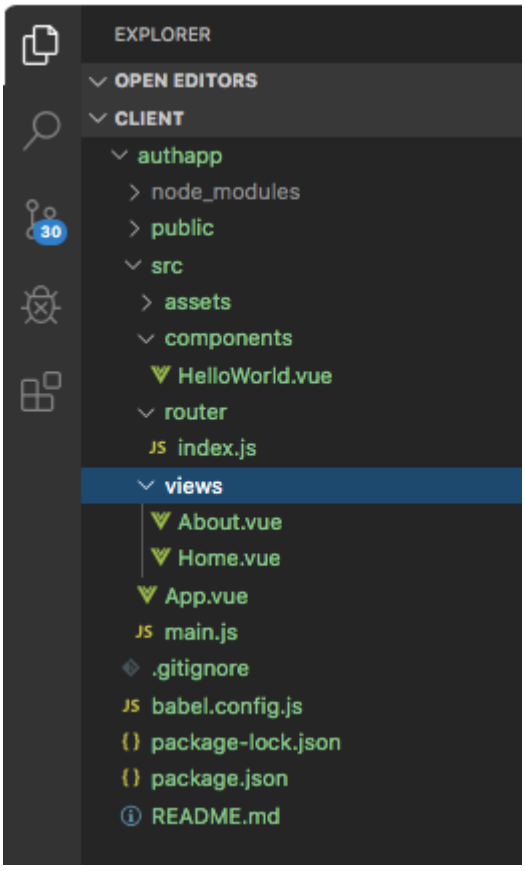
Frontend Inicio de sesión

Objetivos del momento:
Crear un nuevo proyecto de Vuejs con vuejs Cli.
Instalación y configuración de paquetes.
Crear la interfaz de usuario (UI) para nuestra aplicación de inicio de sesión.

Crear un nuevo proyecto de Vuejs con vuejs Cli.
Desarrollar todos los pasos de la sesión anterior para la inicialización de un proyecto en vue-cli hasta obtener la siguiente estructura



Para saber que todo va bien inicializamos el servidor con el comando: **npm run serve**.

Instalación y configuración de paquetes.

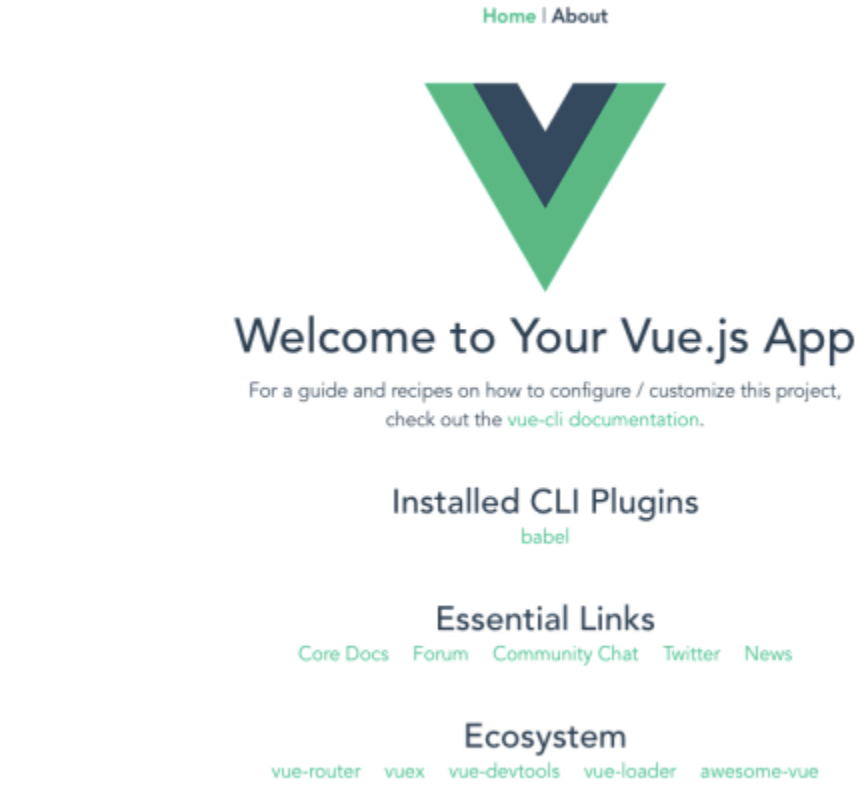
Es el momento de instalar algunos paquetes para nuestra frontend.para eso se instalará vue-jwt-decode, bootstrap, sweetalerty axios. Así que abramos nuestra terminal y escriba el siguiente comando:

```
C:\Users\andre>npm i vue-jwt-decode axios bootstrap sweetalert
```

