Vue Project

# Documentation Rick & Morty Web

Silvia Venegas

Silvia Venegas

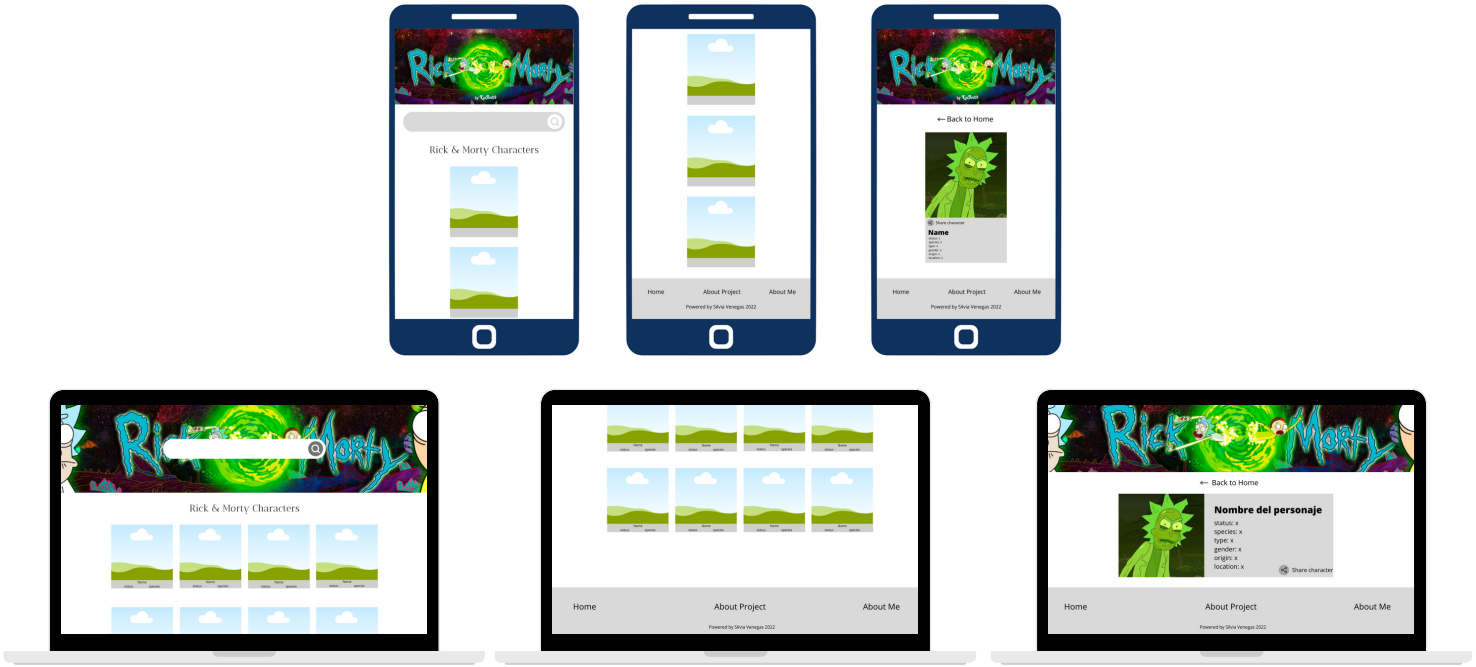
# Index

# Introduction

The Rick & Morty Website has been developed with the purpose of responding to a technical test, this website is composed of two different pages.

First of all, we will see the main page, **Home**, on this page, we have a list of characters with general information.
Secondly, we have an individual page for each character with all their detailed information.
In addition, I have also implemented a search engine to the home page, this feature allow us to filter by character name.

For the design, a quick mockup has been created with which to be guided.



