Projet Santé & Fitness – Documentation

Introduction

L'application « Suivi Santé et Fitness » a été développée dans le cadre de la fin de formation du niveau avancé à DCLIC. Elle a pour objectif principal d'accompagner les utilisateurs dans l'amélioration de leur bien-être physique et de leur hygiène de vie. À travers une interface simple et intuitive, l'application permet aux utilisateurs de :

- Suivre leurs activités physiques (type d'activité, durée, calories brûlées, etc.),
- Gérer leur alimentation en enregistrant leurs repas quotidiens,
- Recevoir des notifications personnalisées pour les rappels d'activités ou de repas,
- Consulter un tableau de bord pour visualiser leur progression personnelle en matière de santé et de bien-être.

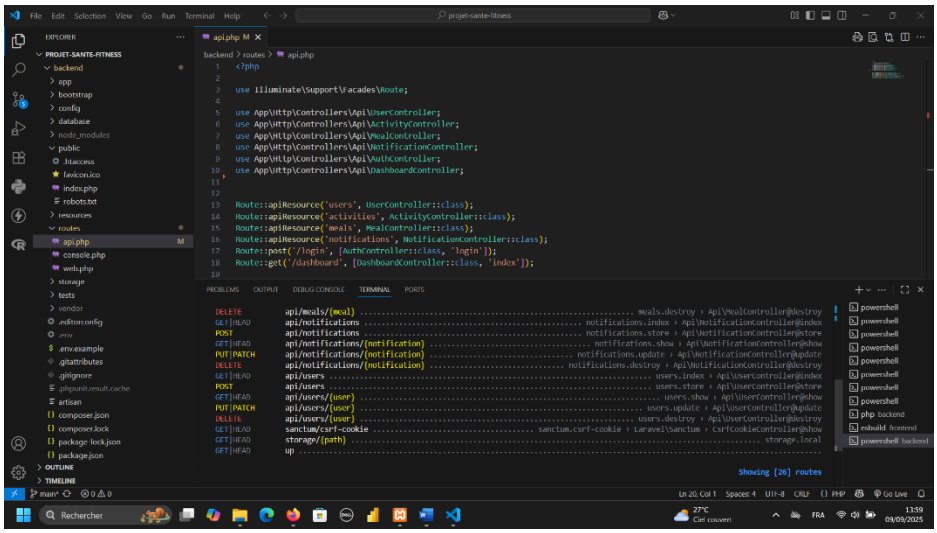
Le présent document a pour objectif de fournir une description claire du projet, accompagnée de captures d'écran de l'application et d'extraits de code source, afin d'illustrer la conception, les fonctionnalités et le fonctionnement global de la solution.

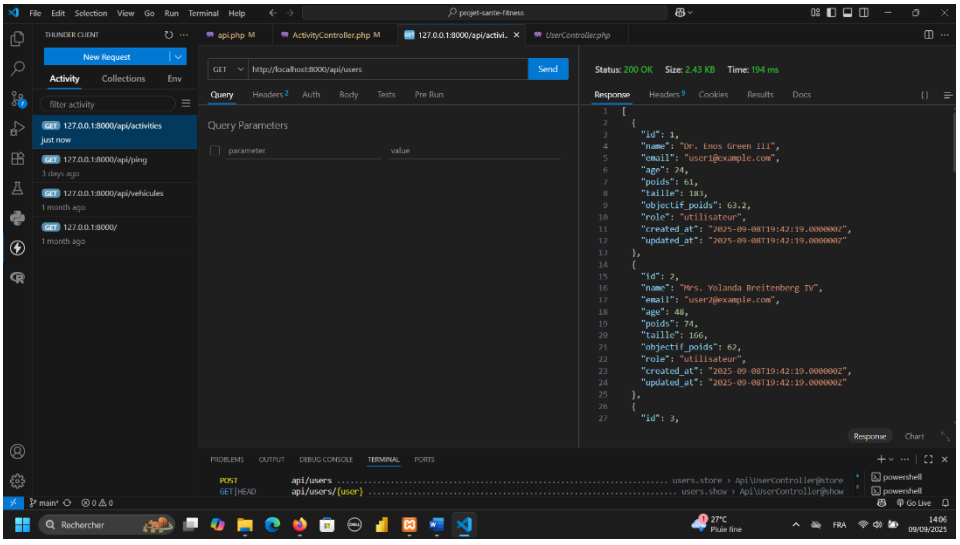
I. Technologies utilisées

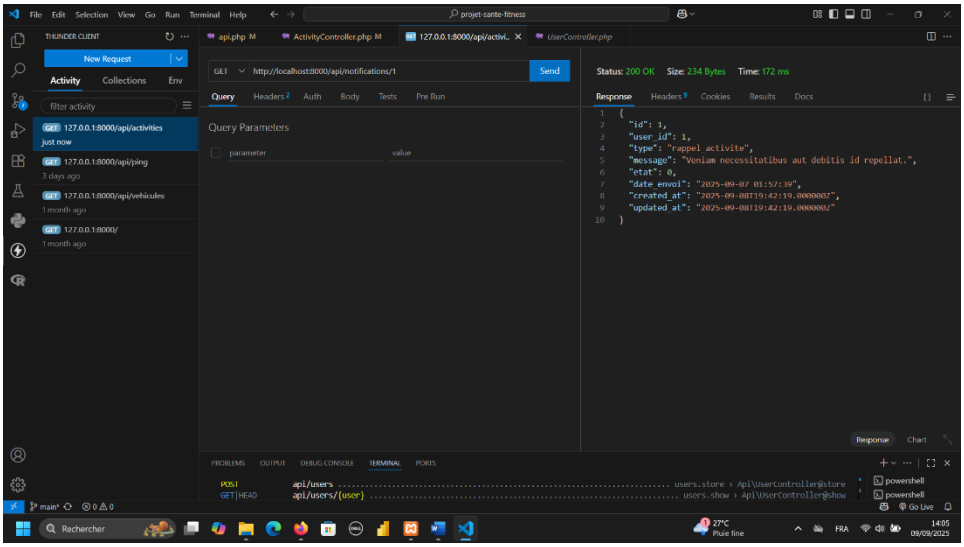
Le développement de l'application **« Suivi Santé et Fitness »** s'appuie sur un ensemble de technologies modernes et adaptées à la création d'une architecture **full-stack** :