vue-jwt-decode : Este es un decodificador JWT para Vuejs.
Bootstrap : Este es un marco CSS para desarrollar sitios web receptivos y móviles.
sweetalert: Este es un reemplazo hermoso, receptivo, personalizable y accesible para los cuadros emergentes de JavaScript.
Después de instalar estos paquetes, debemos configurarlos en nuestro archivo /src/main.js. El /src/main.js archivo debería verse así:

```
import Vue from "vue";
import App from "./App.vue";
import router from "./router";
import store from "./store";
import "bootstrap/dist/css/bootstrap.css";
import axios from "axios";
const base = axios.create({
    baseURL: "http://localhost:3000"
});
Vue.prototype.$http = base;

Vue.config.productionTip = false;

new Vue({
    router,
    store,
    render: h => h(App)
}).$mount("#app");
```

Cree la interfaz de usuario (UI) para nuestra aplicación
Por defecto, nuestra aplicación debería verse así:



Necesitamos limpiar la interfaz de usuario. Vayamos a nuestra carpeta src / components y eliminemos el archivo HelloWorld.vue. Después de eliminarlo, deberíamos obtener un error en nuestra consola. Para corregir ese error, creamos un nuevo componente en la carpeta components llamado home.vue que tendrá el siguiente aspecto:

Home.vue

```
<template>
  <div>
    <nav class="navbar navbar-expand-lg navbar-dark bg-dark">
      <div class="container">
        <a class="navbar-brand" href="#">Navbar</a>
        <button
          class="navbar-toggler"
          type="button"
          data-toggle="collapse"
          data-target="#navbarNav"
          aria-controls="navbarNav"
          aria-expanded="false"
          aria-label="Toggle navigation"
        >
          <span class="navbar-toggler-icon"></span>
```

```
        </button>
        <div
          class="collapse navbar-collapse justify-content-end"
          id="navbarNav"
        >
          <ul class="navbar-nav">
            <li class="nav-item active">
              <a class="nav-link" @click="logUserOut"> Logout</a>
            </li>
          </ul>
        </div>
      </div>
    </nav>
    <section>
      <div class="container mt-5">
        <div class="row">
          <div class="col-md-12">
            <ul class="list-group">
              <li class="list-group-item">Name : {{ user.name }}</li>
              <li class="list-group-item">Email : {{ user.email }}</li>
            </ul>
          </div>
        </div>
      </div>
    </section>
  </div>
</template>
<script>
import VueJwtDecode from "vue-jwt-decode";
export default {
  data() {
    return {
      user: {}
    };
  },
  methods: {
    getUserDetails() {
      let token = localStorage.getItem("jwt");
      let decoded = VueJwtDecode.decode(token);
      console.log('hola decode',decoded);
      this.user = decoded;
    },
    logUserOut() {
      localStorage.removeItem("jwt");
      this.$router.push("/");
    }
  },
  created() {
    this.getUserDetails();
```

```
  }
};
</script>
<style scoped></style>
```

Estamos haciendo uso del paquete vue-jwt-decode. El paquete decodifica el token cifrado que obtenemos de nuestro almacenamiento local y devuelve los detalles del usuario y luego de eso, mostramos los detalles del usuario. También agregamos una función para cerrar la sesión de un usuario. Esto elimina el token de usuario del almacenamiento local y lo lleva de regreso a la página de inicio de sesión.
Luevo vamos a la carpta src/views/Home.vue y lo actualizamos con el siguiente fragmento de código:

```
<template>
  <div class="home">
    <img alt="Vue logo" src="../assets/logo.png" />
    <home />
  </div>
</template>

<script>
// @ is an alias to /src
import home from "@/components/home.vue";

export default {
  name: "Home",
  components: {
    home
  }
};
</script>
```

Simplemente se invocó el componente creado anteriormente.

Luego nos ubicamos src/App.vue y reemplace el código allí con los códigos siguientes:

```
<template>
  <div id="app">
    <router-view />
  </div>
</template>

<style>
#app {
  font-family: Avenir, Helvetica, Arial, sans-serif;
```

```
  -webkit-font-smoothing: antialiased;
  -moz-osx-font-smoothing: grayscale;
  text-align: center;
  color: #2c3e50;
}

#nav {
  padding: 30px;
}

#nav a {
  font-weight: bold;
  color: #2c3e50;
}

#nav a.router-link-exact-active {
  color: #42b983;
}
</style>
```

