

Vuetify + Vue.js + Firebase



Un framework progresivo

## Agradecimientos

*Dar agradecimiento a todas aquellas personas que han ayudado a la creación de este proyecto, ya sea de un modo u otro, incluyendo a mi familia.*

*Muchas gracias a los compañeros del ciclo por el gran compañerismo y los buenos momentos, ¡Seguiremos en contacto!*

*Dar las gracias al equipo educativo del FP de Mislata por la enseñanza y los valores recibidos y por ayudarme a adentrarme en este mundo.*

*Dar las gracias al equipo de Digital Service Center de Sopra Steria por la fantástica formación recibida, incluyendo a mis compañeros que me han ayudado con el transcurso de las FCT's, incluyendo a Rubén Rica, Carles Sanz y Juan Ripolles.*



## Tabla de contenido

1.-Marco de investigación .....	6
1.1.-Temática elegida.....	7
1.2.- Contextualización .....	8
2.-Organización del proyecto.....	9
2.1.-Temporalización.....	9
2.2.-Recursos.....	11
2.2.1.-Entornos de desarrollo .....	18
3.-Aplicación práctica .....	20
3.1.-Introducción .....	20
3.2.2.-Diseño .....	24
3.2.3.-Implementación.....	26
3.2.4.-Pruebas.....	39
4.-Manual de uso.....	41
5.-Valoración personal.....	49
6. Fuentes bibliográficas .....	51

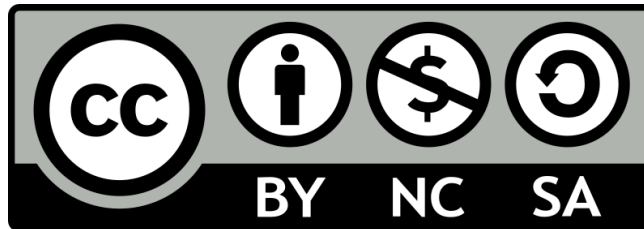
Este trabajo se encuentra bajo la licencia de "Creative Commons"

**Reconocimiento-NoComercial-CompartirIgual**  
**CC-BY-NC-SA**

Esta licencia permite a otros entremezclar, ajustar y construir a partir de esta obra con fines no comerciales, siempre y cuando le reconozcan la autoría y sus nuevas creaciones estén bajo una licencia con los mismos términos

Sobre esta [licencia](#)

Sobre el [codigo legal](#)



# 1.-Marco de investigación

El objetivo de este proyecto es de aprender a utilizar el Framework, llamado **Vue.js**, el cual ha cogido fuerza durante los últimos años, pero no solamente de este **frameworks**, si no todas las herramientas que envuelven el desarrollo web, como pueden ser las librerías visuales, como **Bootstrap** (En este proyecto se utilizará **Vuetify**), herramientas de gestión de tareas (En este caso **WebPack**), para poder trabajar más rápido y facilitar las cosas y la utilización de una base de datos no relacional para tener algún tipo de '**Back**'.



Las principales razones de utilizar **Firebase** junto con **Vue.js** es que ambas se combinan muy bien, a base de diferentes sentencias que lanzamos desde el lado del cliente, **Firebase** nos devuelve unos datos almacenados con los cuales podemos trabajar, normalmente este es un objeto de tipo observable, el cual nos devuelve datos en tiempo real de la tabla a la cual estemos haciendo referencia, algo muy útil cuando queremos trabajar con datos en tiempo real desde el lado del cliente.

## 1.1-Temática elegida

Ahora bien, ¿Por qué he decidido utilizar Vue.js?, sobre todo era porque había mucha gente que hablaba de este framework, y eso me dio curiosidad, ya había visto muy por encima AngularJS y React, por lo que esto me incitó a empezar a utilizarlo. Esto me daría una visión más amplia de los que es un framework, y las diferentes formas de embarcar un proyecto.

Para este proyecto he decidido crear el proyecto de **Stick a note**, el cual es muy sencillo, simplemente se trata de una lista de tareas, con un sistema de Log In, llevado por Firebase donde el usuario puede poner todas las tareas que el desee. Con esta idea podía empezar algo sencillo sin necesidad de hacer otra cosa que requiera de una lógica mayor como una tienda E-Commerce, con la idea de aprender los conceptos básicos de este framework y cómo hacerlo trabajar con Firebase para poder montar una aplicación.

Para este proyecto he seguido varios tutoriales sobre Vue.js, incluido el gratuito que ofrece la página de VueSchool, estarán incluidos en la bibliografía al final de esta memoria, al igual que todo el código comentado en este proyecto, el cual estará tanto en GitHub como en BitBucket.

## 1.2.- Contextualización

La idea de la realización de este proyecto vino gracias a una tarea parecido a la realizada en el ciclo, en este caso era una lista de tareas, en la cuál se podían añadir, borrar y modificar tareas. En este caso simplemente se trabaja con Angular en el lado del cliente, no existía ningún 'back' real, ya que al recargar los datos se borraban las nuevas tareas introducidas

Así que me dispuse a aplicar alguna especie de back, y me decidí por firebase, ya que se adaptaba perfectamente con Vue.

En el anterior proyecto, tampoco existía un sistema de login de usuarios, así que también me dispuse a realizar uno con firebase, ya que tenía una parte dedicada a los usuarios, así que aprovechando lo que ofrecía me dispuse a realizar un sistema de usuarios sencillo, para poder distinguir las tareas de cada usuario.



Simplemente ***'Keep it Simple'***



## 2.-Organización del proyecto

### 2.1.-Temporalización

Este proyecto se basa en tres fases bien diferenciadas, la primera es la fase de preparación donde se decide cual va a ser la aplicación y la tecnología que se va a implementar, luego la segunda fase es sobre todo de investigación sobre las tecnologías que se van a utilizar y la última es la realización de la propia aplicación, donde se aplicarían los conceptos aprendidos en la fase de investigación.

#### **1.-Preparar la aplicación**

Esta fase fue la primera de todas, durante la primera semana de abril, estuve tanteando en que realizar el proyecto, si sobre Angular o Vue, el primero ya lo había visto al menos un poco, el segundo simplemente lo había oído que existía, pero al final decidí hacerla sobre Vue

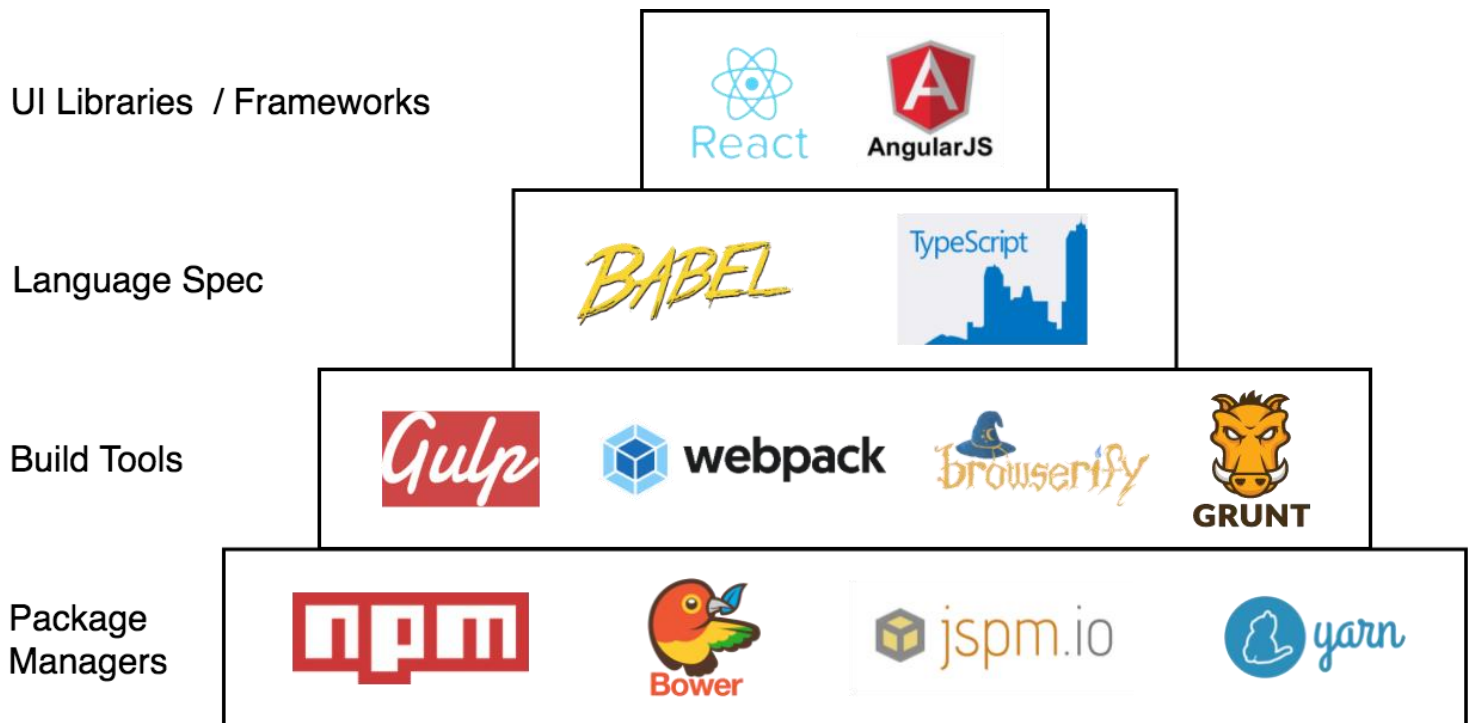
#### **2.-Investigación de tecnologías**

Una vez decidida la tecnología que se va a implementar en el proyecto, hay que aprender a utilizarla, por lo que las siguientes 2 semanas me dispuse a ver tutoriales y haciendo proyectos de prueba para aplicar los diferentes conceptos de Vue, esta fase fue la más difícil, ya que no solo se trataba de Vue, si no de todas las tecnologías web que hay alrededor, sobre todo WebPack, que iba a ser mi principal administrador de tareas, con todo lo que ello incluye (Node.js, Npm, etc.).

### 3.-Aprender a aplicarlos en una 'App' real

Esta fase consistió básicamente en aplicar los conceptos aprendidos durante esas dos semanas a practicar y a utilizar los conceptos en una aplicación real y funcional, esta es la fase más ardua, y donde me toco más de una vez empezar desde el principio. Esta fase duró aproximadamente 3 semanas y media. Incluyendo el diseño de la página web y las diferentes funcionalidades. Realmente podemos implementar la segunda fase aquí también, porque nunca se para de visitar nuevos tutoriales.

Pirámide estructural



## 2.2.-Recursos

### ¿Qué lenguajes han sido necesarios para la creación de este proyecto?

Empezando por el principio, los dos principales lenguajes de toda aplicación que se aprecie, HTML y CSS3

**HTML:** Lenguaje de etiquetas, que sirve principalmente para definir la estructura básica de una página web.

**CSS:** En este caso en nuestro proyecto al utilizar Vue.js, el css está limitado a las etiquetas definidas dentro del archivo, pero aun así nos ayuda a maquetar y definir los estilos y formas que necesitamos.