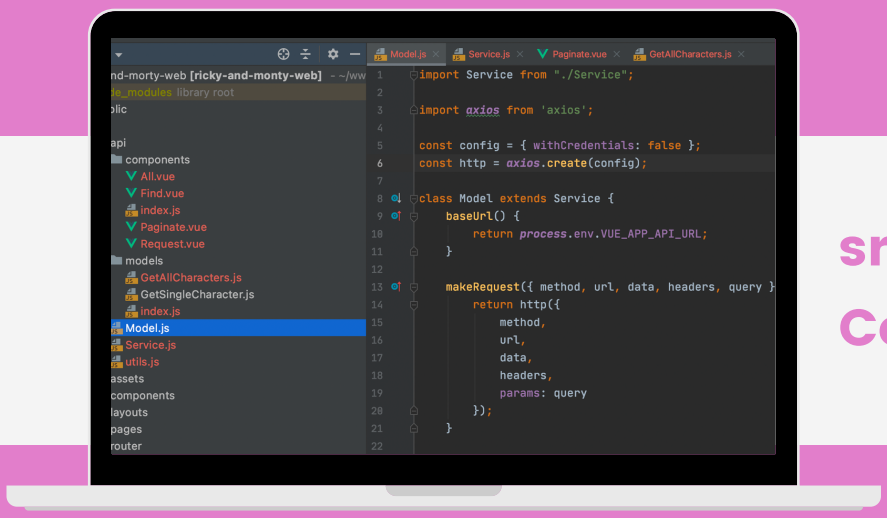

The tech stack used for create the website is: Vue.js, Tailwind Css and Axios.
The project has been structured in API (models and components), components, layouts, pages and router.

# API

For consuming the APi and execute actions against it, I have used Axios and some components based on JavelJS.
JavelJS is a JS package, is a mix of JS and Laravel which help us to don't repeat the same logic in our code (DRY) and provide us a a base model to configure our project with all the requirements.

Service.js -> Is the file where we will define our database connection, for instance, the URL where we will have to use in our API calls.
Model.js -> The file of our database models, there we will import Axios and we will add our configurations and overwrite our customizations from Service.js

Finally, we have 2 folders, Models folder, is where we will add all our API models which we need to work with (is the place where we will define our URL, configs, queries, etc...).
The components folder, the place where we will have all our base actions with all the logic that we need for each kind of API call (GET/POST/UPDATE/PATCH/DELETE..., only GET is used).




src/api/Modej.js
Connection base

# How we use it

The API components are the most important files of the project.
Their use is quite simple and in this project we are going to use Paginate and Find.

## 1 Import a Component

We have to import the components in all the places where we want to do a request to the API.

## 2 Send the needed Props

In order to make API calls works, we have to let our component know which model we want to use, the query in case we need to use arguments to filter, like "for searching" or a "paginate".
If we want to have a call without press any bottom or similar, we need to use the parameter "immediate" -> true , and finally, we have to collect the data received inside the response inside the v-slot.

```
<template>
    <Paginate
        ref="paginateCharacters"
        :immediate="true"
        model="GetAllCharacters"
        :query="queryParams"
        v-slot="{ data: characters, loading, pages, request }"
    >
        <div class="w-full h-full">
            <div class="w-full h-64 flex items-center justify-center bg-header-image">
                <RenderlessSearch
                    v-model="search"
                    :await-search="true"
                    renderless-class="min-w-32"
                    @input="request"
                />
            </div>
```

```
:immediate="true"
model="GetAllCharacters"
:query="queryParams"
v-slot="{ data: characters, loading, pages, request }"
```

# Router

The VUE router is composed of 2 files, index.js and router.js
The first one, is where we import Vue, we import and declare VueRouter, then, we import the routes that we have declared in the following file, we declared the VueRouter and defined the needed functions for routing.
The Router.js file is where we defined all the routes that we want in our project (path, name, meta, redirects in the case they are needed and component, which is the route to the file that we want to show)


## authGuard:

In this project, we don't have login, so we only need the kind authGuard

## Keys del objeto:

```
{
    path: '/',
    name: 'characters',
    meta: { show: false, layout: 'app' },
    component: () => import(/* webpackChunkName: "characters-page" */
    '@/pages/characters'),
},
```

In the meta, we declared if we want to be visible or not in the first entry and the kind of layout that we want to show.
Also, we can have redirects declared, for instance, in the home, and in the components having them declared the chunkName in the route to avoid errors.

# To consider

This project has been created from my base in my personal Vue base skeleton, in order to follow with all the requirements, as they are simple, I did neither use Utils nor Store (which is the part where the localStorage is managed and the project state), the majority of the API components either.

I would like to develop the character's paginator, but due to the little availability of time that I have, I decide to don't do it.
I had issues with CORS, which took me so much time to fixing them, but finally were fixed.