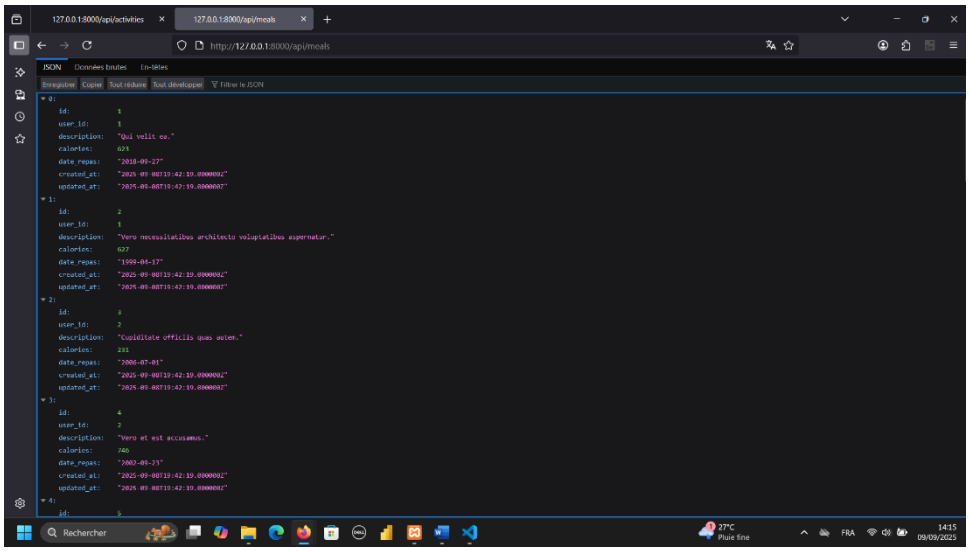
a. Back-end : Laravel 12.28.1

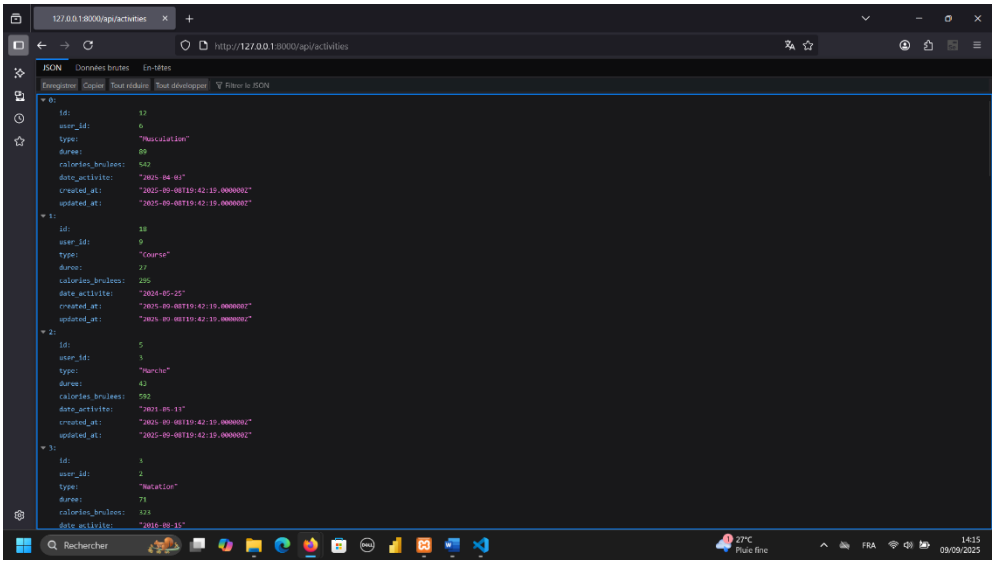
Le back-end de l'application est développé avec Laravel 12.28.1. Il met en place une API REST structurée pour gérer l'ensemble de la logique métier et les échanges de données. Cette API propose des endpoints CRUD pour les utilisateurs, les activités, les repas et les notifications, centralisant ainsi la gestion des informations dans une base de données MySQL. La communication avec le front-end React est assurée via un middleware CORS personnalisé, garantissant des échanges sécurisés et fluides. Cette architecture permet de séparer clairement la logique serveur de l'interface utilisateur, rendant le système modulaire, évolutif et facile à maintenir.

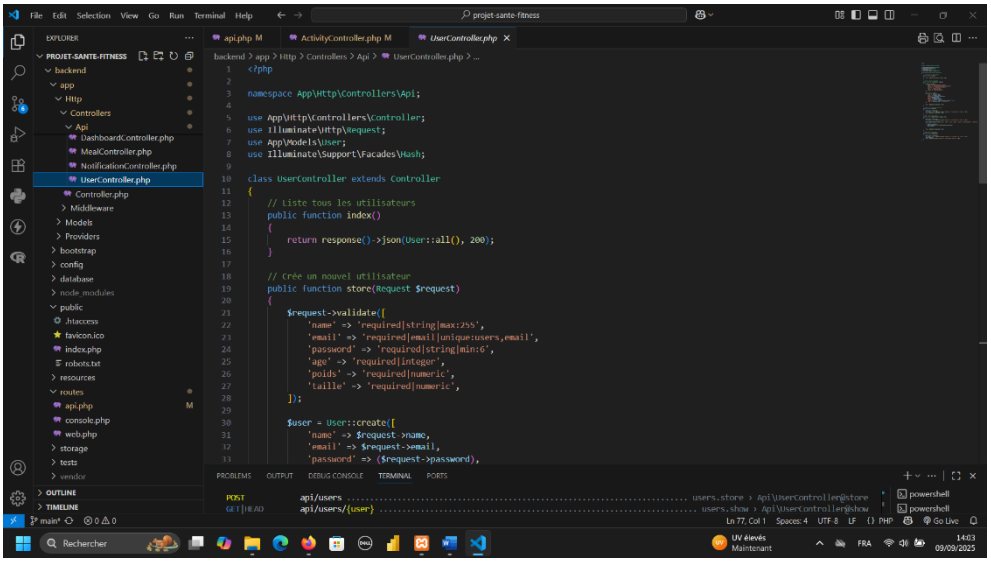

api.php M
backend > routes > api.php
<?php
use Illuminate\Support\Facades\Route;
use App\Http\Controllers\Api\UserController;
use App\Http\Controllers\Api\ActivityController;
use App\Http\Controllers\Api\MealController;
use App\Http\Controllers\Api\NotificationController;
use App\Http\Controllers\Api\AuthController;
use App\Http\Controllers\Api\DashboardController;
Route::apiResource('users', UserController::class);
Route::apiResource('activities', ActivityController::class);
Route::apiResource('meals', MealController::class);
Route::apiResource('notifications', NotificationController::class);
Route::post('/login', [AuthController::class, 'login']);
Route::get('/dashboard', [DashboardController::class, 'index']);
DELETE api/meals/{meal}
GET|HEAD api/notifications
POST api/notifications
GET|HEAD api/notifications/{notification}
PUT|PATCH api/notifications/{notification}
DELETE api/notifications/{notification}
GET|HEAD api/users
POST api/users
GET|HEAD api/users/{user}
PUT|PATCH api/users/{user}
DELETE api/users/{user}
GET|HEAD sanctum/csrf-cookie
GET|HEAD storage/{path}
GET|HEAD up
Showing [26] routes


THUNDER CLIENT
New Request
GET http://localhost:8000/api/users
Send
Status: 200 OK Size: 2.43 KB Time: 194 ms
Query Parameters
"id": 1,
"name": "Dr. Emos Green III",
"email": "user1@example.com",
"age": 24,
"poids": 61,
"taille": 183,
"objectif_poids": 63.2,
"role": "utilisateur",
"created_at": "2025-09-08T19:42:19.000000Z",
"updated_at": "2025-09-08T19:42:19.000000Z"
"id": 2,
"name": "Mrs. Yolanda Breitenberg IV",
"email": "user2@example.com",
"age": 48,
"poids": 74,
"taille": 166,
"objectif_poids": 62,
"role": "utilisateur",
"created_at": "2025-09-08T19:42:19.000000Z",
"updated_at": "2025-09-08T19:42:19.000000Z"
"id": 3,
POST api/users
GET|HEAD api/users/{user}