Y estos dos lenguajes no podían ser acompañados sin el lenguaje **JavaScript**, en este caso utilizado en su actual versión de ECMAScript6 (del 2016), el cuál dota de dinamismo toda la web que vamos a ver, y es el lenguaje que utiliza Vue.js





## ¿Qué es Vue.js?

Vue.js es un framework de Javascript, **Open-Source** creado originalmente por **Evan You**, se trata de un **framework progresivo**, eso significa que podemos adaptar nuestras necesidades al programa o software que estemos desarrollando, por lo que esta tecnología sirve para proyectos grandes, o para proyectos pequeños, de ahí su funcionalidad **progresiva**.

Este framework lo podemos utilizar sin la necesidad de utilizar Webpack o cualquier otro tipo de tecnología, incluso se puede importar con un sencillo script el cual podemos poner directamente en nuestro código HTML, evidentemente esto no se haría en un proyecto más grande.

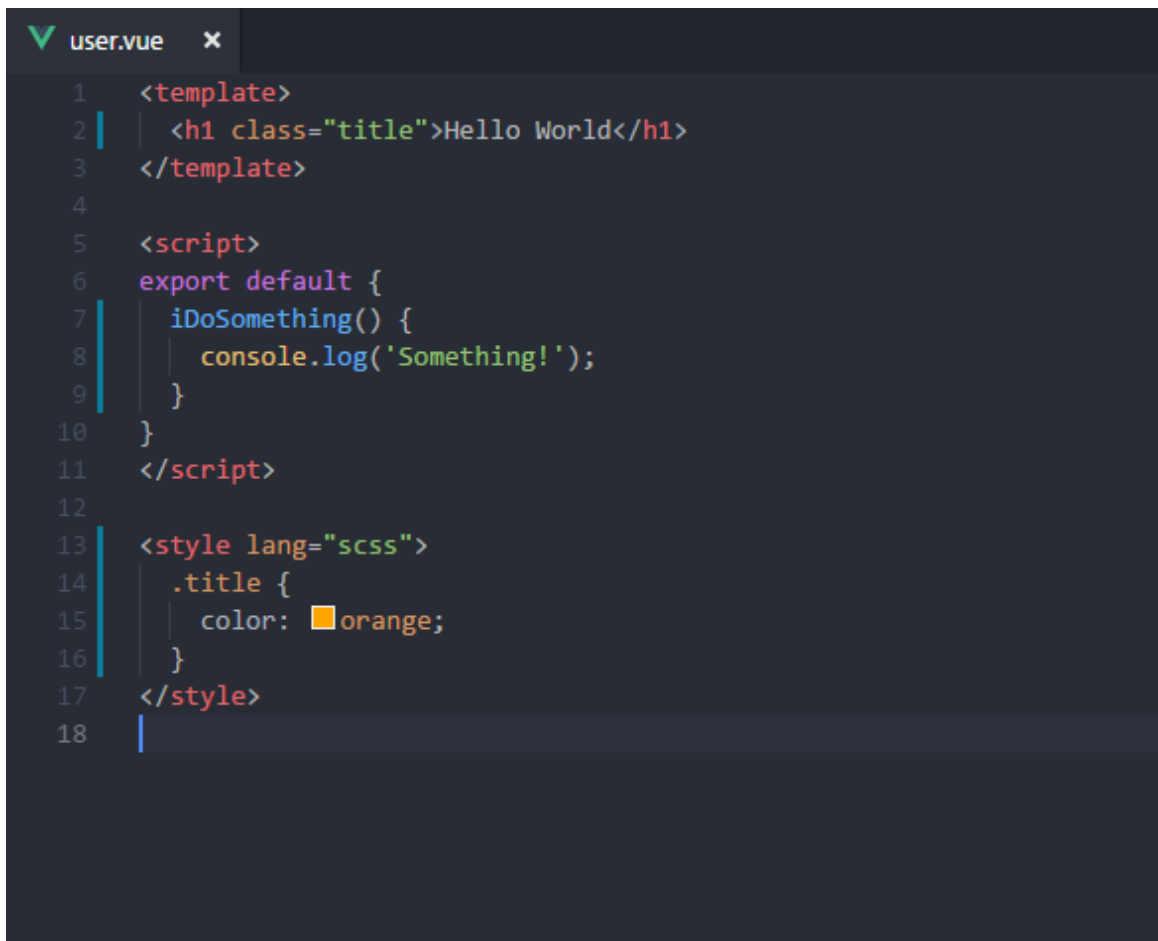
Vue.js junta varios conceptos con los existentes frameworks, sobretodo de **ReactJS** y **Angular**. Está basado en la programación reactiva y está orientado a la **reutilización** de los componentes.

Lo que me ha resultado curioso es que es más amigable y sencillo de aprender a usar que los frameworks como Angular, debido a su extensa documentación y la gran cantidad de usuarios y páginas que ofrecen cursos para aprender a usar este framework.

¿Y cómo está organizado un archivo .vue ?

Para empezar un archivo .vue está dividido en tres grandes bloques:

- **Template:** Esta etiqueta contendría todo el código HTML
- **Script:** Aquí se pondría todo el código JavaScript que deseamos ejecutar.
- **Styles:** Dentro de esta etiqueta se pondrían todos los estilos que se aplicaría a el código HTML del template



```
1  <template>
2  |   <h1 class="title">Hello World</h1>
3  | </template>
4
5  <script>
6  | export default {
7  |   iDoSomething() {
8  |     console.log('Something!');
9  |   }
10 | }
11 | </script>
12
13 | <style lang="scss">
14 |   .title {
15 |     color: orange;
16 |   }
17 | </style>
18 |
```

Así quedaría la estructura principal de un archivo .vue



## ¿Qué es Vuetify?

**Vuetify** es un proyecto Open Source de carácter **MIT** que ofrece una gran cantidad de componentes semánticos para Vue, los cuales son bastante elegantes y están de acuerdo conforme la especificación de Google sobre "Material Design".

Aparte de ofrecernos una gran cantidad de componentes reutilizables con unos estilos predefinidos, nos ofrece también un sistema de Grid, como el de la famosa librería de **Bootstrap**.

Lo bueno de esta librería es que es soportada por la mayoría de los navegadores modernos, incluyendo IE11 y Safari 9+ (Usando PolyFills). Esta librería nos permite pasar de móvil a escritorio de manera sencilla, sin necesidad de estar configurándolo nosotros, ya que los propios componentes están diseñados para ello.

Lo interesante de esta librería visual, aparte de ofrecernos todos los componentes visuales que hay, los podemos importar como deseemos en nuestra aplicación sin necesidad de importar toda la librería, simplemente al iniciar el proyecto en WebPack, nos preguntará si queremos la versión 'A-la-carte-components', al elegir esta opción ya tendremos a nuestra disposición el trozo de código que nos ayudará a implementar los componentes que nos interesen.



¿Qué es firebase?

**Firestore** es una base de datos **NoSQL**, es decir, no es relacional. Aparte de ofrecer hosting gratuito (Ya que está en los servidores de **Google**), te incluye: estadísticas, datos en **tiempo real** de tu aplicación, un método para **autenticar** a los usuarios de tu aplicación, etc.

Firestore consta de 2 tipos distintos de bases de datos, la base de datos en tiempo real, que es la que tiene establecida firestore por defecto. Y **firestore**, que es una base de datos también en tiempo real, pero con más potencia en lo que son las 'Queries' o sentencias que tu le puedas mandar. Es muy parecida a la base de datos no relacional **MongoDB**, el problema es que se encuentra en la fase 'Alfa', y no es demasiado estable de momento.

Al utilizar Firestore todo lo relacionado con el 'Back', estaría solucionado, por lo que podíamos establecer que firestore es un **PaaS** (Platform as a Service), ya que te ofrece toda una serie de herramientas para que el desarrollador pueda concentrarse en lo que es el desarrollo de la aplicación sin tener que preocuparse mucho por los datos que le llegan, algo que en este caso me venía como anillo al dedo.

# Otras tecnologías utilizadas



*BABEL*



webpack



*Sass*



**Node.JS:** Aquí realmente no llegamos a utilizar Node.js como tal para abrir un servidor. Sobre todo se usa los módulos que nos ofrece, en este caso **NPM**, el cual nos ofrece una gran variedad de paquetes para trabajar.

**Webpack:** Es principalmente el sistema de 'Bundling' que utilizamos para preparar nuestro desarrollo. También nos permite **automatizar** los procesos principales que son **transpilar** y **preprocesar** código de .scss a .css, de ES7 a ES5/6 con la ayuda de Babel...etc. Ya que al final de todo, nos producirá una versión de **producción** compilada, la cual estará completamente **minificada**.

**Babel:** Aquí babel se utiliza básicamente para poder utilizar la última versión de Javascript sin preocuparse, ya que Babel se dedica a pasarlas a una sintaxis que pueda entender el navegador, básicamente convierte la sintaxis de ES6/7 a JavaScript plano

Put in next-gen JavaScript	Get browser-compatible JavaScript out
<pre>var miFuncion = (num) =&gt; num + num;</pre>	<pre>var miFuncion = function miFuncion(num) {   return num + num; };</pre>

### 2.2.1.-Entornos de desarrollo

Para la realización de este proyecto he trabajado sobre todo con **Visual Studio Code**, el cual está dotado de una gran cantidad de opciones de desarrollo, sobre todo con la gran cantidad de plugins que pone a disposición del desarrollador, entre ellas se encontraba el plugin de **Vetur**, que me ha ayudado bastante en cuanto a la **sintaxis** y el proceso de **formateo** del código.

Es posiblemente el mejor editor que ha pasado por mis manos de momento, también he probado '**Atom**' o '**Sublime text**', pero sin duda alguna **Visual Studio Code** se lleva la medalla, al menos para mí.



Durante el desarrollo, todos los **comandos** y las tareas de terminal, las he realizado o bien con la terminal que integra Visual Studio Code, o con la terminal '**Git Bash**', que se asemeja bastante a la terminal que ofrece Ubuntu, con todos los comandos exactamente iguales.

Para poder comprobar errores y debugar he utilizado las herramientas de desarrollador que ofrece Google Chrome para ello.

Desde casa he trabajado con **Windows** sobre todo, pero mientras estábamos en las FCT's el principal sistema operativo que utilizábamos era **Ubuntu**, el cual me ha sorprendido gratamente.

También he utilizado '**Terminator**', una herramienta que me recomendó un compañero de trabajo y que la verdad ha resultado muy útil a la hora de trabajar con varias **terminales**, ya que se podían tener varias terminales abiertas en la misma pantalla, sin necesidad de tener dos ventanas distintas.



Para la gestión de versiones se ha trabajado principalmente con **Git**, y el lugar principal donde se guardaban los repositorios del código fuente han sido **GitHub** y **BitBucket**.

## 3.-Aplicación práctica

La aplicación esta subida en la siguiente dirección Url

<https://fcojaviermt.github.io/#/>

Todo el código comentado a continuación se encuentra en el siguiente repositorio de GitHub.

<https://github.com/fcojavierMT/proyecto-final>

### 3.1.-Introducción

Para el despliegue de la aplicación se ha utilizado el servicio que ofrece github, las github pages, las cuales te permiten subir un repositorio con la versión de producción de tu aplicación, en mi caso al tener firebase, puedo subir un proyecto sin necesidad de utilizar otros medios para desplegar la aplicación.

# GitHub Pages

Para la implementación de este proyecto hablaremos de los primeros pasos y de cómo está diseñada (diagramas) así mismo de la estructura de carpetas y de los diferentes tipos de componentes que forman la aplicación en sí.

