YISU Travel GmbH

Laravel Backend

1. Powershell als Admin und:

# Run as administrator...

Set-ExecutionPolicy Bypass -Scope Process -Force; [System.Net.ServicePointManager]::SecurityProtocol = [System.Net.ServicePointManager]::SecurityProtocol -bor 3072; iex ((New-Object System.Net.WebClient).DownloadString('https://php.new/install/windows/8.4'))

**Version: v11.5.0**

**LARAVEL v11.36.1 plugin v1.1.1**

ausführen.

<https://laravel.com/docs/11.x/installation>

https://www.youtube.com/watch?v=WumgBzENYYk&t=308s

1. Ein Ordner stellen und:
   1. Laravel new YISUTravelBackend
   2. Starter kit: none
   3. Pest Framework
   4. MySQL nutzen
   5. ➜ cd YISUTravelBackend
   6. ➜ npm install && npm run build
   7. ➜ composer run dev
   8. php artisan install:api
2. Checke .evn file um MySQL Daten einzutragen

* Du kannst in IONOS dir die Daten ansehen oder MySQL80 starten, Datenbank schema ist wichtig
* composer require laravel/sanctum
* php artisan vendor:publish --provider="Laravel\Sanctum\SanctumServiceProvider"

Ein Bild, das Text, Screenshot, Software, Multimedia-Software enthält.

Automatisch generierte Beschreibung

Ein Bild, das Text, Screenshot, Display, Software enthält.

Automatisch generierte Beschreibung

Ein Bild, das Text, Screenshot, Schrift enthält.

Automatisch generierte Beschreibung

Laravel IONOS:  
Composer version 2.8.4 2024-12-11 11:57:47

Erstelle Subdomain:  
Ein Bild, das Text, Screenshot, Schrift, Zahl enthält.

Automatisch generierte Beschreibung

Folgende dem Video:  
https://www.youtube.com/watch?v=Qqp9qPgULEA

php artisan key:generate

Um Database Tabellen zu migraten, füre das aus:  
php8.3 artisan migrate

**1. Set Up Laravel Authentication API**

Ensure your Laravel backend has an authentication system set up. Here’s how to proceed:

1. **Install Laravel Sanctum for API Authentication**:

composer require laravel/sanctum

php artisan vendor:publish --provider="Laravel\Sanctum\SanctumServiceProvider"

php artisan migrate

1. **Enable Sanctum Middleware**: Update api.php routes to include Sanctum's middleware:

use Illuminate\Http\Request;

Route::middleware('auth:sanctum')->get('/user', function (Request $request) {

return $request->user();

});

**Create Login Endpoint**: In your Laravel routes/api.php, add a route for login:

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

**Create the AuthController**: Generate the controller:

php artisan make:controller AuthController

Implement the login method:

use Illuminate\Http\Request;

use Illuminate\Support\Facades\Auth;

public function login(Request $request)

{

$credentials = $request->only('name', 'password');

if (!Auth::attempt($credentials)) {

return response()->json(['message' => 'Unauthorized'], 401);

}

$user = Auth::user();

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

return response()->json(['token' => $token, 'user' => $user]);

}

1. **Update Angular Frontend**

ng generate service auth

import { Injectable } from '@angular/core';

import { HttpClient } from '@angular/common/http';

import { Observable } from 'rxjs';

@Injectable({

providedIn: 'root',

})

export class AuthService {

private apiUrl = 'http://your-laravel-api-url.com/api';

constructor(private http: HttpClient) {}

login(credentials: { name: string; password: string }): Observable<any> {

return this.http.post(`${this.apiUrl}/login`, credentials);

}

}

**Update Angular Component**

Update the onsubmit method in your AdminLoginComponent:

import { Component } from '@angular/core';

import { FormBuilder, FormGroup, Validators } from '@angular/forms';

import { AuthService } from './auth.service';

@Component({

selector: 'app-admin-login',

templateUrl: './admin-login.component.html',

styleUrls: ['./admin-login.component.css'],

})

export class AdminLoginComponent {

contactForm: FormGroup;

constructor(private fb: FormBuilder, private authService: AuthService) {

this.contactForm = this.fb.group({

name: ['', Validators.required],

password: ['', Validators.required],

});

}

onsubmit(): void {

if (this.contactForm.valid) {

const credentials = this.contactForm.value;

this.authService.login(credentials).subscribe({

next: (response) => {

console.log('Login successful', response);

localStorage.setItem('authToken', response.token);

},

error: (err) => {

console.error('Login failed', err);

},

});

}

}

}

**3. Protect Angular Routes**

To secure routes in Angular, you need a guard:

ng generate guard auth

import { Injectable } from '@angular/core';

import { CanActivate, Router } from '@angular/router';

@Injectable({

providedIn: 'root',

})

export class AuthGuard implements CanActivate {

constructor(private router: Router) {}

canActivate(): boolean {

const token = localStorage.getItem('authToken');

if (token) {

return true;

}

this.router.navigate(['/login']);

return false;

}

}

**Apply Guard to Routes**: In your app-routing.module.ts, use the guard:

import { AuthGuard } from './auth.guard';

const routes = [

{ path: 'dashboard', component: DashboardComponent, canActivate: [AuthGuard] },

{ path: 'login', component: AdminLoginComponent },

];