In order to start the project:

1

## npm install

Obviously.... "npm install, installs all the packages (defined in the package.json) in the node_modules folder in your root working directory", which is the same, it puts everything needed to make the all the stuffs to see them awesome.

2

## npm run serve --https

Use to setup a server with the project.
I had issues with CORS, so, for that reason I use the flag --https in order to solve them.

3

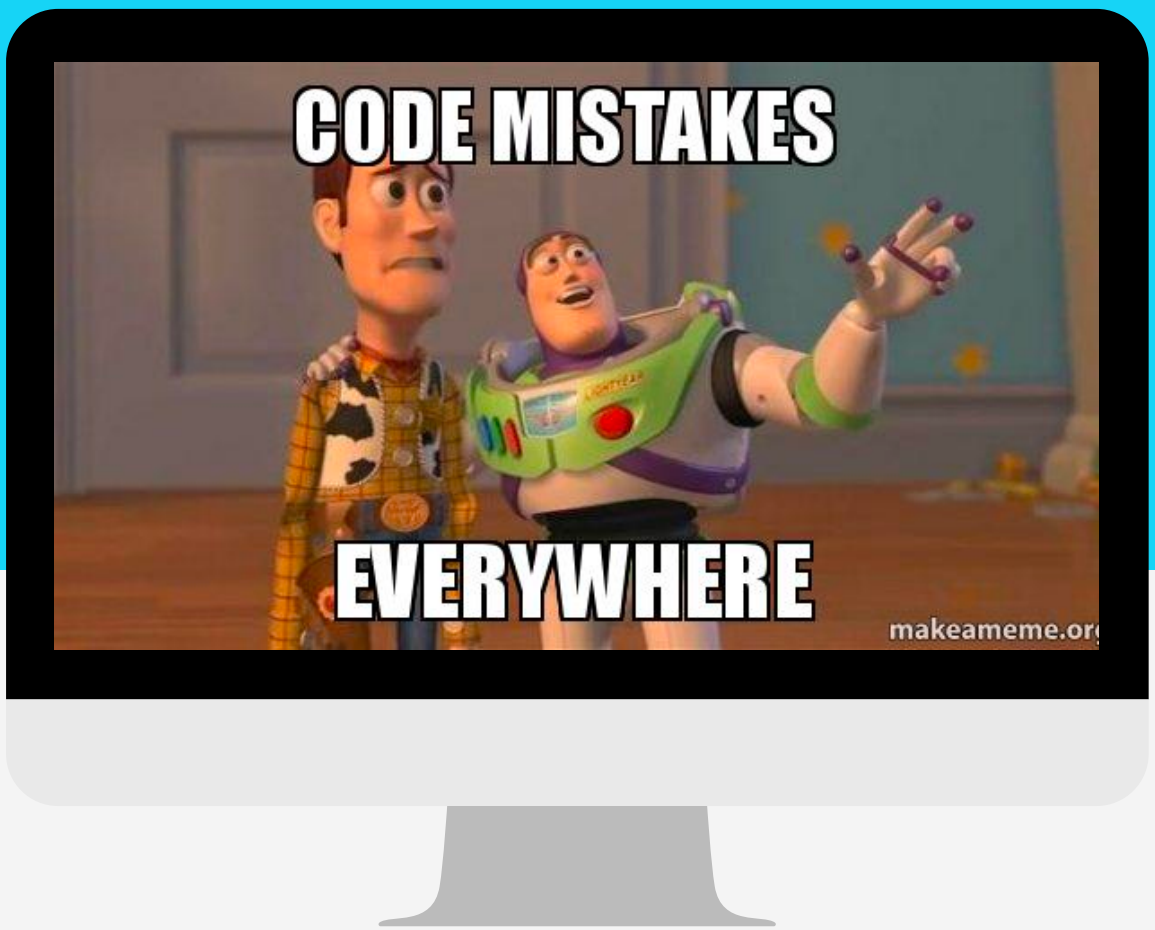
## Pray

Naaah, I'm kidding hahah
But well, maybe for the CORS topic will be reality (who knows)
I had to map the route (a part of doing some witchcraft) to work without problems.

# Conclusion

**I hope I have fulfilled all the requirements of the test and that we keep in touch :)**

**Thanks!**



Silvia Venegas