También se cubrirán los aspectos visuales y requerimientos necesarios para poder llevar acabo esta aplicación.

El proyecto consistía en tener una lista de tareas, pero con un 'Back' real y realizado con Firebase. Este proyecto requería la necesidad de tener usuarios, así que gracias a firebase y a su sistema de validación de usuarios, no era necesario crear otra tabla en la base de datos, ya que la autenticación venía por separado. Con eso ya teníamos cubierto la necesidad de tener usuarios.

Firebase al tener separado la parte de los usuarios asigna de manera automática un identificador único, el cual nos servirá para poder separar cada una de las notas de los usuarios individualmente.

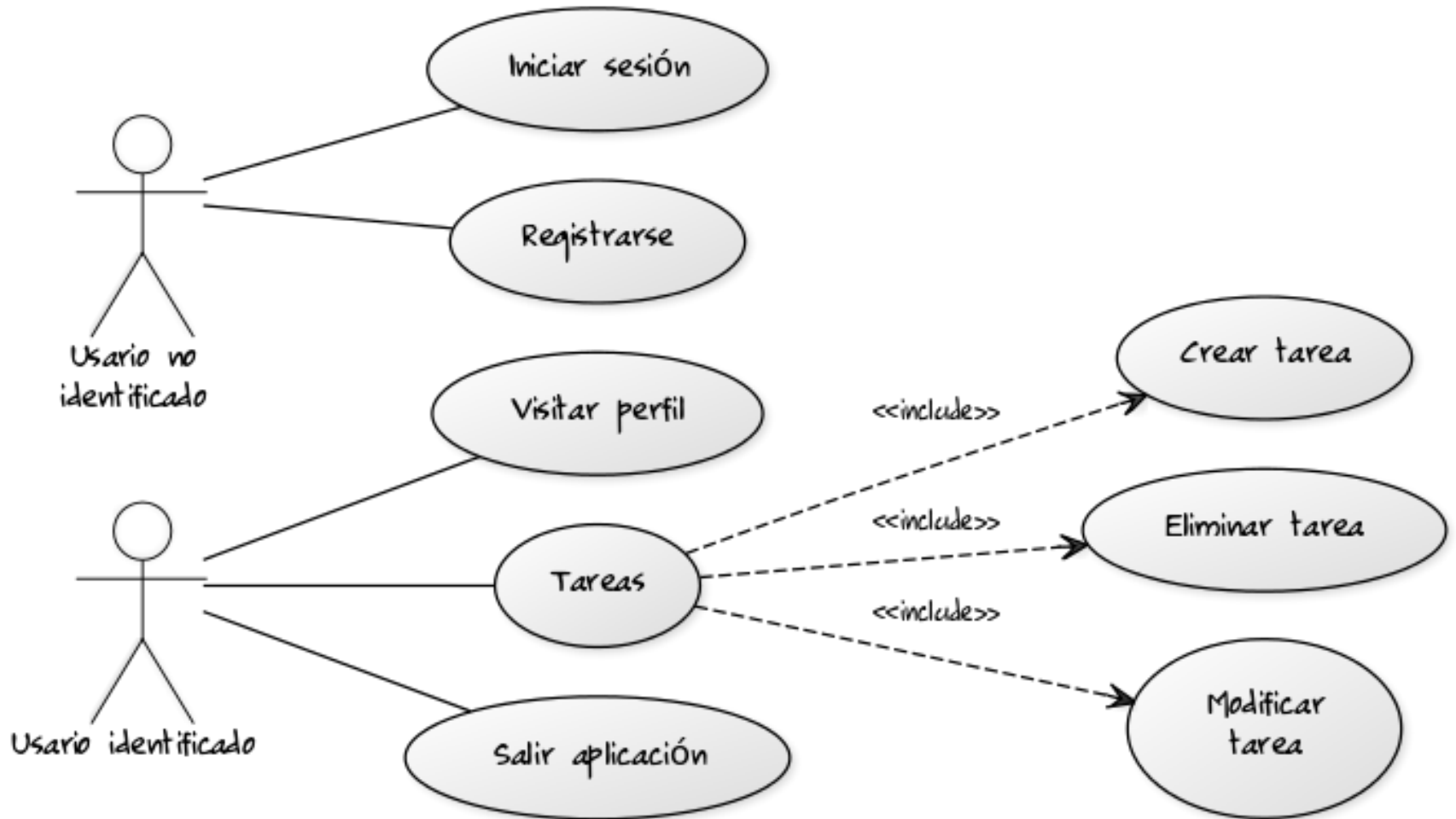
Lo segundo era tener cinco pantallas: Perfil, Tareas, Cerrar sesión, Iniciar Sesión y Registrarse. Para ello tendremos que tener dos tipos de usuario identificados:



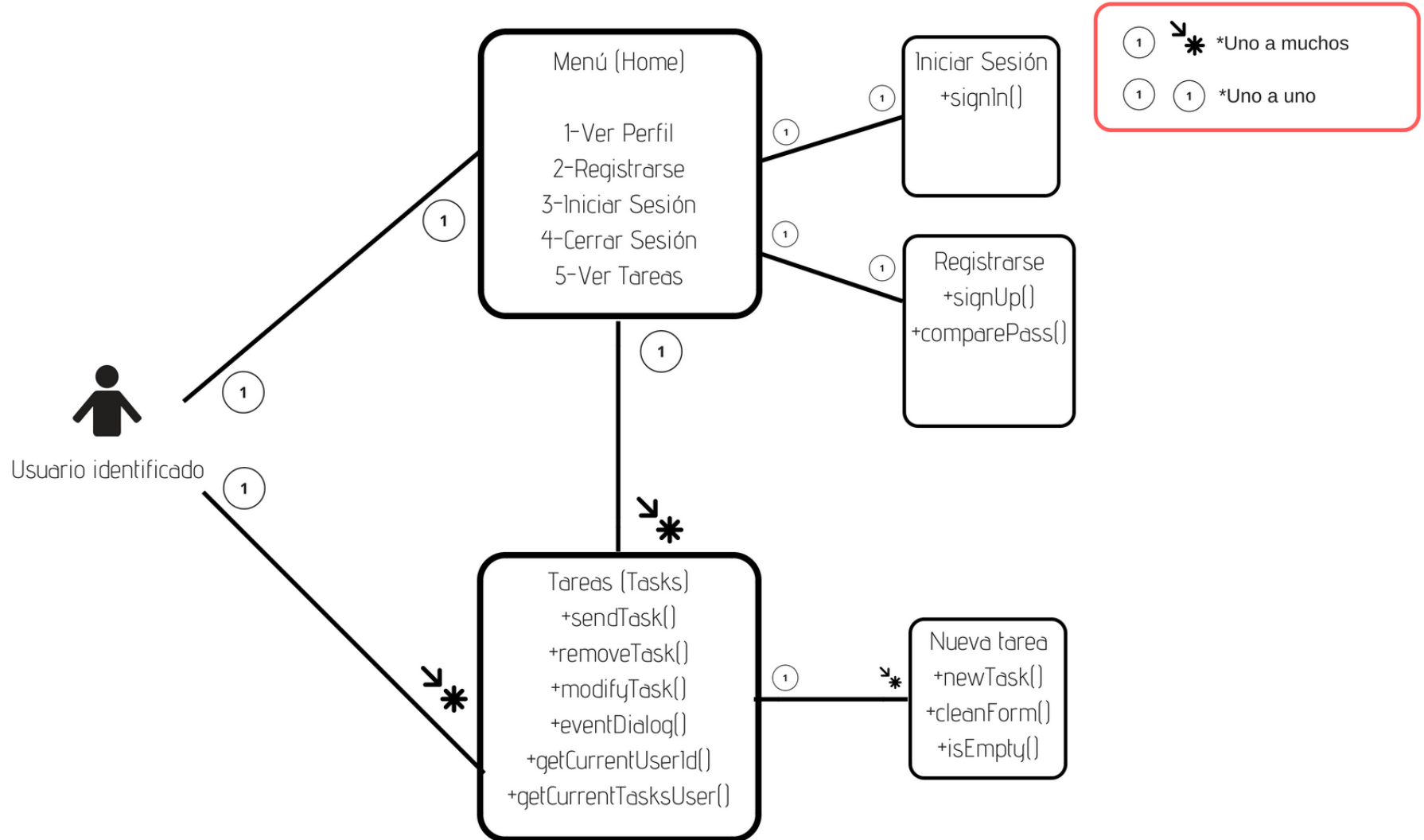
Requerimientos principales	Descripción del requerimiento
Usuarios	-Tener un sistema de 'Login', manejado por firebase
Lista de tareas (To-do)	-Que el usuario registrado pueda crear, eliminar y modificar cada una de sus tareas
Página de perfil	-Una página de perfil del usuario, donde pueda consultar sus datos.
Separación de componentes	-Tener dos tipos principales de componentes los componentes listos, y los componentes tontos Los listos, harán todas las consultas y las operaciones lógicas y los tontos, pintarán datos.

Requerimientos principales	Descripción del requerimiento
Requisitos mínimos (Móvil)	-La resolución mínima tiene que ser la que ofrece Iphone 5 (320px - 568px)
Requisitos mínimos (Desktop)	-Pantalla adaptativa, cuando llegue a resolución de tablet, ocultar menú principal, y habilitar menú lateral
Sitio Web	-La web estará habilitada para todo el mundo que quiera verla (Github Pages)
Conexión a internet	-Evidentemente, al ser una aplicación web, se necesita conexión con acceso a internet, para poder visualizar las diferentes tareas.

