Vue Project

Documentation Rick & Morty Web

Silvia Venegas

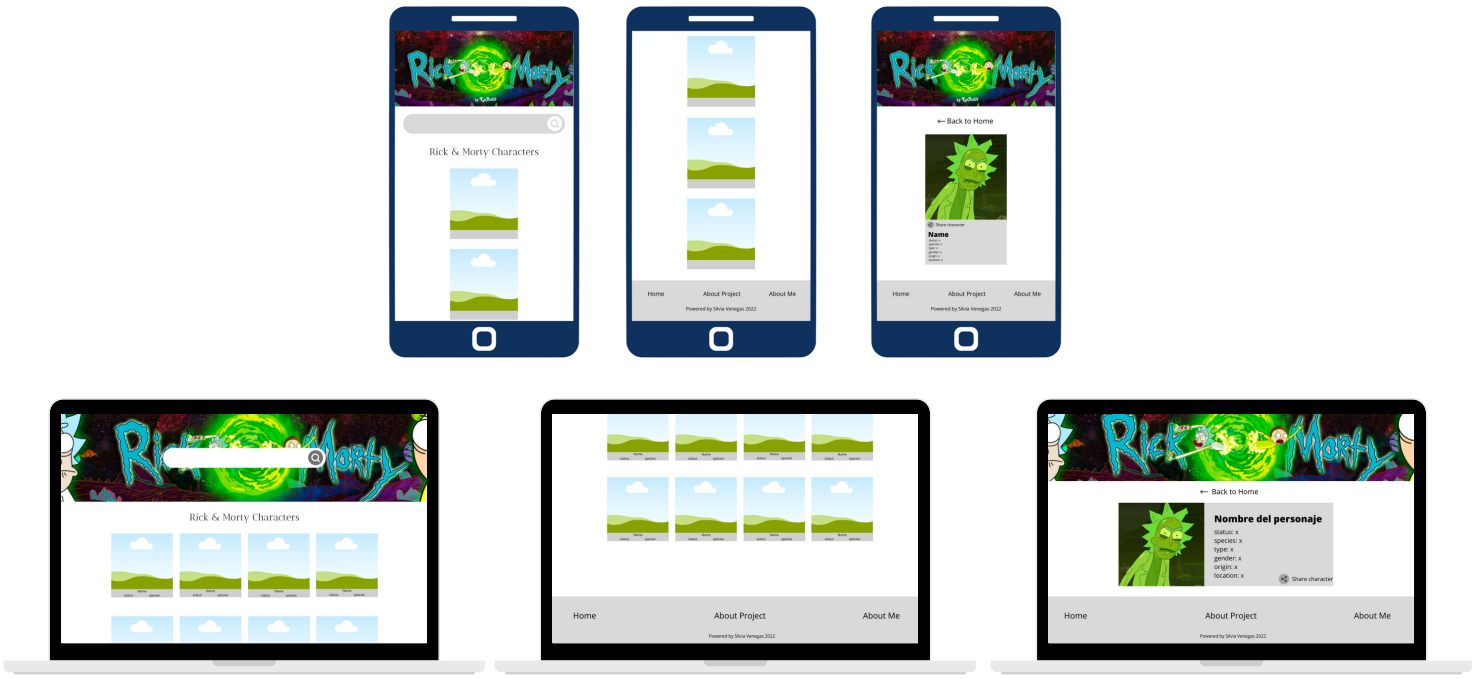
Silvia Venegas

Index

Introduction

The Rick & Morty website is made to respond to the technical test where a website made of two type of pages was requested.
On the one hand, making Home we have a list of characters with general information, and another individual page for each character with all their information.
In addition, a search engine has been added to the home page to be able to filter by character name.

