# Authentification Laravel – VueJS

# LARAVEL

# Créer les Controllers

php artisan make:controller Auth/LoginController

php artisan make:controller Auth/RegisterController



<?php

namespace App\Http\Controllers\Auth;

use App\Http\Controllers\Controller;

use Illuminate\Http\Request;

use Illuminate\Support\Facades\Auth;

use Illuminate\Validation\ValidationException;

class LoginController extends Controller

{

    public function login(Request $request)

    {

        $request->validate([

            'email' => 'required|email',

            'password' => 'required',

        ]);

        if (Auth::attempt($request->only('email', 'password'))) {

            $user = Auth::user();

            $token = $user->createToken('token-name')->plainTextToken;

            return response()->json([

                'status' => true,

                'message' => 'User Logged In Successfully',

                'token' => $token,

                'user' =>$user

            ], 200);

        }

        throw ValidationException::withMessages([

            'email' => ['The provided credentials are incorrect.'],

        ]);

    }

    public function logout(Request $request)

    {

        $request->user()->tokens()->delete();

        return response()->json([

        'message' => 'Successfully logged out'

        ]);

    }

}



<?php

namespace App\Http\Controllers\Auth;

use App\Http\Controllers\Controller;

use Illuminate\Http\Request;

use App\Models\User;

use Illuminate\Support\Facades\Hash;

class RegisterController extends Controller

{

    public function register(Request $request)

    {

        $request->validate([

                'name' => 'required',

                'email' => 'required|email|unique:users,email',

                'password' => 'required|min:6'

        ]);

        $user = User::create([

            'name' => $request->name,

            'email' => $request->email,

            'password' => Hash::make($request->password),

        ]);

        $token = $user->createToken('token-name')->plainTextToken;

        return response()->json([

            'status' => true,

            'message' => 'User Created Successfully',

            'token' => $token

        ], 200);

    }

}

# Créer les routes

# 

use App\Http\Controllers\Auth\LoginController;

use App\Http\Controllers\Auth\RegisterController;

Route::post('/login', [LoginController::class, 'login']);

Route::post('/register', [RegisterController::class, 'register']);

Route::middleware('auth:sanctum')->post('/logout', [LoginController::class, 'logout']);

# Tester les APIs

# 

# 

Pour le test du logout, ça nécessite de passer à l’onglet Headers puis ajouter la ligne Authorization à laquelle on associe la valeur Bearer puis espace puis coller le token généré dans le login.

# 

# VueJS

# Créer Component Vuejs pour Login

Créez un fichier dans le dossier resources/js/components/**authentification** ayant le nom **login.vue**



<template>

    <div class="marge">

        <div class="row justify-content-center">

            <div class="col-md-8">

                <div class="card card-default">

                    <div class="card-header">Login</div>

                    <div class="card-body">

                        <form>

                            <div class="form-group row">

                                <label for="email" class="col-sm-4 col-form-label text-md-right">E-Mail Address</label>

                                <div class="col-md-6">

                                    <input id="email" type="email" class="form-control" v-model="user.email" required

                                           autofocus autocomplete="off">

                                </div>

                            </div>

                            <div class="form-group row">

                                <label for="password" class="col-md-4 col-form-label text-md-right">Password</label>

                                <div class="col-md-6">

                                    <input id="password" type="password" class="form-control" v-model="user.password"

                                           required autocomplete="off">

                                </div>

                            </div>

                            <div class="form-group row mb-0">

                                <div class="col-md-8 offset-md-4">

                                    <button type="submit" class="btn btn-primary" @click="handleLogin">

                                        Login

                                    </button>

                                </div>

                            </div>

                        </form>

                    </div>

                </div>

            </div>

        </div>

    </div>

</template>

<script setup>

  import axios from 'axios';

  import { useRouter } from 'vue-router';

  const router = useRouter()

      let  user= {}

    const   handleLogin=async()=> {

           axios.post('http://localhost:8000/api/login/', user)

                      .then((response) => {

                        console.log(response)

                         router.push("/dashboard")

                         localStorage.setItem('user',response.data.user.name)

                         localStorage.setItem('token',response.data.token)

                      })

                      .catch(err => {console.log(err);

                       alert(err)

                    })

        }

  </script>

<style scoped>

.marge {

    position: fixed;

    width: 100%;

    height: 300px;

    margin: 5% auto;

    padding: 20px;

    border: 5px solid #ccc;

    background-color: #fff;