### 3.2.2.-Diseño





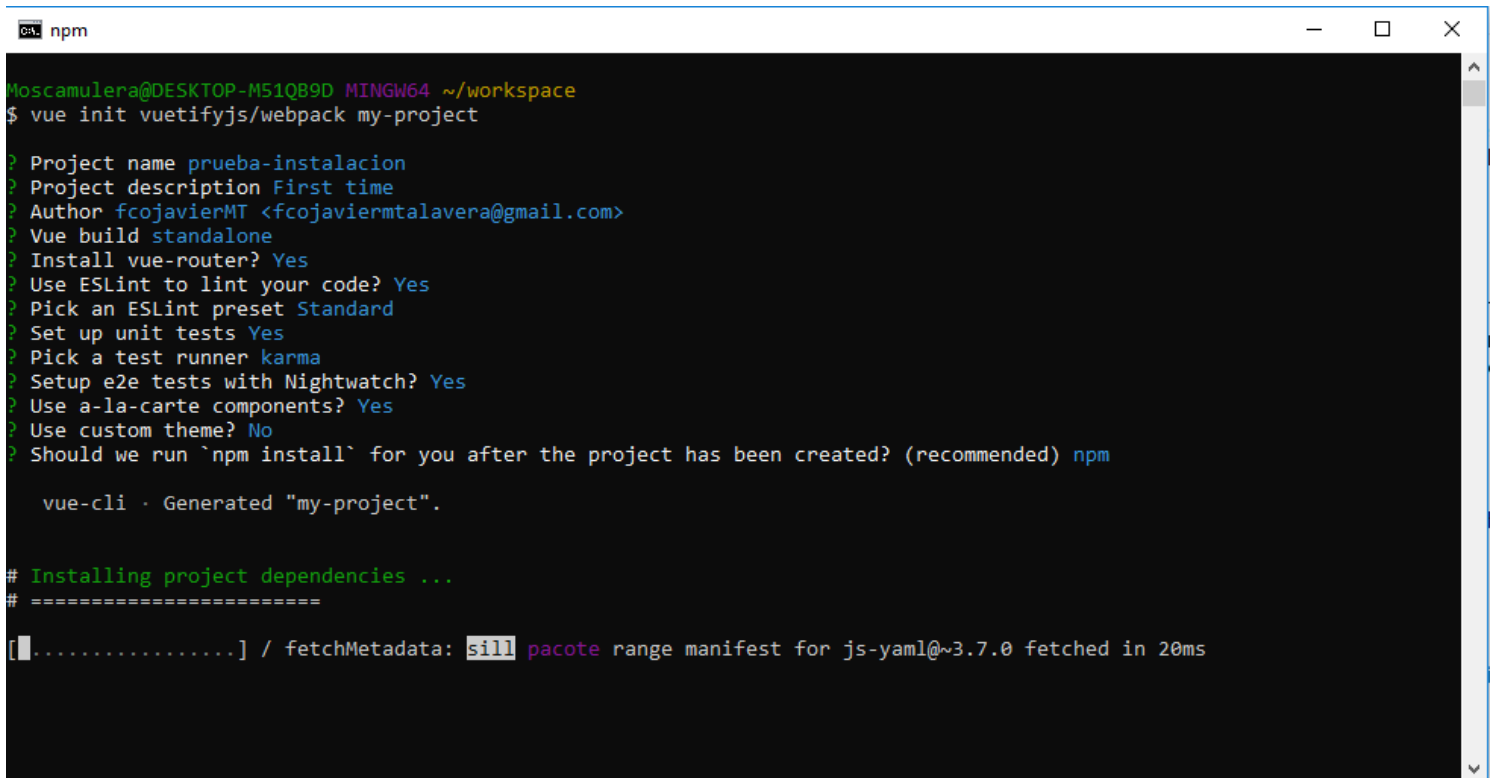


### 3.2.3.-Implementación

Para comenzar, tendremos que **generar** nuestro proyecto con WebPack, para ello ejecutaríamos los siguientes comandos, sobre todo tendríamos que tener instalado **node.js** previamente para poder **ejecutar** el primer comando que aparece:

```
npm install -g vue-cli
vue init vuetifyjs/webpack my-project
cd my-project
npm install
npm run dev
```

Al ejecutar el **comando** de WebPack, nos aparecerán las siguientes preguntas



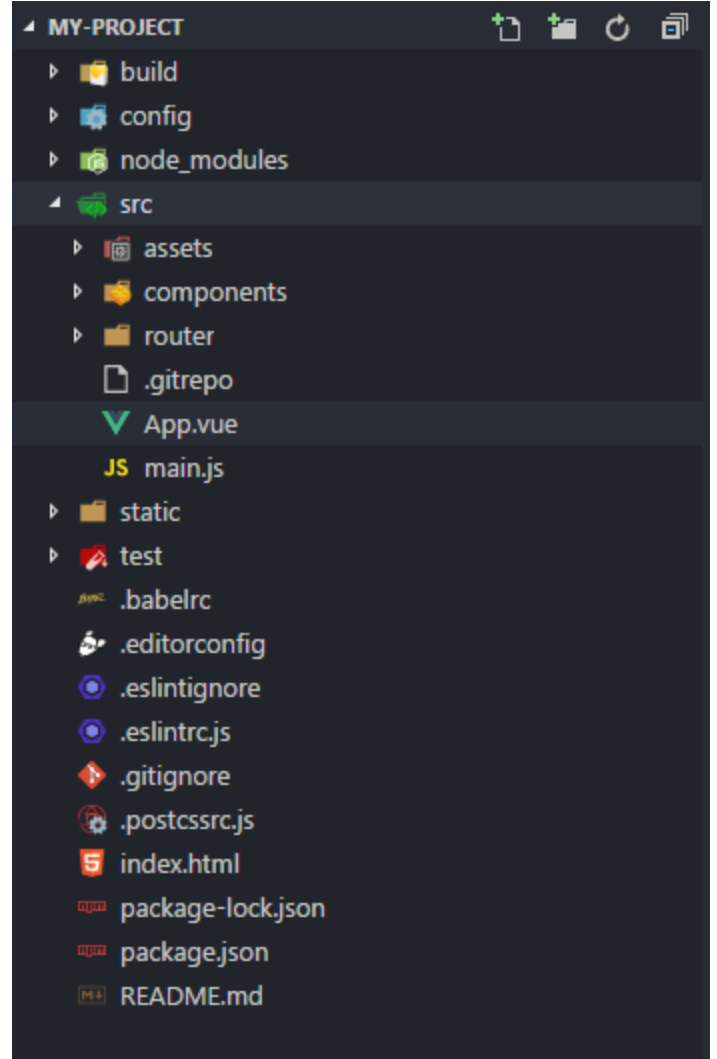
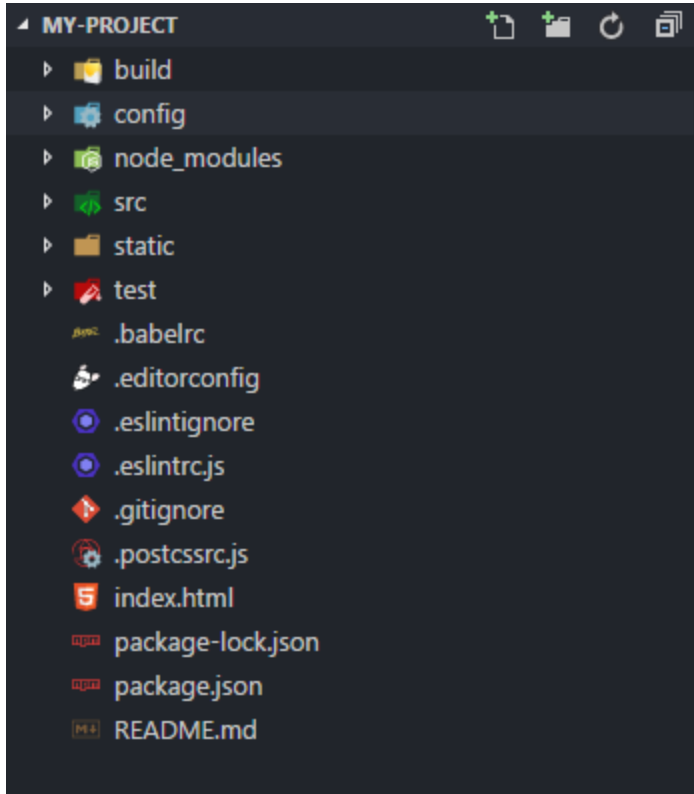
```
npm
Moscamlera@DESKTOP-M51QB9D MINGW64 ~/workspace
$ vue init vuetifyjs/webpack my-project

? Project name prueba-instalacion
? Project description First time
? Author fcojavierMT <fcojaviermtalavera@gmail.com>
? Vue build standalone
? Install vue-router? Yes
? Use ESLint to lint your code? Yes
? Pick an ESLint preset Standard
? Set up unit tests Yes
? Pick a test runner karma
? Setup e2e tests with Nightwatch? Yes
? Use a-la-carte components? Yes
? Use custom theme? No
? Should we run `npm install` for you after the project has been created? (recommended) npm

vue-cli · Generated "my-project".

# Installing project dependencies ...
# =====
[.....] / fetchMetadata: sill pacote range manifest for js-yaml@~3.7.0 fetched in 20ms
```

Estructura que nos crea el comando de Vue-cli.



Se podrían haber seguido diferentes estructuras, como la que ofrece Vuex, a base de contenedores, y componentes, pero para no complicarlo demasiado he decido seguir esta estructura.

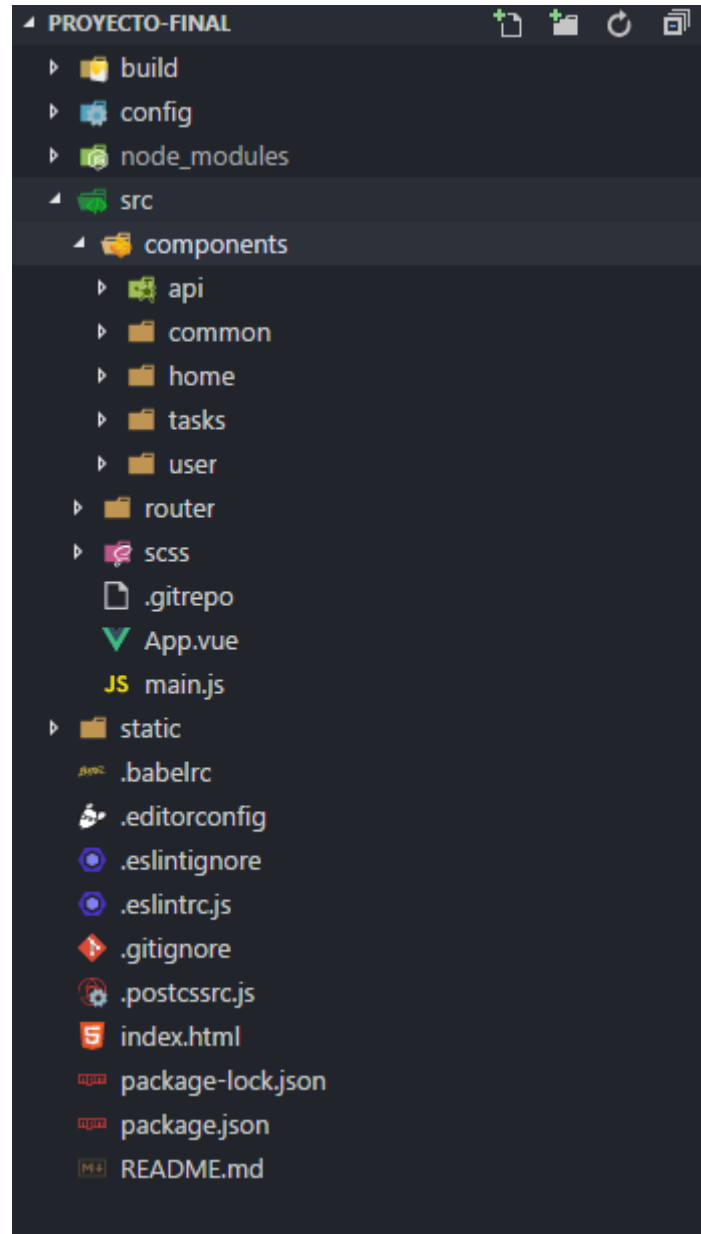
La documentación oficial también trae consigo el uso de una carpeta que se llama 'Store', que básicamente se usa para recoger eventos globales, en mi caso al ser una aplicación más sencilla, no utilizamos la 'store' como tal, sino que usamos eventos como el que nos ofrece JavaScript.