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



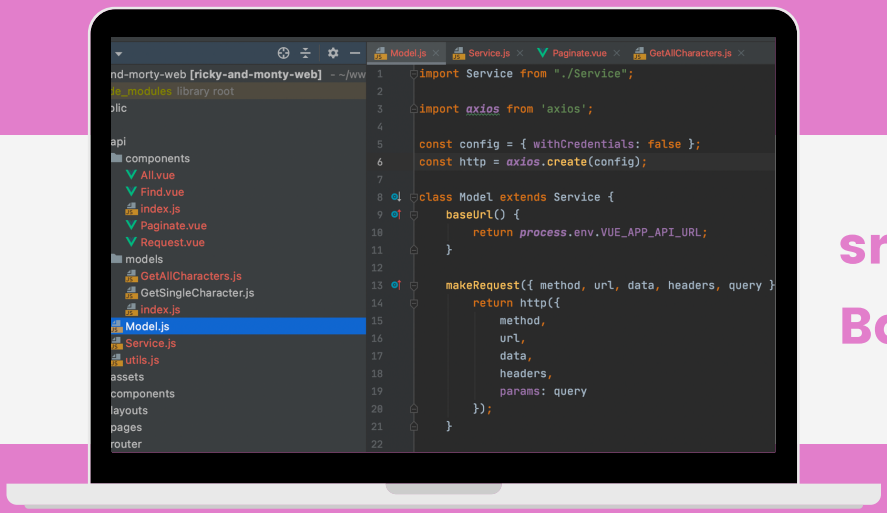
The test is done with Vue.js, Tailwind Css and Axios.
It has been structured in Api (models and components), components, layouts, pages and router.

Api

Para crear las conexiones con la api y realizar las acciones pertinentes he utilizado Axios y una serie de componentes que están basados en Javeljs. -Javel es un paquete, mezcla de JavaScript y Laravel, que nos ayuda a no repetir la misma lógica en el código y proporciona un modelo base que podemos configurar para que cumpla todos los requisitos de nuestro proyecto.-

Tendremos por un lado el archivo Services.js que es donde definiremos las bases de nuestra conexión, como por ejemplo montar la url a donde tenemos que llamar.
De otra parte tenemos el archivo Model.js que será la base de todos nuestros modelos. Ahí es donde importamos Axios, añadimos nuestras configuraciones y sobre-escribiremos lo que necesitemos personalizar del Services.js.

Por último tenemos 2 carpetas, la carpeta models que es donde añadiremos todos los modelos de la api con los que necesitemos trabajar (es donde terminamos de definir las ulrs, configs, query, etc.) y la carpeta components que es donde tenemos definidos nuestras acciones base con toda la lógica que necesitamos para cada tipo de llamada (GET/POST/UPDATE, aunque en este caso solo necesitaremos hacer GET).



src/api/Modej.js
Base de toda la conexión

Como lo usamos

Los componentes de la api son los archivos más importantes del proyecto. Su uso es muy sencillo y en este proyecto los que vamos a utilizar son el Paginate y el Find.

1 Importar Componente

Como con todos los componentes que utilizamos, debemos importarlo en todos los sitios donde queramos hacer llamadas a la api.

2 Mandar las props necesarias

Para que las llamadas funcionen necesitamos decirle al componente el modelo al que queremos llamar, la query en el caso que necesitemos pasar parámetros para filtrar como un "search" o un "paginate", si queremos que sea una llamada que se ejecute sin necesidad de botón o similar necesitamos pasarle el parametro "inmediate" a true y por último debemos recoger los datos que nos devuelve la llamada en el 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

El router de vue se compone de 2 archivos, el index.js y el router.js.
En el primero es donde importamos Vue, Vue Router y las rutas que tenemos declaradas en el siguiente archivo, declaramos el VueRouter y definimos las funciones necesarias y listo. En el archivo router.js es donde definimos las rutas que queremos en nuestro proyecto (path, name, meta, redirects en el caso de ser necesarios y component, que es la ruta del archivo que queremos mostrar)

authGuard:

En este proyecto no tenemos un login por lo que solo necesitamos el tipo de guarda auth

Keys del objeto:

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

En el meta declaramos si queremos que sea visible o no en la primera entrada y el tipo de layout que queremos mostrar.
También podemos tener redirects declarados como por ejemplo en la home y en el component tenemos declarado el chunkName de la ruta para evitar errores.

To consider

El proyecto es un mini proyecto creado desde mi proyecto vue base personal, para cumplir con los requisitos y al ser estos muy sencillos, no se utiliza ni un utils, ni la store (que es donde se controla el localStorage y el state del proyecto), ni la mayoría de componentes de la api.
Me hubiese gustado hacer un paginador para los personajes pero debido a la poca disponibilidad de tiempo libre que tengo y que la api de rick & morty tampoco es que esté muy apañada para ello decidí no hacerlo.
Tuve problemas de CORS que me robaron mucho tiempo también aunque finalmente se solucionó.