THUNDER CLIENT
New Request
GET http://localhost:8000/api/notifications/1
Send
Status: 200 OK Size: 234 Bytes Time: 172 ms
Query Parameters
"id": 1,
"user_id": 1,
"type": "rappel activite",
"message": "Veniam necessitatibus aut debitis id repellat.",
"etat": 0,
"date_envoi": "2025-09-07 01:57:39",
"created_at": "2025-09-08T19:42:19.000000Z",
"updated_at": "2025-09-08T19:42:19.000000Z"
POST api/users
GET|HEAD api/users/{user}

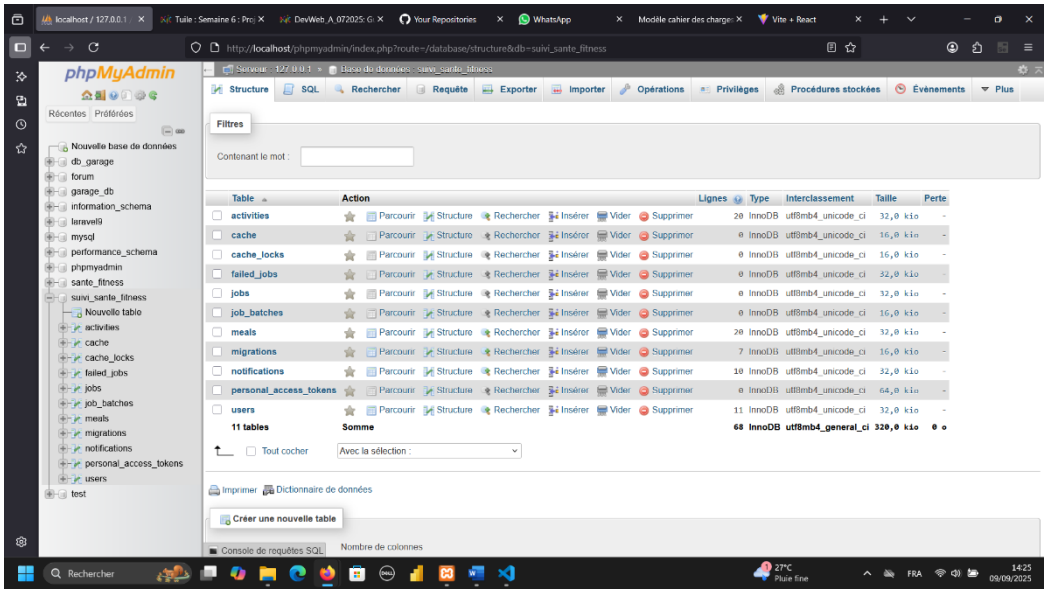

127.0.0.1:8000/api/activities
127.0.0.1:8000/api/meals
http://127.0.0.1:8000/api/meals
JSON
Données brutes
En-têtes
Enregistrer
Copier
Tout réduire
Tout développer
Filtrer le JSON
0:
id: 1
user_id: 1
description: "Qui velit ea."
calories: 623
date_repas: "2018-09-27"
created_at: "2025-09-08T19:42:19.000000Z"
updated_at: "2025-09-08T19:42:19.000000Z"
1:
id: 2
user_id: 1
description: "Vero necessitatibus architecto voluptatibus aspernatur."
calories: 627
date_repas: "1999-04-17"
created_at: "2025-09-08T19:42:19.000000Z"
updated_at: "2025-09-08T19:42:19.000000Z"
2:
id: 3
user_id: 3
description: "Cupiditate officiis quas autem."
calories: 281
date_repas: "2006-07-01"
created_at: "2025-09-08T19:42:19.000000Z"
updated_at: "2025-09-08T19:42:19.000000Z"
3:
id: 4
user_id: 2
description: "Vero et est accusamus."
calories: 760
date_repas: "2002-05-23"
created_at: "2025-09-08T19:42:19.000000Z"
updated_at: "2025-09-08T19:42:19.000000Z"
4:
id: 5
Rechercher
27°C
Pluie fine
FRA
14:15
09/09/2025

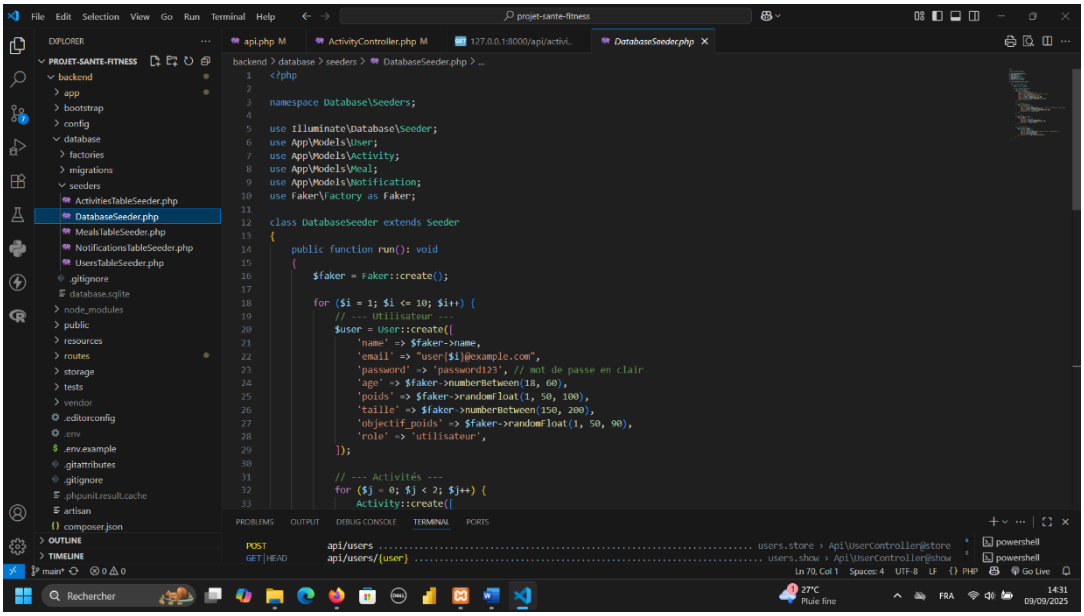

127.0.0.1:8000/api/activities
http://127.0.0.1:8000/api/activities
JSON
Données brutes
En-têtes
Enregistrer
Copier
Tout réduire
Tout développer
Filtrer le JSON
0:
id: 12
user_id: 6
type: "Musculation"
duree: 89
calories_brulees: 542
date_activite: "2025-04-01"
created_at: "2025-09-08T19:42:19.000000Z"
updated_at: "2025-09-08T19:42:19.000000Z"
1:
id: 18
user_id: 9
type: "Course"
duree: 27
calories_brulees: 295
date_activite: "2024-05-25"
created_at: "2025-09-08T19:42:19.000000Z"
updated_at: "2025-09-08T19:42:19.000000Z"
2:
id: 5
user_id: 3
type: "Marche"
duree: 43
calories_brulees: 592
date_activite: "2021-05-11"
created_at: "2025-09-08T19:42:19.000000Z"
updated_at: "2025-09-08T19:42:19.000000Z"
3:
id: 1
user_id: 2
type: "Natation"
duree: 71
calories_brulees: 324
date_activite: "2016-08-15"
Rechercher
27°C
Pluie fine
FRA
14:15
09/09/2025