`this.$emit('nombre')` <= El comando para lanzar el evento hacia arriba

Estos lo veremos con más profundidad más adelante

```
.
├─ build/                # webpack config files
│  └─ ...
├─ config/
│  ├─ index.js           # main project config
│  └─ ...
├─ src/
│  ├─ main.js            # app entry file
│  ├─ App.vue            # main app component
│  ├─ components/        # ui components
│  │  └─ ...
│  └─ assets/            # module assets (processed by webpack)
│     └─ ...
└─ static/              # pure static assets (directly copied)
```

## Estructura final del proyecto



La carpeta '**api**' es donde se guarda la conexión propia de **firebase**, la cual luego declaramos en el **main.js**. Y ahora, ¿Qué es **main.js**? Este archivo lo genera **WebPack** y es el primero en ejecutarse, este archivo es esencial de comprender ya que es el que vamos a ejecutar, es el primer 'script'.

Aquí lo que haremos es importar todos los componentes que vayamos a utilizar de manera global y sin necesidad de importarles en cada archivo de manera individual, también cargaremos todos los componentes de la librería de **Vuetify** que nos interese utilizar, **incluyendo** la **conexión** a la base de datos.

```
12
13 import {
14   Vuetify,
15   VApp,
16   VNavigationDrawer,
17   VFooter,
18   VList,
19   VBtn,
20   VIcon,
21   VGrid,
22   VToolbar,
23   transitions,
24   VCard,
25   VForm,
26   VTextField,
27   VParallax,
28   VAlert,
29   VDialog,
30   VSelect,
31   VChip,
32   VAvatar
33 } from 'vuetify'
```

Al **importarlos** lo primero que tenemos que hacer es incluirlos para que Vue los pueda utilizar, en este caso utilizaremos la sintaxis que nos ofrece `Vue.use()`

```
3 import Vue from 'vue'
4 import App from './App'
5 import router from './router'
6 import Loading from './components/common/loading.vue'
7 import Advise from './components/common/advise.vue'
8 import NewTask from './components/tasks/newtask.vue'
9 import TaskCard from './components/tasks/taskCard.vue'
10 import UserCard from './components/user/userCard.vue'
11 import { db } from './components/api/firebaseInit'
```

```
37 Vue.use(Vuetify, {
38   components: {
39     VApp,
40     VNavigationDrawer,
41     VFooter,
42     VList,
43     VBtn,
44     VIcon,
45     VGrid,
46     VToolbar,
47     transitions,
48     VCard,
49     VForm,
50     VTextField,
51     VParallax,
52     VAlert,
53     VDialog,
54     VSelect,
55     VChip,
56     VAvatar
57   }
58 },
59 db)
```

Luego se crearán los diferentes componentes con la sintaxis de

`Vue.component('nombre-etiqueta', 'nombre-componente')`

```

60
61   Vue.config.productionTip = false
62   Vue.use(router)
63   Vue.component('loading-component', Loading)
64   Vue.component('advise-component', Advise)
65   Vue.component('newTask-component', NewTask)
66   Vue.component('task-card', TaskCard)
67   Vue.component('user-card', UserCard)
68
69   /* eslint-disable no-new */
70   new Vue({
71     el: '#app',
72     router,
73     components: { App },
74     template: '<App/>'
75   })
76

```

Todo lo que vamos a utilizar se va a **renderizar** en la etiqueta con el nombre **"App"**, el cual está en un **contenedor** div, por eso al final de todo el script de 'main.js' encontramos la primera instancia de Vue, que es donde carga el primer componente de Vue.js

```

index.html
1  <!DOCTYPE html>
2  <html>
3    <head>
4      <meta charset="utf-8">
5      <meta name="viewport" content="width=device-width,initial-scale=1.0">
6      <title>Stick a note</title>
7      <link href='./static/font-family.css' rel="stylesheet">
8      <link rel="icon" type="image/png" href="static/favicon.png">
9    </head>
10   <body>
11     <div id="app"></div>
12     <!-- Aquí es donde los archivos se renderizarán -->
13   </body>
14 </html>
15

```



En el archivo App.js se encuentra el **layout** principal de toda la aplicación, es decir el **menú principal**, el **menú lateral**, etc... que es el que se va a ver durante toda la aplicación. El código HTML es demasiado largo para enseñarlo, pero básicamente se utilizan los componentes que nos ofrece **Vuetify** para crear los diferentes **menus**. Para ello se ha iterado sobre un array llamado `menuItems = []` que se crea en el la etiqueta `<script>`.

Lo que utilizamos en la función `data()`, son los diferentes atributos que le asignamos al componente Vue.

```
61 <script>
62 import firebase from 'firebase'
63
64 export default {
65   data () {
66     return {
67       sideNav: false,
68       menuLoggedIn: false,
69       userIsConnected: false,
70       userEmail: '',
71       menuItems: []
72     }
73   }
74 }
```

```
39 <v-toolbar-items class='hidden-sm-and-down' v-for='item in menuItems' :key='item.title'>
40   <v-btn flat :to='item.link'>
41     <v-icon left>{{ item.icon }}</v-icon>
42     <span class="title-navbar">{{item.title}}</span>
43   </v-btn>
44 </v-toolbar-items>
```

En este caso utilizamos la sintaxis que nos ofrece Vue.js de **v-for**, que nos sirve para recorrer el **array** de **menuItems** = [], el cual tenemos declarado abajo en el script. El cuál lo rellenamos según el estado del usuario, ya este registrado o no.

```

4  computed: {
5    isLoggedIn: function () {
6      firebase.auth().onAuthStateChanged((user) => {
7        if (user) {
8          this.userIsConnected = true
9          this.userEmail = user.email
10         this.menuItems = [
11           {
12             icon: 'home',
13             title: 'Inicio',
14             link: '/home'
15           },
16           {
17             icon: 'account_circle',
18             title: 'Perfil',
19             link: '/user'
20           },
21           {
22             icon: 'library_books',
23             title: 'Mis tareas',
24             link: '/tasks'
25           },
26           {
27             icon: 'exit_to_app',
28             title: 'Salir',
29             link: '/logout'
30           }
31         ]
32       })
33     }
34   }
35 }

```

```

102   } else {
103     this.userIsConnected = false
104     this.userEmail = 'Conectate'
105     this.menuItems = [
106       {
107         icon: 'home',
108         title: 'Inicio',
109         link: '/home'
110       },
111       {
112         icon: 'group_add',
113         title: 'Registrarse',
114         link: '/signUp'
115       },
116       {
117         icon: 'lock_open',
118         title: 'Iniciar Sesión',
119         link: '/signIn'
120       }
121     ]
122   }
123 })
124 }
125 },
126 name: 'App'
127 }
128 </script>

```

Luego simplemente lo recorremos en el template y lo mostramos en el menú principal, cargamos el título, el link y su correspondiente icono.

```

39  <v-toolbar-items class='hidden-sm-and-down' v-for='item in menuItems' :key='item.title'>
40    <v-btn flat :to='item.link'>
41      <v-icon left>{{ item.icon }}</v-icon>
42      <span class="title-navbar">{{item.title}}</span>
43    </v-btn>
44  </v-toolbar-items>

```

Cada uno de los links carga un componente principal, este se verá reflejado en la pantalla, para ello Vue ofrece un archivo llamado routes, el cuál en nuestro caso nos lo crea el comando que genera la estructura principal del proyecto. Entonces aquí exportamos las rutas que tendrá nuestro proyecto.

```
JS index.js x
1  import Vue from 'vue'
2  import Router from 'vue-router'
3  import Home from '@/components/home/home'
4  import User from '@/components/user/user'
5  import Tasks from '@/components/tasks/tasks'
6  import SignIn from '@/components/user/signIn'
7  import SignUp from '@/components/user/signUp'
8  import Logout from '@/components/user/logout'
9
10 Vue.use(Router)
11
12 export default new Router({
13   routes: [
14     {
15       path: '/',
16       name: 'home',
17       component: Home
18     },
19     {
20       path: '*',
21       redirect: '/'
22     },
23     {
24       path: '/user',
25       name: 'user',
26       component: User
27     },
28     {
29       path: '/tasks',
30       name: 'tasks',
31       component: Tasks,
32       meta: {
33         requiresAuth: true
34       }
35     },
36     {
37       path: '/signIn',
38       name: 'signIn',
39       component: SignIn
40     },
41     {
42       path: '/signUp',
43       name: 'signUp',
44       component: SignUp
45     },
46   ],
47 })
```

Si el usuario introdujera otra ruta que no estuviera registrada sería redirigido a la página principal.

Para registrar un usuario en la aplicación firebase ofrece una función llamada, `createUserWithEmailAndPassword(email, contraseña)`

```
70      signUp: function () {
71          if (!this.comparePasswords()) {
72              this.buttonDisabled = true
73              firebase.auth().createUserWithEmailAndPassword(this.email, this.password).then(
74                  (user) => {
75                      this.alertSuccess = true
76                      this.alertError = false
77                      user.sendEmailVerification()
78                      setTimeout((user) => {
79                          |   this.$router.replace(['home'])
80                      }, 3000)
81                  },
82                  (error) => {
83                      this.alertError = true
84                      this.buttonDisabled = false
85                      this.errorMessage = error.message
86                  }
87              )
88          } else {
89              this.buttonDisabled = false
90          }
91      }
```

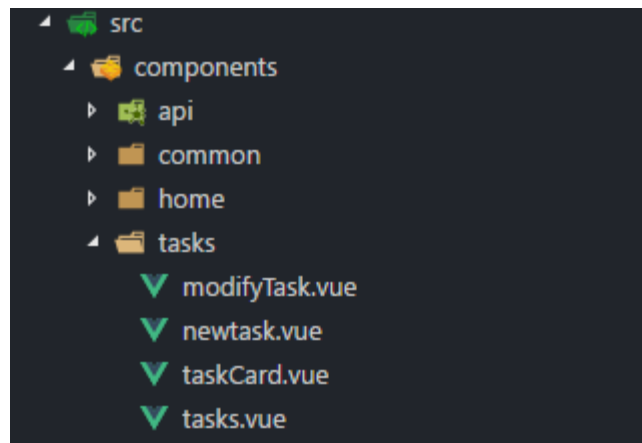
Lo mismo cuando el usuario quiere logearse en nuestra aplicación para ello, simplemente utilizamos otro método que nos ofrece firebase llamado `signInWithEmailAndPassword(email, contraseña)`

Para obtener el id del usuario actual, firebase ofrece una función llamada `onAuthStateChanged()` la cual nos permite añadir un **listener**, para detectar los **cambios** que ha realizado el usuario, en nuestro caso si está conectado o desconectado.

Evidentemente si queremos utilizar estas funcionalidades como queramos tenemos que importar firebase en el archivo que vayamos a utilizar estos métodos, en este caso sería `import firebase from 'firebase'`

En cuanto a los componentes hay una gran diferenciación, los componentes tontos, que lo único que hacen es pintar y mandar eventos, y luego un componente más grande que se ocupe de la lógica de enviar los datos a firebase.

En este caso el componente más grande es 'tasks'.



Los componentes tontos en este caso son, 'newTask', 'modifyTask', y la carta de la tarea como tal.