Ahora es el momento de crear nuestros componentes de login. En el src/components directorio crearemos un nuevo directorio llamado auth. En ese directorio de autenticación crearemos un archivos vue llamado: login.vue
Que tendrá el siguiente aspecto:

```
<template>
  <div class="container">
    <div class="row">
      <div class="col-lg-6 offset-lg-3 col-sm-10 offset-sm-1">
        <form
          class="text-center border border-primary p-5"
          style="margin-top:70px;height:auto;padding-top:100px !important;"
          @submit.prevent="loginUser"
        >
        <h1 class="h3 mb-3 font-weight-normal" style="text-
align: center"> Iniciar de sesión</h1>
          <input
            type="text"
            id="email"
            class="form-control mb-5"
            placeholder="Email"
            v-model="login.email"
          />
          <!-- Password -->
          <input
            type="password"
            id="password"
            class="form-control mb-5"
            placeholder="Contraseña"
```

```
            v-model="login.password"
          />
          <!-- inicio sesion button -->
          <center>
            <button class="btn btn-primary btn-block w-75 my-
4" type="submit">
              Inicio de sesion
            </button>
          </center>
        </form>
      </div>
    </div>
  </div>
</template>
<script>
import swal from "sweetalert";
export default {
  data() {
    return {
      login: {
        email: "",
        password: ""
      }
    };
  },
  methods: {
    async loginUser() {
        try {
        let response = await this.$http.post("/api/auth/signin", this.login)
;
        console.log(response.data);
        let token = response.data.accessToken;
        localStorage.setItem("jwt", token);
        if (token) {
          swal("Exitoso", "login exitoso", "success");
          this.$router.push("/home");
        }
      } catch (err) {
        swal("Error", "datos incorrectos", "error");
        console.log(err.response);
      }
    }
  }
};
</script>
```

En el hook data tenemos un objeto login compuesto de email y password,estarán vinculados por medio de la etiqueta v-model de los inputs del formulario , ejemplo:

```
<input
            type="password"
            id="password"
            class="form-control mb-5"
            placeholder="Contraseña"
            v-model="login.password"
          />
```

En el método **loginUser** verifica por medio de una llamada Ajax a la api del backend que tendrá la siguiente url **/api/auth/signin** (la construiremos en la siguiente sesión) que verifica si los detalles del usuario son correctos, si es correcto, lleva al usuario a la página de inicio donde se muestran los detalles del usuario; de lo contrario, arroja un error.
Pero esta aplicación aún no es muy segura. El usuario aún puede obtener acceso a la página de inicio incluso cuando no hay ningún token en el almacenamiento local del usuario. Así que arreglemos eso. Escribiremos un pequeño método en el archivo del enrutador vuejs que buscará usuarios autenticados. Así que nos ubicamos src/router/index.js reemplace el código con el siguiente código:

```
import Vue from "vue";
import VueRouter from "vue-router";
import Home from "../views/Home.vue";

Vue.use(VueRouter);

const routes = [{
        path: "/home",
        name: "home",
        component: Home,
        meta: {
            requiresAuth: true
        }
    },
    {
        path: "/",
        name: "login",
        component: () =>
            import ("../views/login.vue")
    },
];

const router = new VueRouter({
    mode: "history",
    base: process.env.BASE_URL,
    routes
});

router.beforeEach((to, from, next) => {
```

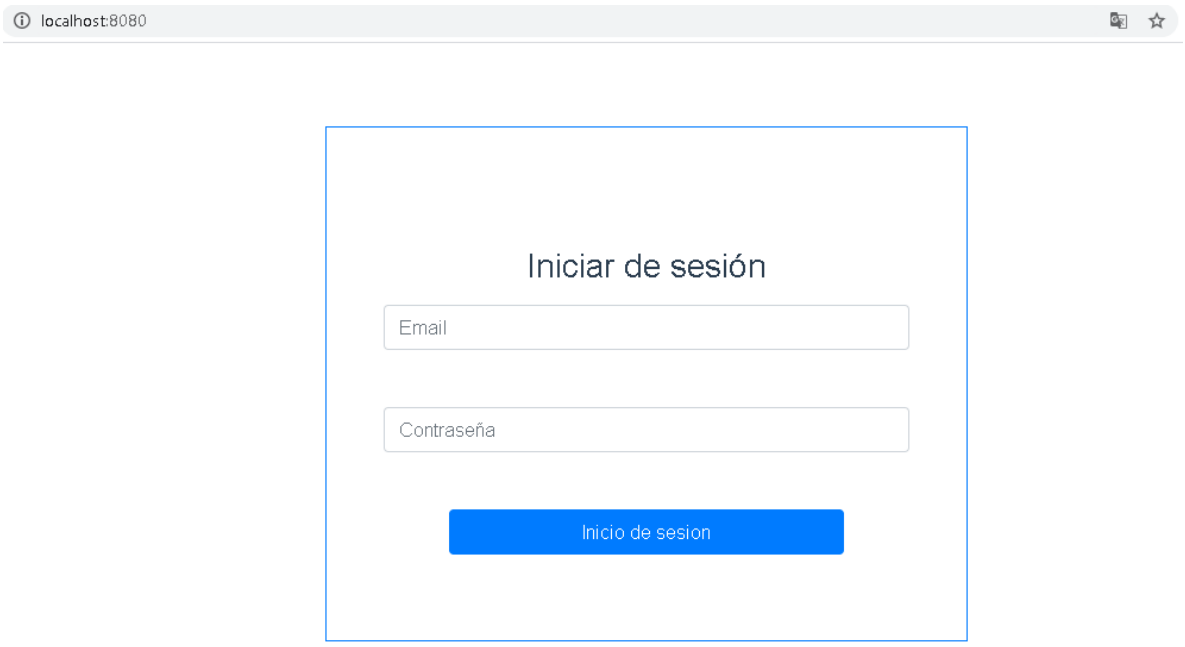
```
    if (to.matched.some(record => record.meta.requiresAuth)) {
        if (localStorage.getItem("jwt") == null) {
            next({
                path: "/"
            });
        } else {
            next();
        }
    } else {
        next();
    }
});

export default router;
```