}

</style>

# Créer Component Vuejs pour Register

Créez un fichier dans le dossier resources/js/components/authentification ayant le nom **register.vue**



<template>

    <div class="marge">

        <div class="row justify-content-center">

            <div class="col-md-8">

                <div class="card card-default">

                    <div class="card-header">Register</div>

                    <div class="card-body">

                        <form>

                            <div class="form-group row">

                                <label for="name" class="col-sm-4 col-form-label text-md-right">Name</label>

                                <div class="col-md-6">

                                    <input id="name" type="text" class="form-control" v-model="user.name" required

                                           autofocus autocomplete="off">

                                </div>

                            </div>

                            <div class="form-group row">

                                <label for="email" class="col-sm-4 col-form-label text-md-right">E-Mail Address</label>

                                <div class="col-md-6">

                                    <input id="email" type="email" class="form-control" v-model="user.email" required

                                           autofocus autocomplete="off">

                                </div>

                            </div>

                            <div class="form-group row">

                                <label for="password" class="col-md-4 col-form-label text-md-right">Password</label>

                                <div class="col-md-6">

                                    <input id="password" type="password" class="form-control" v-model="user.password"

                                           required autocomplete="off">

                                </div>

                            </div>

                            <div class="form-group row">

                                <label for="password" class="col-md-4 col-form-label text-md-right">Password Confirmation</label>

                                <div class="col-md-6">

                                    <input id="password\_confirmation" type="password" class="form-control" v-model="user.password\_confirmation"

                                           required autocomplete="off">

                                </div>

                            </div>

                            <div class="form-group row mb-0">

                                <div class="col-md-8 offset-md-4">

                                    <button type="submit" class="btn btn-primary" @click="handleSubmit">

                                        Register

                                    </button>

                                </div>

                            </div>

                        </form>

                    </div>

                </div>

            </div>

        </div>

    </div>

</template>

<script setup>

  import axios from "axios"

  import { useRouter } from 'vue-router';

  const router = useRouter()

      let  user= {}

    const  handleSubmit=async()=> {

      axios.post('http://localhost:8000/api/register/', user)

                  .then(response => {

                          console.log(response)

                          router.replace({ name: 'login' })

                   })

                  .catch(err => {console.log(err);alert(err) })

    }

  </script>

<style scoped>

.marge {

    position: fixed;

    width: 100%;

    height: 300px;

    margin: 5% auto;

    padding: 20px;

    border: 5px solid #ccc;

    background-color: #fff;

}

</style>

# Créer Component dashboard

Créer resources/js/components/**admin**/**dashboard.vue**

<template>

                    <div >

                        <h3> Dashboard</h3>

                    </div>

</template>

<script setup>

</script>

# Ajouter les routes

Dans : resources/js/routes.js ajouter :



import  Login from './components/authentification/login.vue'

import Register from './components/authentification/register.vue'

import  Dashboard from './components/admin/dashboard.vue'

export const routes = [

**………..**

    {

        name: "login",

        path: "/login",

        component: Login,

    },

    {

        name: "register",

        path: "/register",

        component: Register,

    },

     {

                name: "dashboard",

                path: '/dashboard',

                component: Dashboard,

   }

]

# Modifier Menu.vue

Ajouter dans ce fichier ces ligne de code :



……………..

 <li class="nav-item">

          <router-link class="nav-link active"  to="/login">Login

            </router-link>

        </li>

        <li class="nav-item">

          <router-link class="nav-link active"  to="/register">Register

            </router-link>

        </li>

        <li class="nav-item">

          <button @click="logout">Logout</button>

        </li>

…………..

<script setup>

import axios from 'axios';

import { useRouter } from 'vue-router';

const router = useRouter();

const   logout=async()=> {

 let token=""

           token=localStorage.getItem('token')

           console.log(token)

           axios.post('http://localhost:8000/api/logout', null, {

  headers: {

    'Authorization': `Bearer ${token}`

  }

})

                      .then((response) => {

                        console.log(response)

                        localStorage.removeItem('token')

                        localStorage.removeItem('user')