Para poder ver el proyecto y dejar el hype:

1

npm install

Esto es esencial, obvio... "npm install installs the package in a node_modules folder in your parent working directory" o lo que es lo mismo, te mete todo lo que necesitas para ver las cosas chachi.

2

npm run serve --https

Para poder levantar el servidor debes ejecutar este comando. Lo habitual es utilizar solo "npm run serve" pero por los problemas de CORS mejor añade la flag --https y nos curamos en salud :)

3

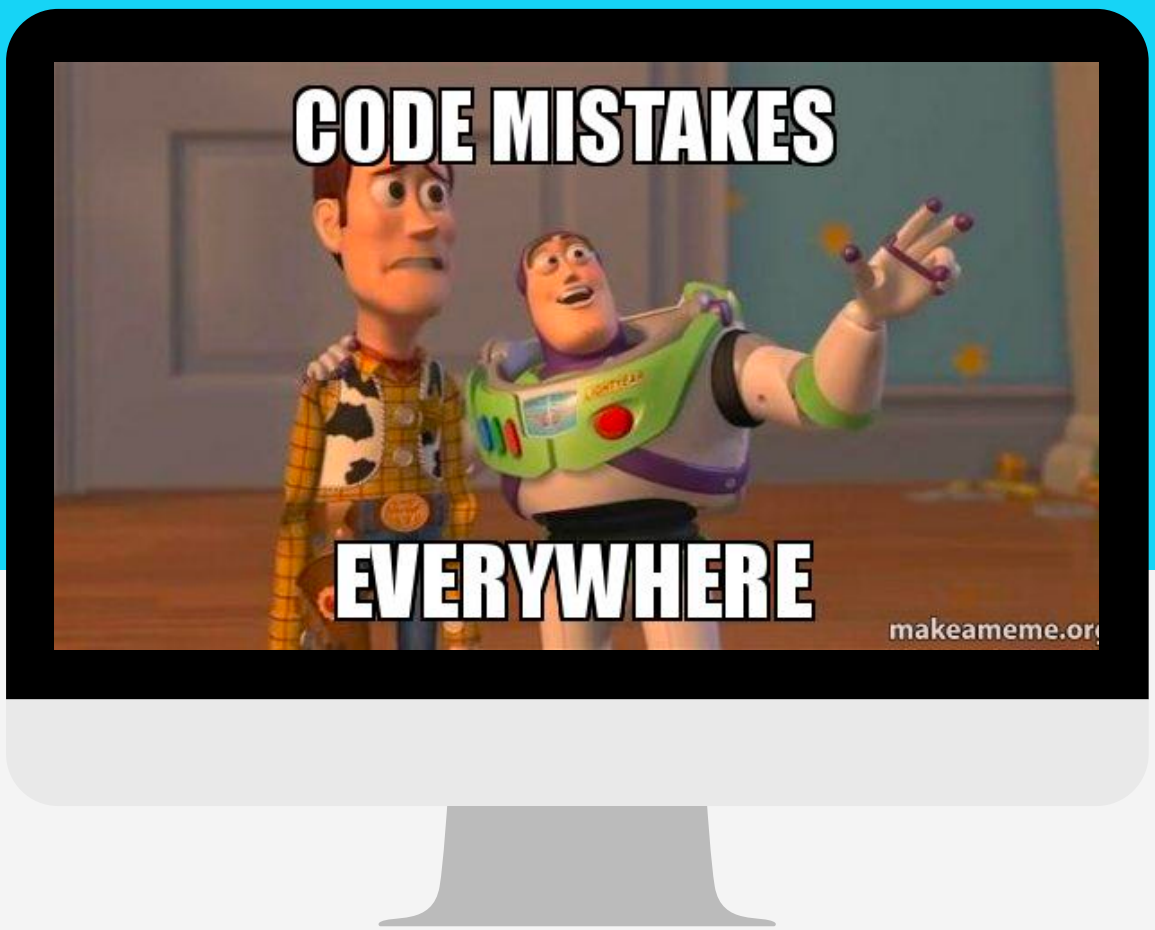
Pray

Naaah, esto es coña jajaja. Aunque bueno, igual por el tema CORS se hace un poco realidad... Yo tuve que mapearme la ruta (aparte de muchas otras brujerias) para poder trabajar sin problemas.

Conclusion

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

Thanks!



Silvia Venegas