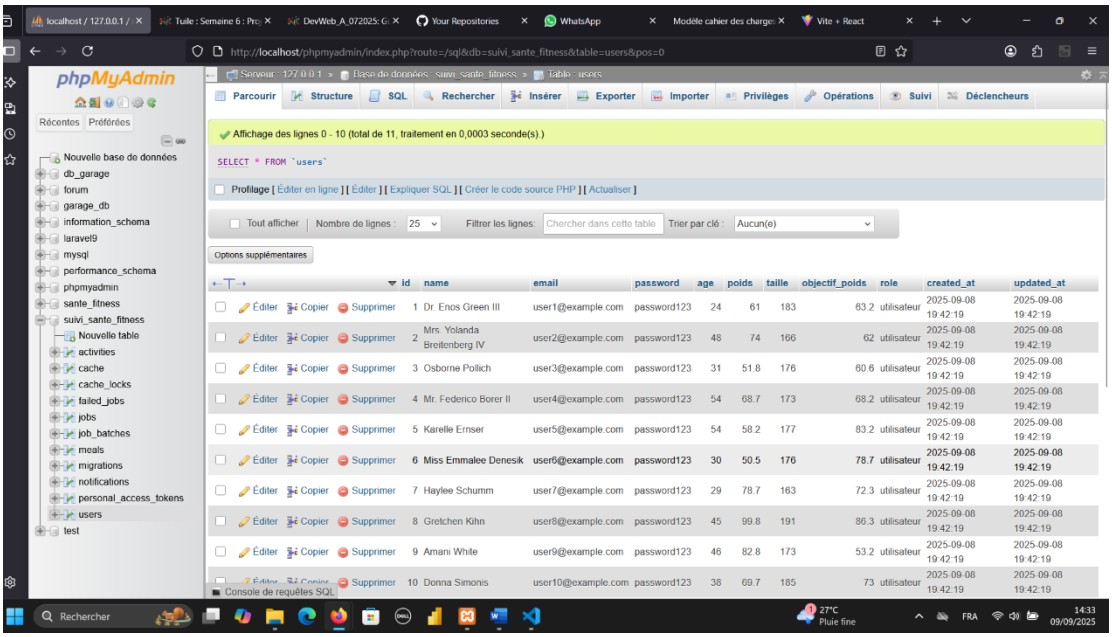

File Edit Selection View Go Run Terminal Help
projet-sante-fitness
EXPLORER
PROJET-SANTE-FITNESS
backend
app
Http
Controllers
Api
DashboardController.php
MealController.php
NotificationController.php
UserController.php
Controller.php
Middleware
Models
Providers
bootstrap
config
database
node_modules
public
.htaccess
favicon.ico
index.php
robots.txt
resources
routes
api.php
console.php
web.php
storage
tests
vendor
OUTLINE
TIMELINE
api.php M
ActivityController.php M
UserController.php
backend > app > Http > Controllers > Api > UserController.php > ...
<?php
namespace App\Http\Controllers\Api;
use App\Http\Controllers\Controller;
use Illuminate\Http\Request;
use App\Models\User;
use Illuminate\Support\Facades\Hash;
class UserController extends Controller
{
// Liste tous les utilisateurs
public function index()
{
return response()->json(User::all(), 200);
}
// Crée un nouvel utilisateur
public function store(Request $request)
{
$request->validate([
'name' => 'required|string|max:255',
'email' => 'required|email|unique:users,email',
'password' => 'required|string|min:6',
'age' => 'required|integer',
'poids' => 'required|numeric',
'taille' => 'required|numeric',
]);
$user = User::create([
'name' => $request->name,
'email' => $request->email,
'password' => ($request->password),
PROBLEMS
OUTPUT
DEBUG CONSOLE
TERMINAL
PORTS
POST api/users users.store › Api\UserController@store
GET|HEAD api/users/{user} users.show › Api\UserController@show
powershell
powershell
main*
Ln 77, Col 1
Spaces: 4
UTF-8
LF
PHP
Go Live
Rechercher
UV élevés
Maintenant
FRA
14:03
09/09/2025

b. Base de données : MySQL (via XAMPP)

La **base de données** de l'application est gérée avec **MySQL** via **XAMPP**. Elle stocke toutes les informations relatives aux utilisateurs, aux activités physiques, aux repas et aux notifications. Les relations entre les tables (users, activities, meals, notifications) permettent une gestion efficace et structurée des données, assurant l'intégrité et la cohérence des informations. Des données factices ont été générées via des **seeders**, facilitant le test des fonctionnalités de l'application et l'illustration du fonctionnement complet du système.