a las rutas que solo podrán acceder con autenticación le agregaremos un nuevo parámetro llamado meta :

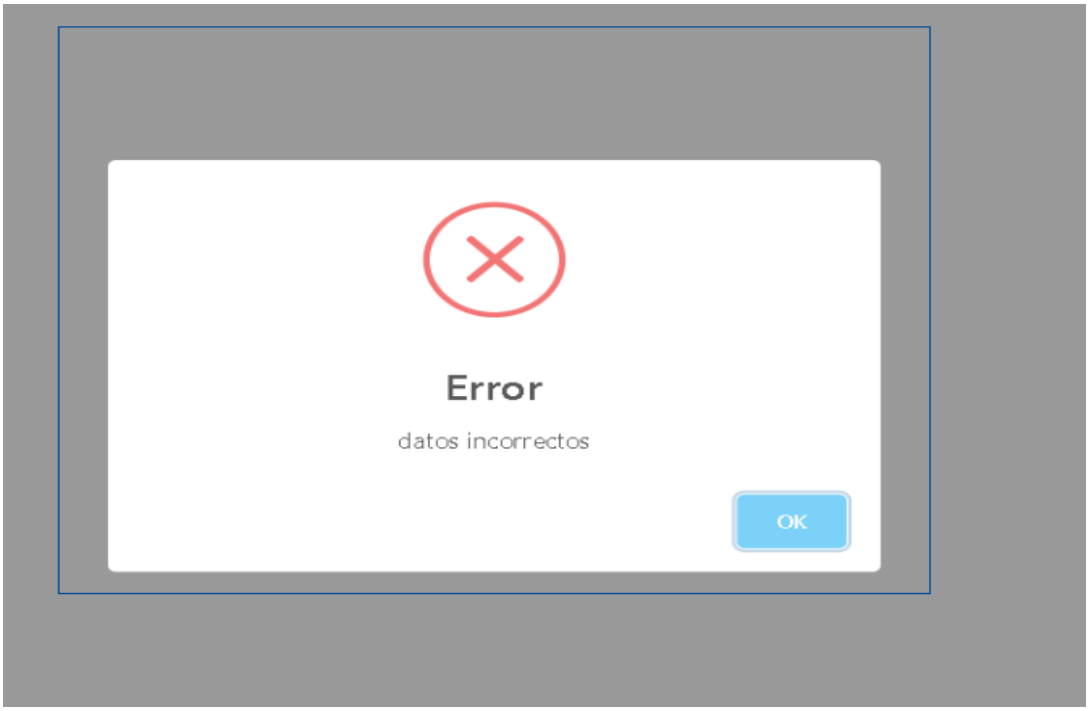
```
meta: {
            requiresAuth: true
        }
```

En este caso la ruta /home tendrá este parámetro. Cuando se llama a la ruta de inicio, llama a un método **`router.beforeEach`** que verifica si es un token en el almacenamiento local del usuario, si hay alguno, lleva al usuario a la página de inicio, si el token es nulo, redirige al usuario de nuevo a la página de inicio de sesión . Con este método, un usuario no puede acceder a la ruta local si no está autenticado.

Nuestra plataforma web tendrá el siguiente aspecto



Si le damos al botón de inicio de sesión nos debe dar por ahora el siguiente resultado



Es todo por esta sesión en la próxima desarrollaremos la api en el backend y su integración de la funcionalidad de inicio de sesión .