Por ejemplo 'taskCard', simplemente recoge parámetros de manera que lo único que le pasamos a este componente es el array de tareas que nos llega desde firebase. Esto se lo indicamos por la propiedad 'props', luego simplemente se lo pasamos al componente a través de un '**bindeo**', para que dinámicamente alternemos clases

```
22 <script>
23 export default {
24   props: ['tasks'],
25   data () {
26     return {
27     }
28   },
```

```
<task-card v-bind:tasks="myTasks"></task-card>
```

myTasks es el **array** de tareas que recibimos de la base de datos

El componente task-card lo único que hace es enviar el evento hacia arriba en este caso lo recibiría el componente 'tasks', como he comentado antes, utilizaremos la función que nos ofrece JavaScript para poder mandar el evento hacia 'arriba'.

```
37   deleteTask: function (taskId) {  
38     this.$emit('delete-task', taskId)  
39   },
```

En este ejemplo mandamos el evento hacia arriba con el id de la tarea para poder borrar la tarea en cuestión.

```
6 | <task-card v-on:delete-task="removeTask"></task-card>
```

Luego simplemente en el componente usamos un **listener** para escuchar si el evento llego correctamente, para ello utilizamos la sintaxis que nos ofrece Vue `v-on:nombre-evento="función"`

Después de esto simplemente en el componente superior, en nuestro caso 'tasks'.

```
48   removeTask: function (taskId) {  
49     taskReference.child(taskId).remove()  
50     nativeToast({  
51       message: 'Tarea eliminada!',  
52       position: 'bottom',  
53       timeout: 3000,  
54       type: 'error'  
55     })  
56   },
```

'**taskReference**' es la referencia que cogemos de firebase para poder subir la información a la base de datos.

### 3.2.4.-Pruebas

Para probar todo el código de esta aplicación simplemente se ha seguido el tradicional uso de prueba y error, de hecho, en la valoración personal de más abajo destaco la idea de haber metido test a modo de 'probar y testear' el código de manera más profesional, pero debido a la falta de tiempo no se ha podido implementar con tiempo.

También he intentado no solamente en probarla yo como desarrollador, si no personas ajenas a mi alrededor. Para que comprueben todo lo que puedan de la aplicación, familias, amigos, etc... Algo que me ha servido de mucho, porque han encontrado errores que yo no he encontrado.

Lo único parecido a las pruebas es el testeo del código, para ello he utilizado ESLint para poder seguir un tipo de estructura a la hora de programar de manera uniforme, que me indique los errores semánticos y estructurales.





STICK A NOTE!

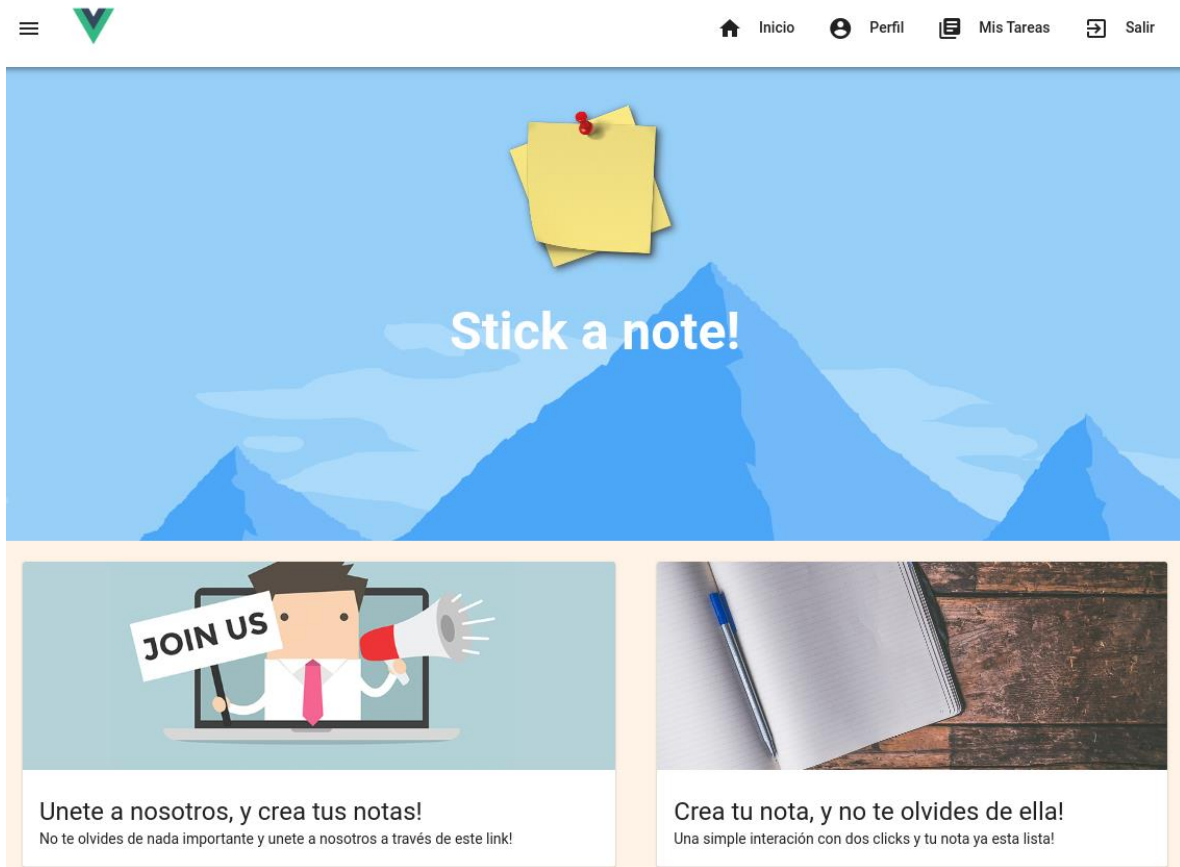
# MANUAL DE USUARIO

CREA TUS PEQUEÑAS NOTAS



## 4.-Manual de uso

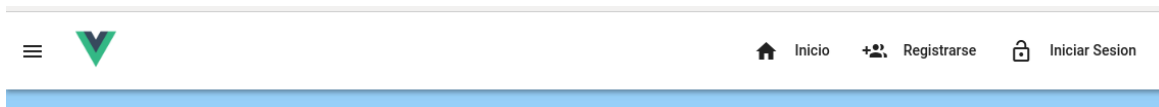
Lo primero que obtenemos al entrar es una página de bienvenida (Landing page), esta página equivale a la pestaña de Inicio



Una página sencilla de bienvenida, luego en la parte superior derecha tenemos los botones de navegación principales, los cuales son: Inicio, Perfil, Mis Tareas y Salir.

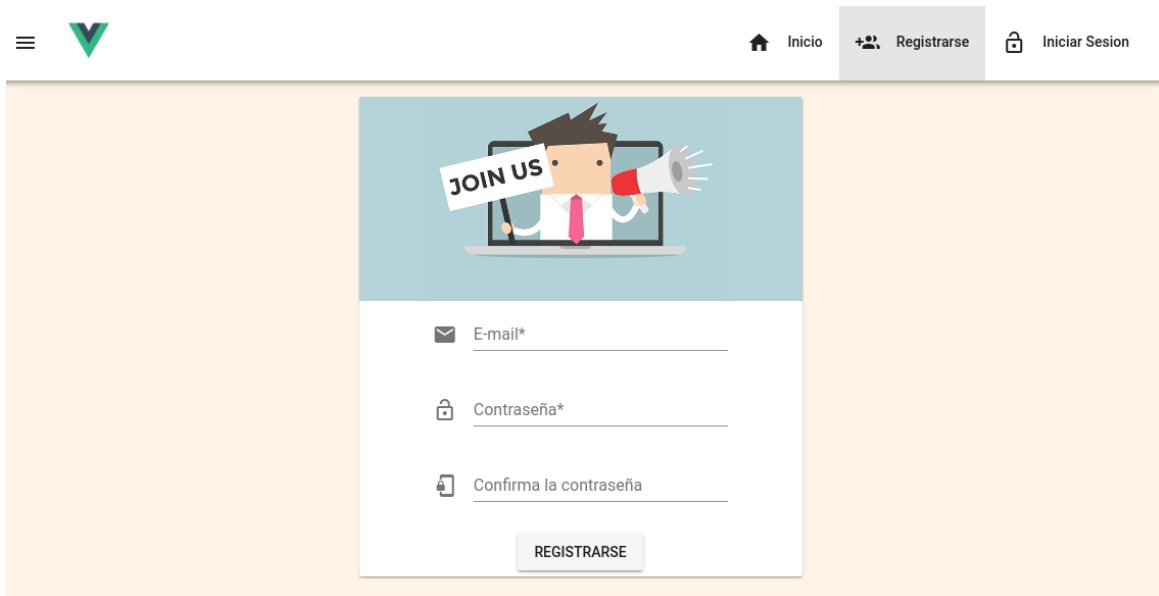


Si el usuario estuviera registrado en la aplicación, si no lo estuviera saldría este menú

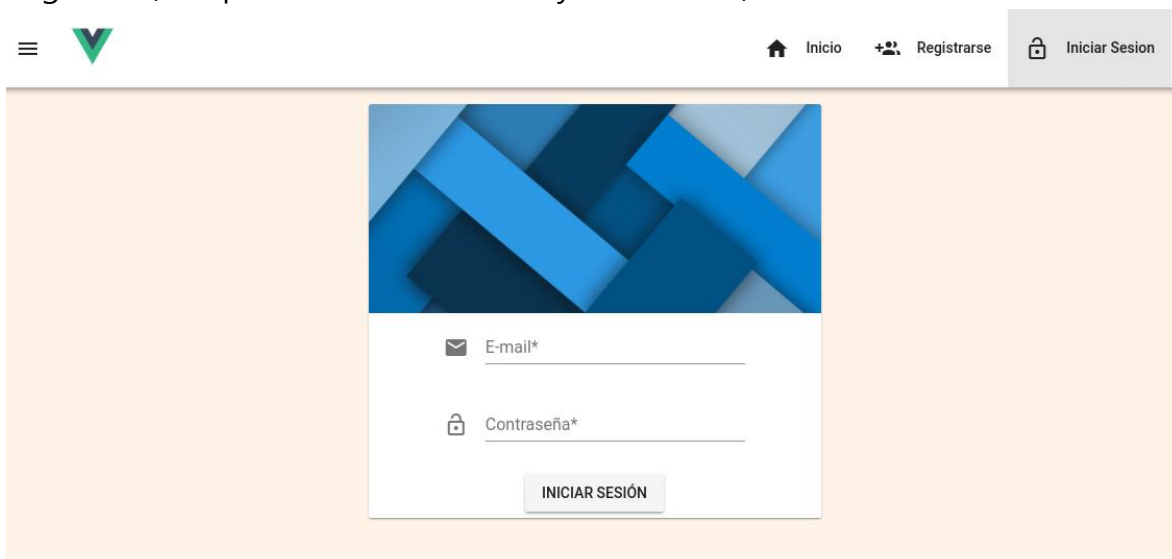


El usuario puede registrarse o logearse si ya posee una cuenta.


Registrarse (Campos necesarios: E-mail, contraseña y confirmación)




Logearse (Campos necesarios: E-mail y contraseña)




Si la cuenta no existe u el email está mal formado entonces la página mostrara una alerta que indica el tipo de error que se trata. El error viene directamente de firebase por lo que el control de errores y la información que proviene de ella es la que mostramos directamente en el mensaje de error sin necesidad de tener nosotros el control.




E-mail\*

 emailvalido@gmail.com

Contraseña\*


 .....