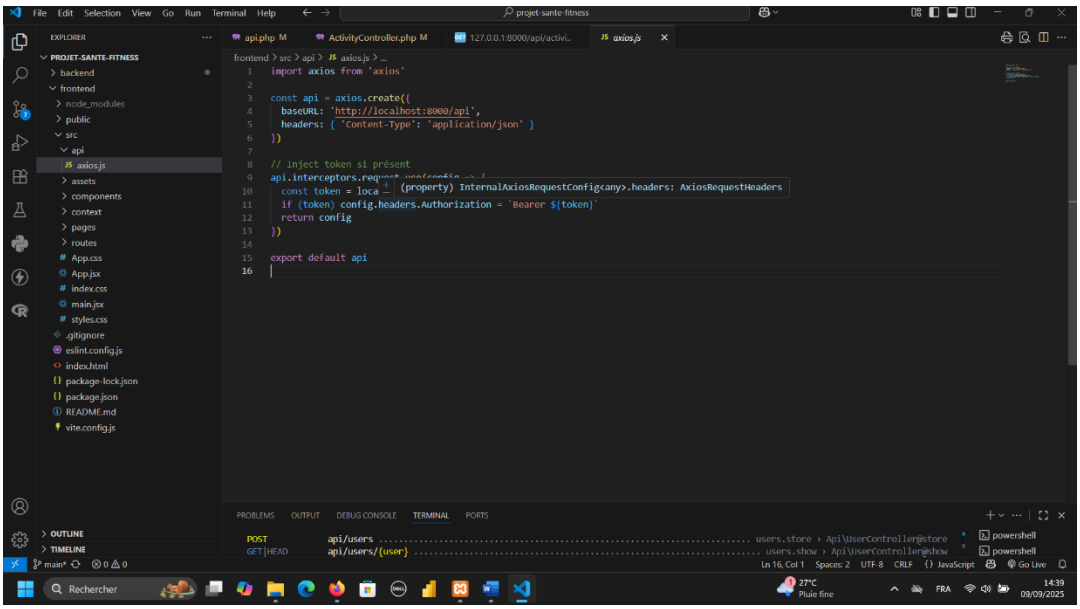


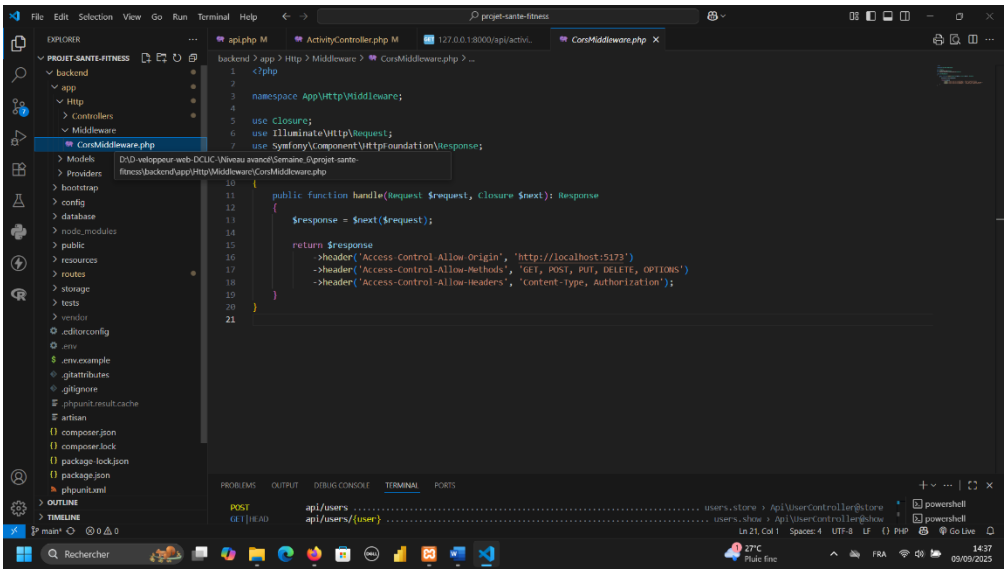


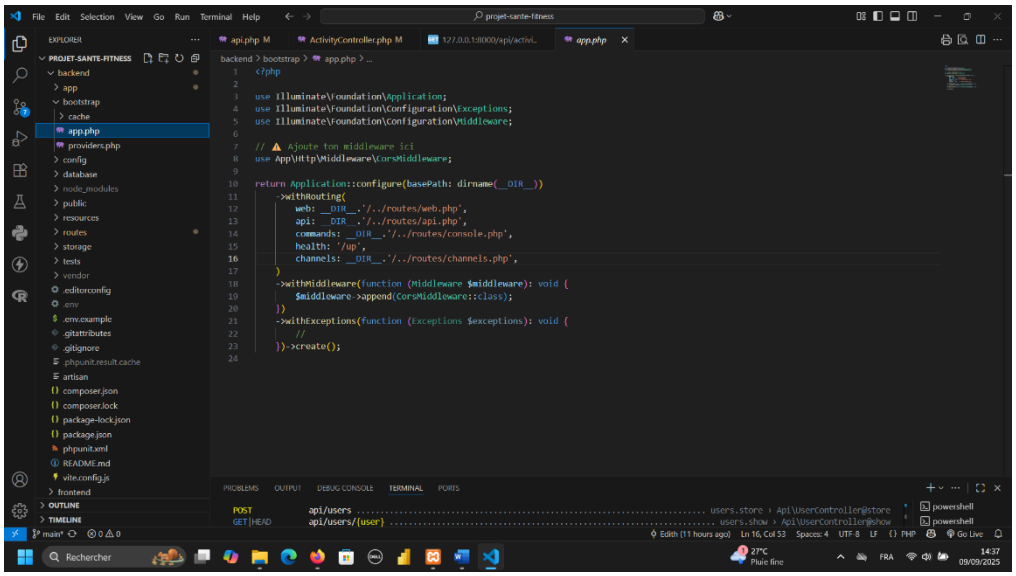


c. Communication Front/Back : CORS

La communication entre le front-end React et le back-end Laravel est assurée via CORS (Cross-Origin Resource Sharing), mise en place grâce à un middleware personnalisé directement configuré dans bootstrap/app.php. Cette configuration permet des échanges sécurisés et fluides entre l'API Laravel et le client React, garantissant le bon fonctionnement des opérations CRUD pour les utilisateurs, activités, repas et notifications. Cette approche simplifie le développement et évite les problèmes de requêtes bloquées par la politique de même origine.