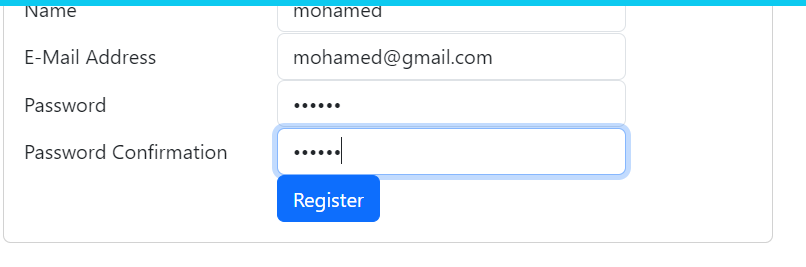
                          router.push("/login")

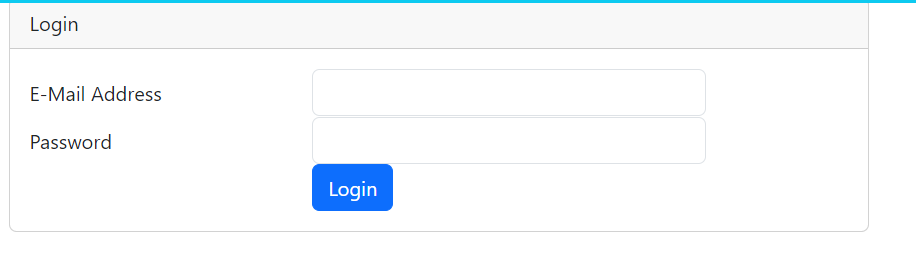
                      })

                      .catch(err => {console.log(err);alert(err) })

        }

</script>





# Protected routes

Afin de protéger le routeur Vue, nous allons utiliser les gardes de navigation vue. En utilisant les gardes de navigation vue, nous pouvons valider avant ou après avoir entré l'itinéraire. Afin de vérifier que l'utilisateur est connecté ou non, nous l'utiliserons avant d'entrer dans le routeur.

Le routeur Vue prend en charge le champs meta, que nous pouvons inclure lors de la définition de la route pour pouvoir la protéger.

Dans **router/index.js**

Ajouter ***meta:{isAuth:true}***

**Exemple :**

{

        name:"Viewcategorie",

        path:"/listcat",

        component:Viewcategorie,

**meta:{isAuth:true}**

    },

Puis ajouter le code suivant dans **app.js** avec lequel nous vérifions d'abord si la route nécessite une authentification en vérifiant to.matched.some(record => record.meta.isAuth). Si c'est le cas, nous vérifions si l'utilisateur est authentifié. Si l'utilisateur n'est pas authentifié et qu'il ne se trouve pas sur la page de login, nous le redirigeons vers "/" pour éviter la boucle infinie de redirection. Sinon, si l'utilisateur est authentifié ou sur la page de login, nous autorisons la navigation.

Aussi, si la route n'a pas de meta.isAuth, nous autorisons également la navigation sans condition.

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

    if (to.matched.some(record => record.meta.isAuth)) {

        let token = localStorage.getItem('token');

        if (!token) {

            next("/login");

        } else {

            console.log(token);

            next(); // Mettre next() ici après avoir vérifié l'authentification

        }

    } else {

        next(); // Mettre next() ici si la route n'a pas besoin d'authentification

    }

  });

beforeEach est une bibliothèque de routage pour les applications Vue.js. Elle est utilisée pour définir des règles et des comportements qui doivent être respectés avant de permettre à un utilisateur de passer d'une page à une autre.

Chaque fois qu'un utilisateur tente de naviguer vers une nouvelle route (par exemple, en cliquant sur un lien ou en entrant manuellement une URL), le hook beforeEach est déclenché.

Les guards peuvent être résolus de manière asynchrone et la navigation est considérée comme en attente avant que tous les hooks n'aient été résolus.

to : l'emplacement de la route cible dans un format normalisé vers lequel vous naviguez.

from : l'emplacement actuel de la route dans un format normalisé à partir duquel la navigation s'éloigne.

Dans la fonction beforeEach, vous avez accès à une fonction next en tant que troisième argument. Vous pouvez utiliser cette fonction pour décider si la navigation doit être autorisée (next()) ou si elle doit être bloquée (next(false)) en fonction des conditions vérifiées. Vous pouvez également rediriger l'utilisateur vers une autre route (next('/nouvelle-route')) en fonction de ces conditions.

Si vous appelez next() sans argument ou avec next(true), la navigation est autorisée. Si vous appelez next(false), la navigation est bloquée. Si vous appelez next('/nouvelle-route'), l'utilisateur est redirigé vers la nouvelle route.