Confirma la contraseña

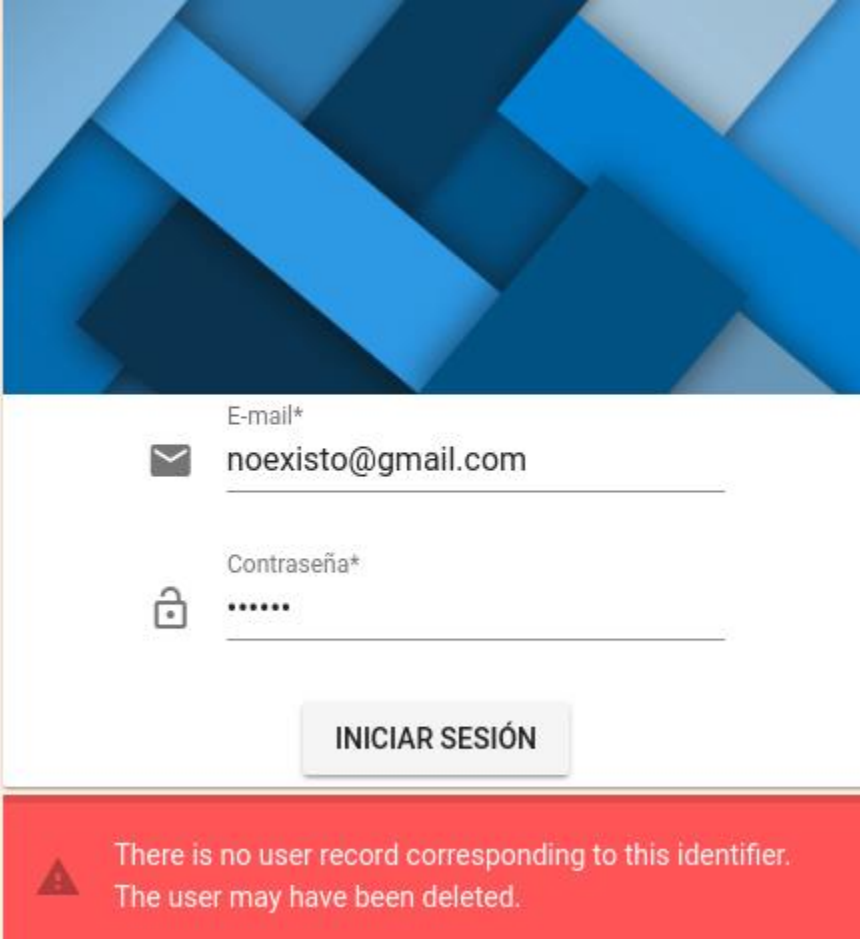
 .....

Las contraseñas no coinciden

**REGISTRARSE**

 The email address is badly formatted.

Lo mismo pasa cuando la confirmación de la contraseña no es válida, no deja enviar la solicitud de registro. Todos los errores los controla firebase por lo que nosotros solo nos ocupamos de la validación a nivel básico, comprobar email, ver si las contraseñas son iguales, etc. Esto es exactamente igual en la pantalla de login



E-mail\*

noexisto@gmail.com

Contraseña\*

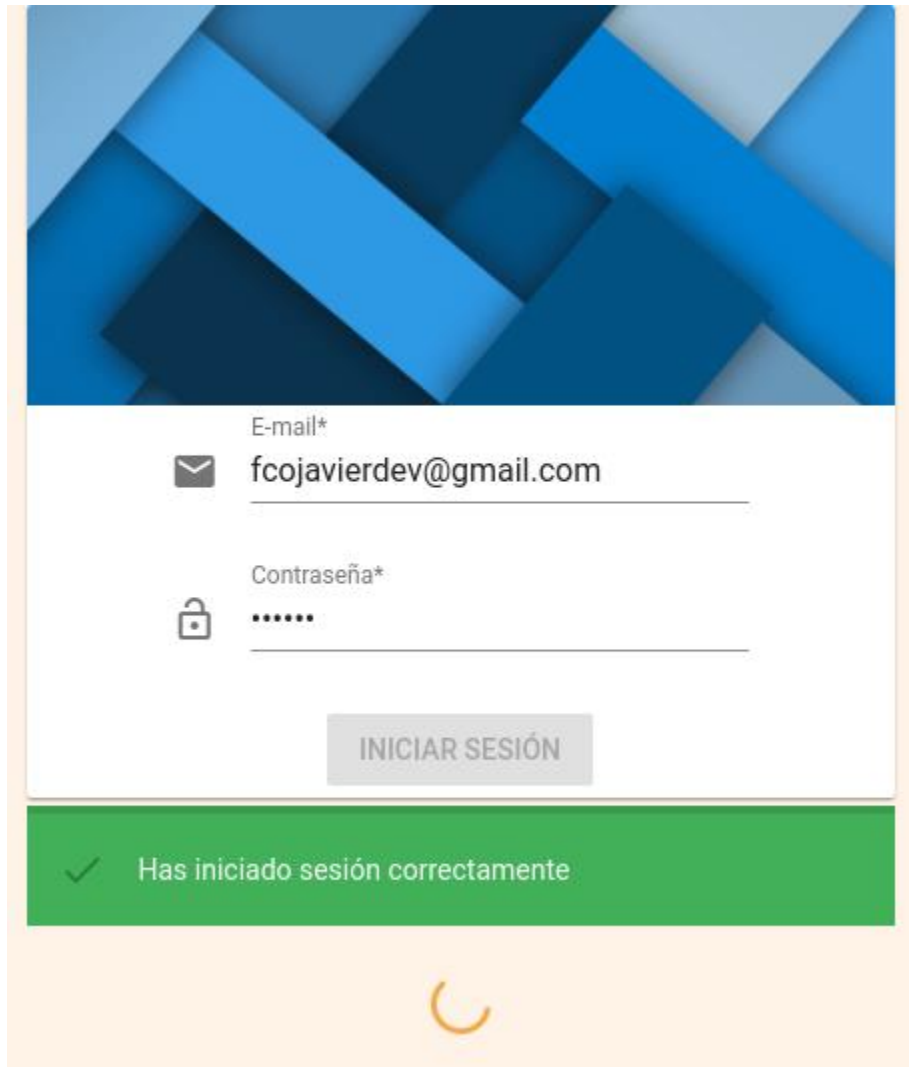
.....

INICIAR SESIÓN

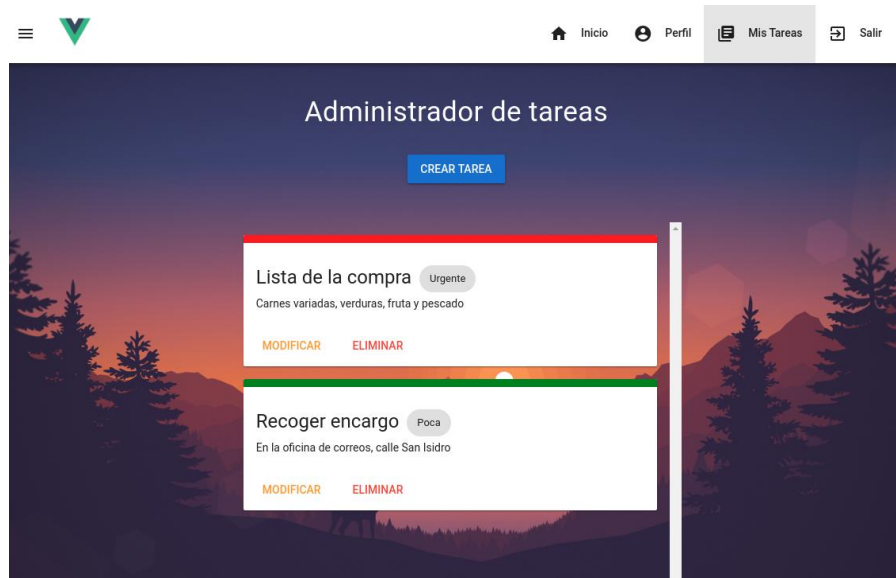
⚠ There is no user record corresponding to this identifier.  
The user may have been deleted.

Si la cuenta existe u el registro ha sido un éxito entonces el usuario es redirigido a la página /home, es decir la primera página que hemos visto, la landing page principal. Este mensaje saldría si la cuenta existe o el registro ha sido correcto.

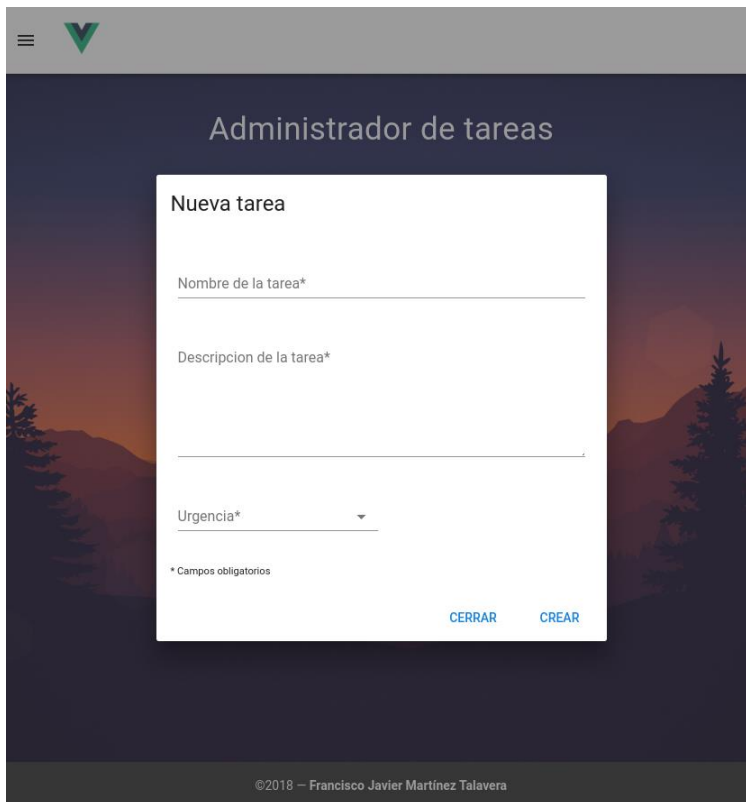
Tanto el login como el registrarse tienen las mismas comprobaciones por parte de firebase por lo que no hace falta mostrar todos los tipos de errores.



Y ahora nos vamos al núcleo principal de nuestra aplicación, las **tareas**.