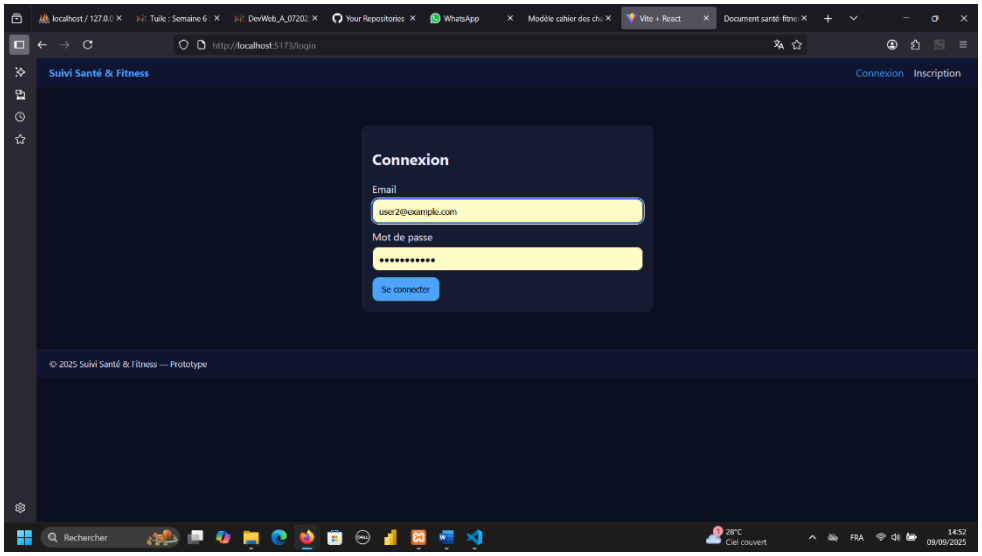


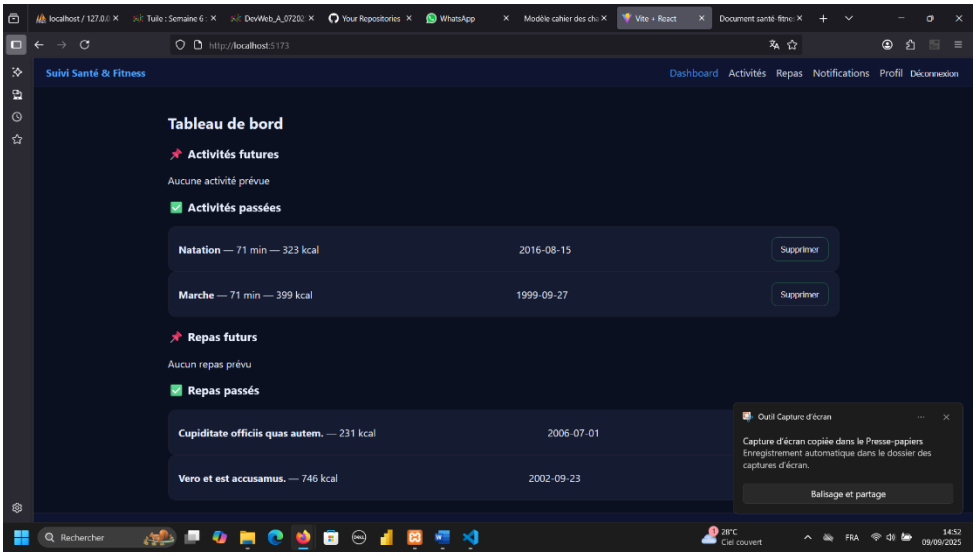


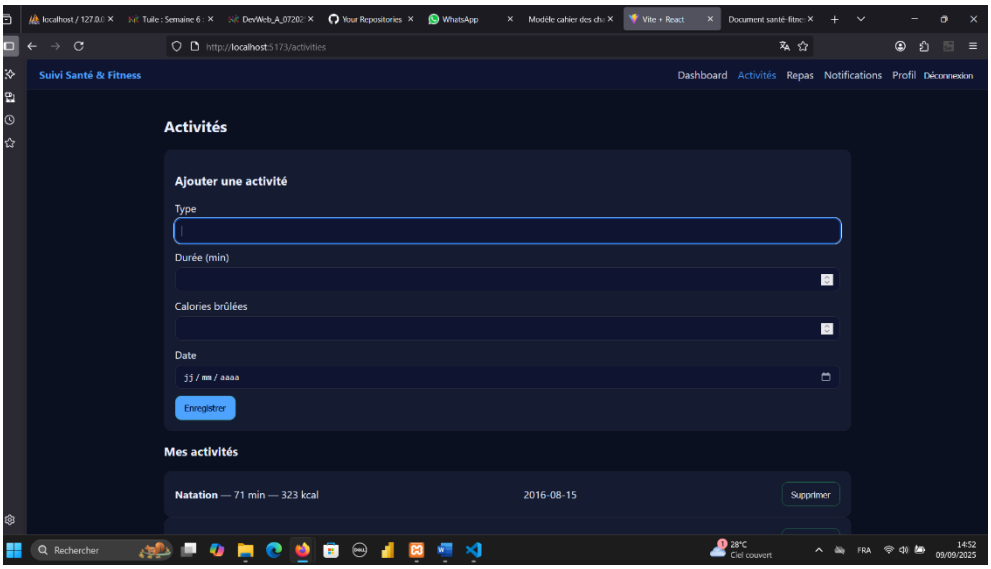
d. Front-end : React.js

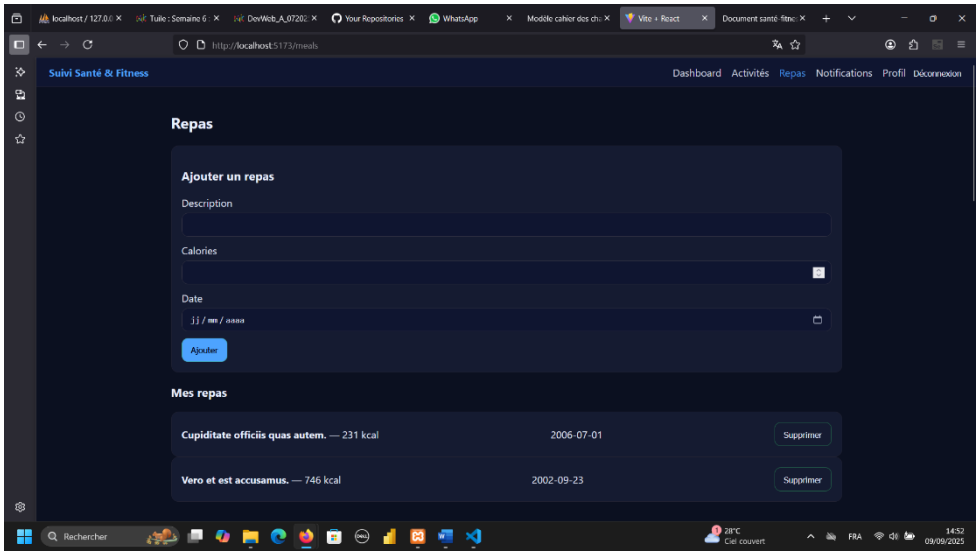
Le front-end de l'application est développé en React.js sous forme de Single Page Application (SPA), permettant une navigation fluide entre les différentes pages sans rechargement. React Router est utilisé pour gérer la navigation entre le tableau de bord, les activités, les repas, le profil et les notifications.

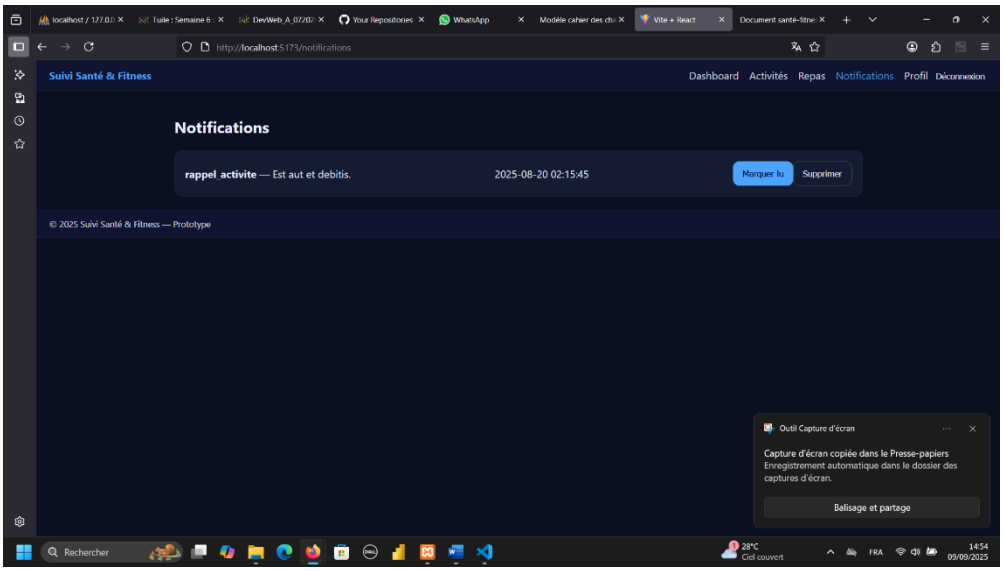
L'état global et l'authentification des utilisateurs sont gérés via Context API, ce qui centralise les données de l'utilisateur connecté ainsi que ses activités et repas. Les composants sont conçus pour être réutilisables et modulaires, tels que les formulaires d'ajout, les cartes d'activités et de repas ou les notifications, assurant une interface cohérente et ergonomique à travers toute l'application.

