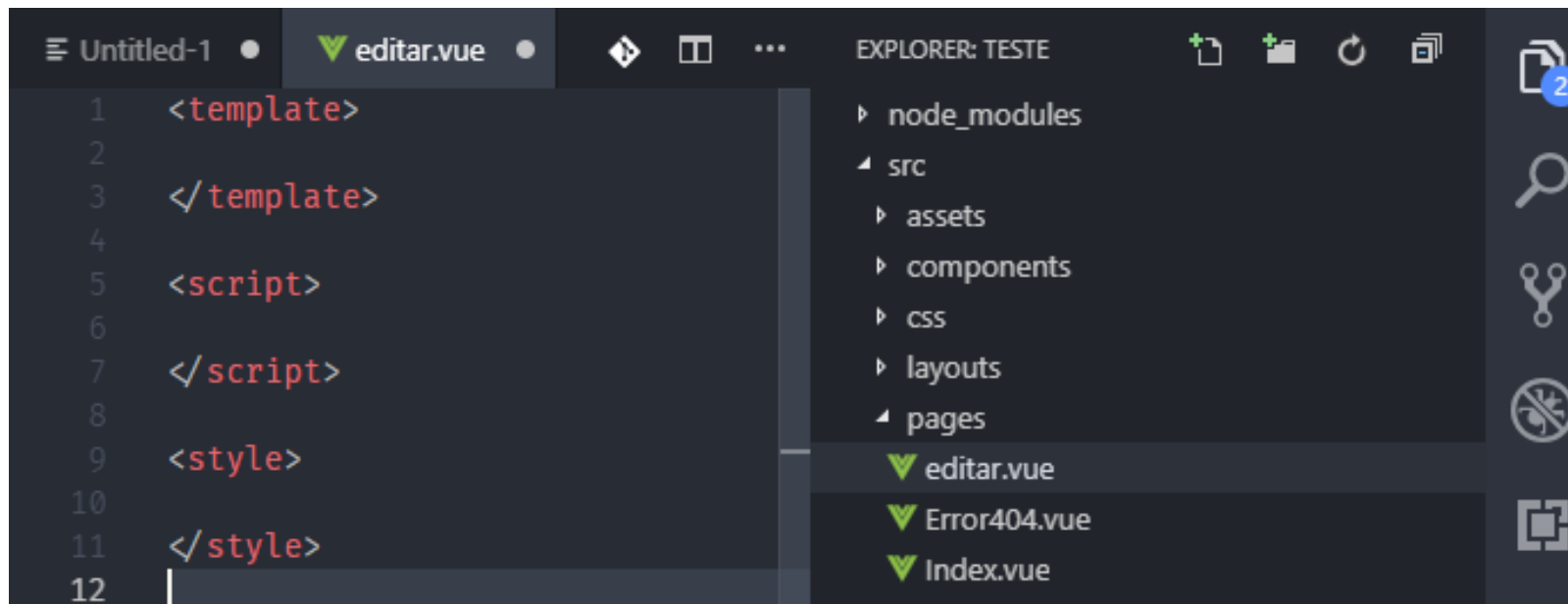
- ▶ Editar o arquivo “quasar.conf.js”, adicionando o código a seguir nas configurações do devServer na linha 36 e descomentar o código da linha 17 e 22.

```
proxy: {  
  '/api': {  
    target: 'http://127.0.0.1:3000',  
    changeOrigin: true  
  }  
},
```


Página para cadastro de clientes

Agora vamos criar nossa página que vai conter o formulário de cadastro de clientes. Para isso, vamos criar “src/pages/editar.vue”.

Por padrão, a estrutura dos arquivos (.vue) utilizada em projetos do Quasar, é composta por: **template**, **script** e **style**.



The screenshot shows a code editor with a dark theme. The active file is 'editar.vue', which contains the following code:

```
1 <template>
2
3 </template>
4
5 <script>
6
7 </script>
8
9 <style>
10
11 </style>
12
```

The Explorer panel on the right shows the project structure:

- node_modules
- src
 - assets
 - components
 - css
 - layouts
 - pages
 - editar.vue (selected)
 - Error404.vue
 - Index.vue

Página para cadastro de clientes

Usando o conceito de components, vamos criar um componente para o formulário em “src/components/form.vue” e usar na página criada anteriormente.

```
▼ form.vue  ▼ editar.vue x
1  <template>
2    <q-page class="row flex-center fit bg-grey-4">
3      <form-cadastro />
4    </q-page>
5  </template>
6
7  <script>
8    import FormCadastro from '../components/form'
9
10   export default {
11     name: 'PaginaCadastroClientes',
12     components: { FormCadastro }
13   }
14
15  </script>
16
```

Componente com o formulário

Agora vamos criar o formulário de cadastro de clientes utilizando alguns dos componentes de formulário do Quasar.

- ▶ Código: <https://pastebin.com/SU9RWLqm>
- ▶ Quasar componentes:
 - ▶ q-card: <https://quasar-framework.org/components/card.html>
 - ▶ q-field: <https://quasar-framework.org/components/field.html>
 - ▶ q-input: <https://quasar-framework.org/components/input-textfield.html>
 - ▶ q-btn: <https://quasar-framework.org/components/button.html>

Listar clientes na página inicial

Para exibir todos os clientes na página inicial, vamos usar o componente q-table do quasar, conforme código a seguir:

- ▶ Código: <https://pastebin.com/JnQUT6ai>
- ▶ Quasar componentes:
 - ▶ q-table: <https://quasar-framework.org/components/datatable.html>

Também vamos adicionar atalhos para home e novo cliente na barra lateral:

- ▶ Código: <https://pastebin.com/SjgjCeGu>
- ▶ Quasar componentes:
 - ▶ q-layout-drawer: <https://quasar-framework.org/components/layout-drawer.html>
 - ▶ q-list: <https://quasar-framework.org/components/lists-and-list-items.html>

Rotas

Por padrão, ao criar o projeto, o Quasar já cria a rota para acesso a home (“/”, página index). Então precisamos criar apenas a rota para criar/editar o cadastro. Copie e cole o código a seguir em “`src/router/routes.js`”:

- ▶ Código: <https://pastebin.com/Kf2zUWJV>
- ▶ Quasar:
 - ▶ App Routing: <https://quasar-framework.org/guide/app-routing.html>
- ▶ Vue:
 - ▶ Vue Router: <https://router.vuejs.org>

Build da aplicação

Depois de concluída a aplicação, vamos fazer o build e salvar o código na pasta “client” da API.

► Comando: `quasar build`

Logo após, configuramos nossa API para utilizar o conteúdo da pasta client para renderizar os dados, adicionando o código a seguir, em “files” linha 43 no arquivo “server/middleware.json” e comentamos todo o código do arquivo “server/boot/root.js”.

```
"loopback#static": {  
  "paths": [  
    "/"  
  ],  
  "params": "$!../client/"  
}
```

E assim, concluímos nossa aplicação.



Aplicação com Node.js e Vue.js/Quasar

Gestão de clientes (CRUD)