Para crear las notas, simplemente hay que hacer click en el botton "Crear Tarea", la cual desplegará el siguiente modal



**Nueva tarea**

Nombre de la tarea\*

Descripción de la tarea\*

Urgencia\*

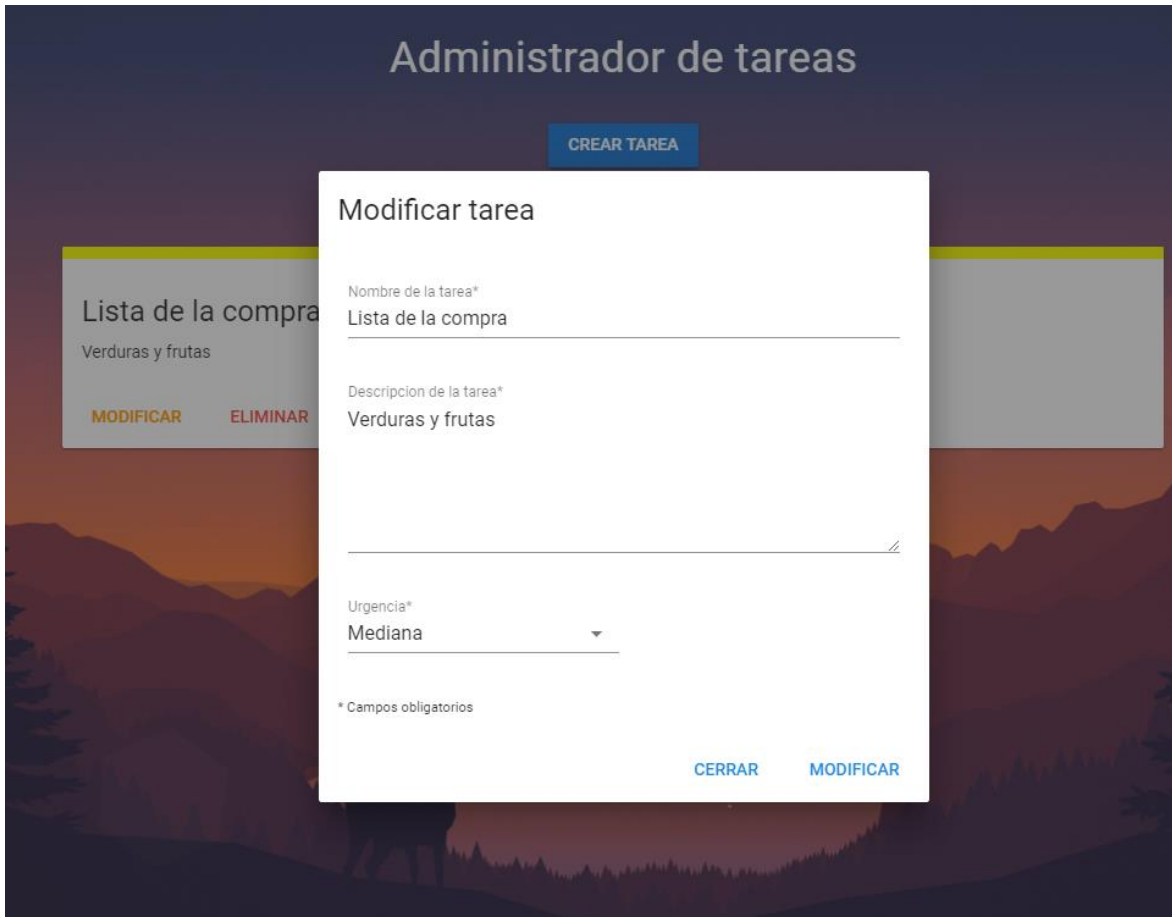
\* Campos obligatorios

CERRAR CREAR

©2018 – Francisco Javier Martínez Talavera

La modal nos pedirá los siguientes datos: nombre de la tarea, descripción de la tarea y la urgencia (poca, mediana, urgente), si alguno de los campos no se llega a rellenar, saltará un error.

Para modificar ocurre lo mismo, se muestra en pantalla una modal, la cual se muestra con los datos de la tarea a modificar en las pantallas.



Administrador de tareas

CREAR TAREA

**Modificar tarea**

Nombre de la tarea\*  
Lista de la compra

Descripcion de la tarea\*  
Verduras y frutas

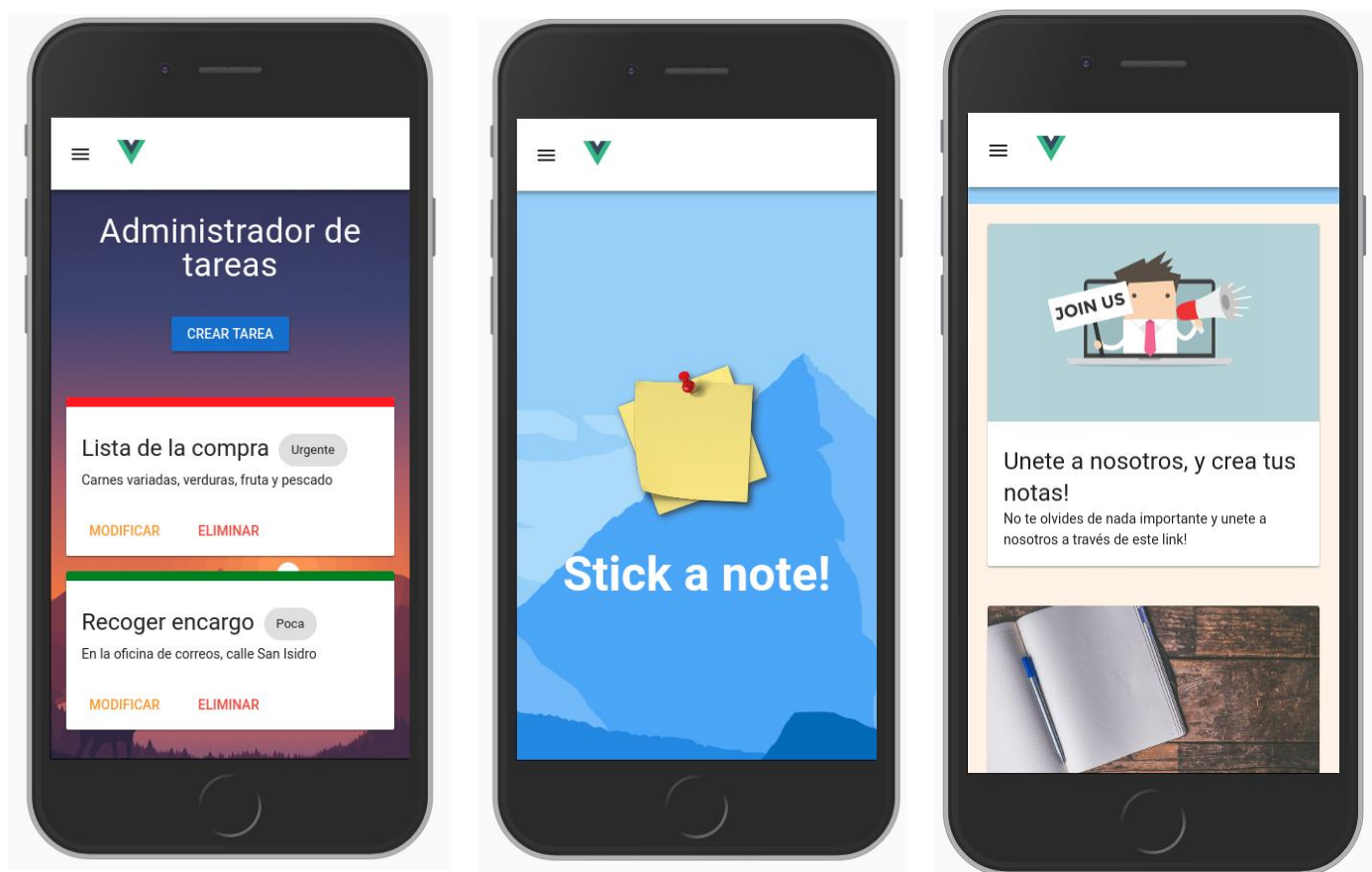
Urgencia\*  
Mediana

\* Campos obligatorios

CERRAR MODIFICAR

Listado de la compra  
Verduras y frutas  
MODIFICAR ELIMINAR

Algunos ejemplos de la aplicación en formato **móvil**





## 5.-Valoración personal

En este caso el proyecto ha sido bastante instructivo, y he aprendido un bastante sobre los diferentes bundles en mi caso WebPack. También aparte de aprender el nuevo framework de Vue.js e indagado más en cuanto al lenguaje de JavaScript, sobre todo con la nueva sintaxis (ES6..), también he aprendido mucho sobre Css, ya no solo en este proyecto si no ya en la empresa, debido a los diversos proyecto de maquetación que me han destinado a hacer, incluyendo todo lo que envuelve al desarrollo web.

En cuanto a lo negativo del proyecto me gustaría destacar la mala documentación que posee vuetify, si bien ofrece muchos componentes, muchos de estos no son muy personalizables, lo que provoca que la customización baje.

Lo que evidentemente siempre se puede mejorar es el diseño y el código, de todas maneras, lo que me hubiera gustado añadir sería más tema de testing, ya que prácticamente no he visto demasiado.

Para este proyecto también me hubiera gustado realizar más implementaciones y abarcar más opciones que ofrece Vue.js como framework. Pero debido al poco tiempo de desarrollo (Menos de un mes), se han quedado varias cosas en el tintero, como por ejemplo añadir más información a los usuarios, no solamente el correo y el email verificado que son opciones que ofrece firebase por defecto, o como tratar los eventos con un plugin exterior (Vuex)

Lo más difícil ha sido hacer todo lo más correcto posible ya que el proyecto principal que tenía en mente no es ni por asomo parecido al resultado final, algo completamente normal, ya que en las prácticas he pedido consejo de cómo hacer la estructura y hacer el mejor diseño posible.

Lo que mejoraría del ciclo, sería básicamente la gestión del tiempo, ya que todos lo vemos se hace muy deprisa y no queda prácticamente tiempo a adentrarse en todas las posibilidades que nos ofrece, también es cierto que es prácticamente imposible darlo todo en el ciclo, debido a la gran cantidad de temario, lo que si que me centraría más es en uso de Javascript como tal y no tanto en JQuery.

También me hubiera gustado indagar más en el tema de Angular, ya que nos quedamos a las puertas de un gran framework, debido al poco tiempo que teníamos, a cambio recibimos un mini-seminario de React, que estuvo bastante bien pero realmente nos quedamos a las puertas de lo que podría ofrecernos estos dos frameworks.

Aun teniendo poco tiempo para según que cosas en el ciclo, considero que he aprendido una gran cantidad de conceptos, sobre todo a la hora de las tecnologías web.

## 6. Fuentes bibliográficas

- Página oficial de Vue
  - <https://vuejs.org/>
  - <https://vuejs.org/v2/guide/>
- Instalación usando node.js y NPM
  - <https://vuejs-templates.github.io/webpack/>
  - <https://vuejs.org/v2/guide/installation.html>
- Página oficial de Vuetify
  - <https://vuetifyjs.com/en/>
- Página oficial de Firebase
  - <https://firebase.google.com/>
- Información adicional sobre Vue
  - <https://carlosazaustre.es/que-es-lo-que-me-gusta-de-vue-js/>
- WebPack y el uso de SASS
  - <https://medium.com/hong-kong-tech/use-sass-scss-of-webpack-in-vuejs-8dde3a83611e>
  - <https://github.com/shakacode/sass-resources-loader>

- Estructura de carpetas
  - <http://vuejs-templates.github.io/webpack/structure.html>
  - <https://vuex.vuejs.org/en/structure.html>
- Tutoriales interesantes sobre Vue
  - [https://www.youtube.com/watch?v=R7\\_QcqsXshw](https://www.youtube.com/watch?v=R7_QcqsXshw)
  - <https://medium.com/@anas.mammeri/vue-2-firebase-how-to-build-a-vue-app-with-firebase-authentication-system-in-15-minutes-fdce6f289c3c>
  - <https://medium.com/codingthesmartway-com-blog/vue-js-2-firebase-e4b2479e35a8>
- VueSchool [Página para aprender a usar Vue]
  - <https://vueschool.io/>
  - <https://vueschool.io/courses/vuejs-firebase-realtime-database>