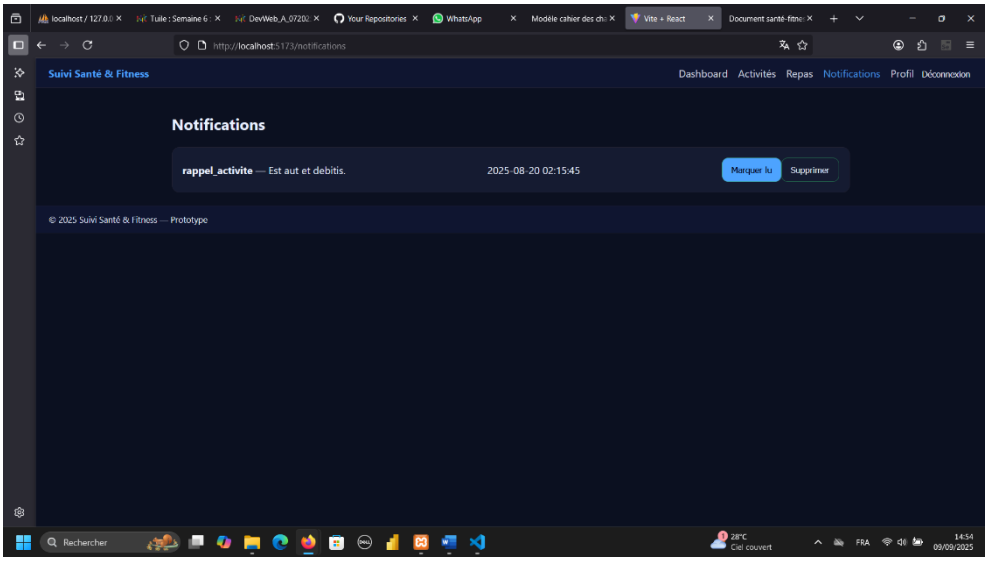

Suivi Santé & Fitness
Connexion
Inscription
Connexion
Email
user2@example.com
Mot de passe
Se connecter
© 2025 Suivi Santé & Fitness — Prototype

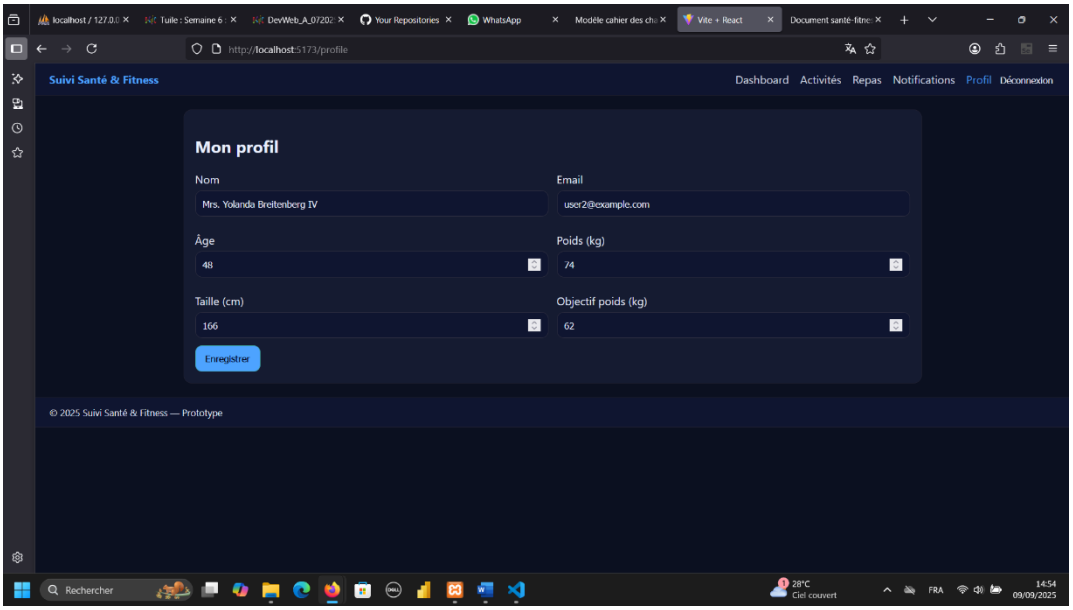

Suivi Santé & Fitness
Dashboard
Activités
Repas
Notifications
Profil
Déconnexion
Tableau de bord
Activités futures
Aucune activité prévue
Activités passées
Natation — 71 min — 323 kcal
2016-08-15
Supprimer
Marche — 71 min — 399 kcal
1999-09-27
Supprimer
Repas futurs
Aucun repas prévu
Repas passés
Cupiditate officiis quas autem. — 231 kcal
2006-07-01
Vero et est accusamus. — 746 kcal
2002-09-23
Outil Capture d'écran
Capture d'écran copiée dans le Presse-papiers
Enregistrement automatique dans le dossier des captures d'écran.
Balisage et partage


Suivi Santé & Fitness
Dashboard
Activités
Repas
Notifications
Profil
Déconnexion
Activités
Ajouter une activité
Type
Durée (min)
Calories brûlées
Date
jj / mm / aaaa
Enregistrer
Mes activités
Natation — 71 min — 323 kcal
2016-08-15
Supprimer


Suivi Santé & Fitness
Dashboard Activités Repas Notifications Profil Déconnexion
Repas
Ajouter un repas
Description
Calories
Date
jj / mm / aaaa
Ajouter
Mes repas
Cupiditate officiis quas autem. — 231 kcal
2006-07-01
Supprimer
Vero et est accusamus. — 746 kcal
2002-09-23
Supprimer


Suivi Santé & Fitness
Dashboard Activités Repas Notifications Profil Déconnexion
Notifications
rappel activite — Est aut et debitis.
2025-08-20 02:15:45
Marquer lu
Supprimer
© 2025 Suivi Santé & Fitness — Prototype
Outil Capture d'écran
Capture d'écran copiée dans le Presse-papiers
Enregistrement automatique dans le dossier des captures d'écran.
Balisage et partage


Suivi Santé & Fitness
Dashboard Activités Repas Notifications Profil Déconnexion
Notifications
rappel_activite — Est aut et debitis.
2025-08-20 02:15:45
Marquer lu
Supprimer
© 2025 Suivi Santé & Fitness — Prototype



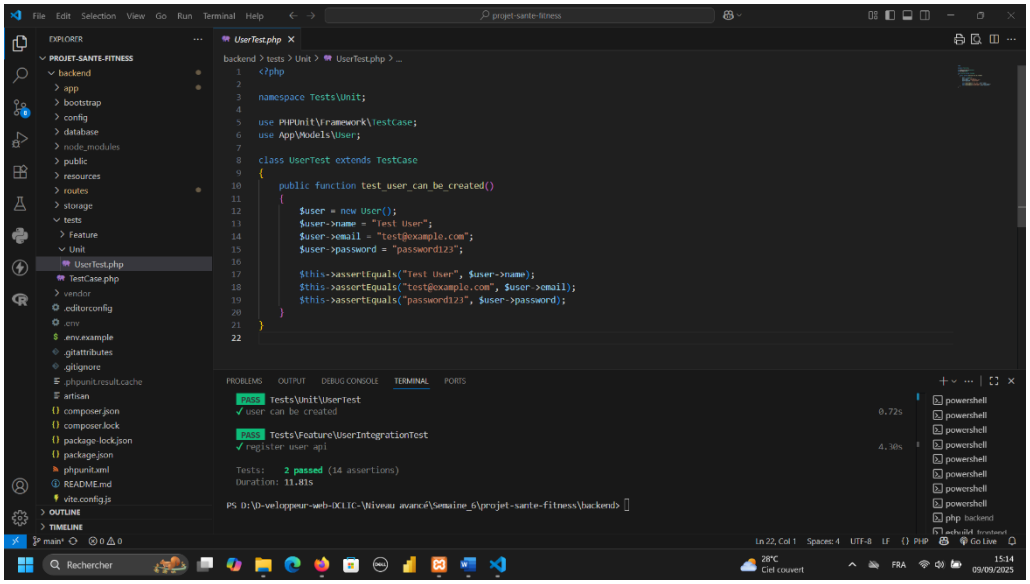
II. Tests Unitaires et d'Intégration

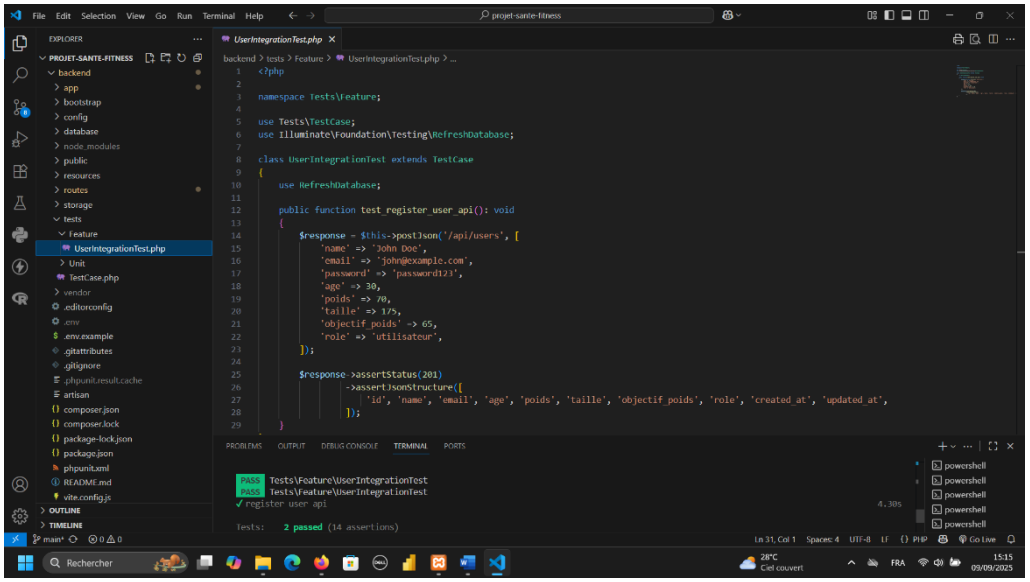
Les tests ont été réalisés pour assurer la fiabilité et la stabilité de l'application.

Les tests unitaires permettent de vérifier le fonctionnement correct des méthodes des modèles et des contrôleurs isolément. Par exemple, les tests sur le modèle User confirment que la création, la mise à jour et la suppression d'un utilisateur se comportent comme attendu.

Les tests d'intégration valident le fonctionnement global des endpoints de l'API REST. Ils s'assurent que les interactions entre le front-end et le back-end sont correctes et que les opérations CRUD sur les utilisateurs, les activités, les repas et les notifications produisent les résultats attendus. Ces tests permettent également de vérifier la sécurité des routes, notamment l'authentification et la gestion des rôles.

L'utilisation combinée de tests unitaires et d'intégration réduit les risques d'erreurs lors de l'évolution de l'application et garantit une expérience utilisateur fiable et cohérente.


UserTest.php
backend > tests > Unit > UserTest.php > ...
<?php
namespace Tests\Unit;
use PHPUnit\Framework\TestCase;
use App\Models\User;
class UserTest extends TestCase
{
public function test_user_can_be_created()
{
$user = new User();
$user->name = "Test User";
$user->email = "test@example.com";
$user->password = "password123";
$this->assertEquals("Test User", $user->name);
$this->assertEquals("test@example.com", $user->email);
$this->assertEquals("password123", $user->password);
}
}
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
PASS Tests\Unit\UserTest
✓ user can be created 0.72s
PASS Tests\Feature\UserIntegrationTest
✓ register user api 4.30s
Tests: 2 passed (14 assertions)
Duration: 11.81s
PS D:\D-veloppeur-web-DCLIC-\Niveau avancé\Semaine_6\projet-sante-fitness\backend>


UserIntegrationTest.php
backend > tests > Feature > UserIntegrationTest.php > ...
<?php
namespace Tests\Feature;
use Tests\TestCase;
use Illuminate\Foundation\Testing\RefreshDatabase;
class UserIntegrationTest extends TestCase
{
use RefreshDatabase;
public function test_register_user_api(): void
{
$response = $this->postJson('/api/users', [
'name' => 'John Doe',
'email' => 'john@example.com',
'password' => 'password123',
'age' => 30,
'poids' => 70,
'taille' => 175,
'objectif_poids' => 65,
'role' => 'utilisateur',
]);
$response->assertStatus(201)
->assertJsonStructure([
'id', 'name', 'email', 'age', 'poids', 'taille', 'objectif_poids', 'role', 'created_at', 'updated_at',
]);
}
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
PASS Tests\Feature\UserIntegrationTest
PASS Tests\Feature\UserIntegrationTest
✓ register user api 4.30s
Tests: 2 passed (14 assertions)

Conclusion

L'application **« Suivi Santé et Fitness »** constitue un outil complet pour le suivi personnalisé des activités physiques et de l'alimentation des utilisateurs. Elle combine un back-end robuste avec Laravel, offrant une API REST bien structurée et sécurisée, et un front-end moderne avec React, garantissant une interface utilisateur réactive et intuitive.

La mise en place des tests unitaires et d'intégration a permis de s'assurer de la fiabilité des fonctionnalités et de la cohérence des échanges entre le front-end et le back-end. La base de données MySQL, accompagnée de relations bien définies entre les tables, assure une gestion efficace des informations utilisateurs, activités, repas et notifications.

Grâce à cette architecture modulaire et évolutive, l'application est prête à être maintenue, enrichie de nouvelles fonctionnalités, et utilisée pour aider les utilisateurs à suivre et améliorer leur santé et leur bien-être.

.