

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.

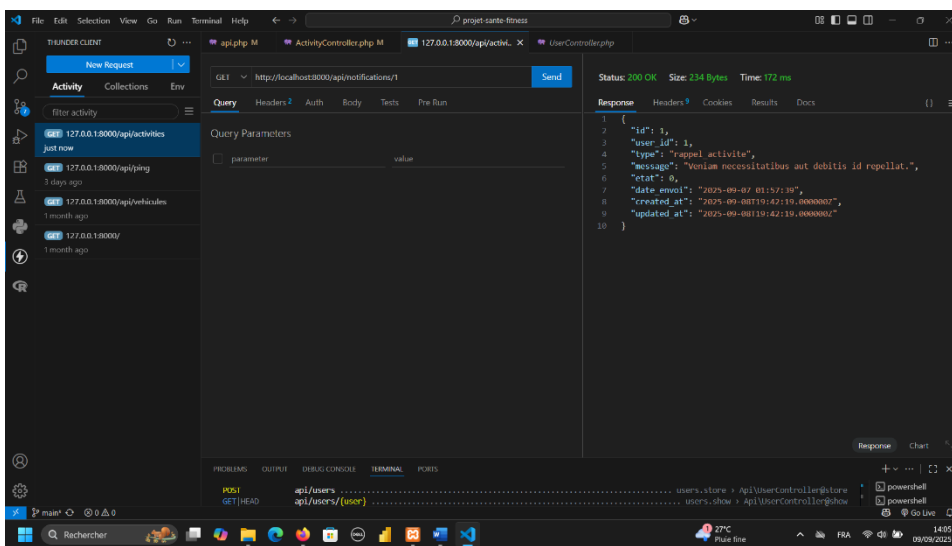
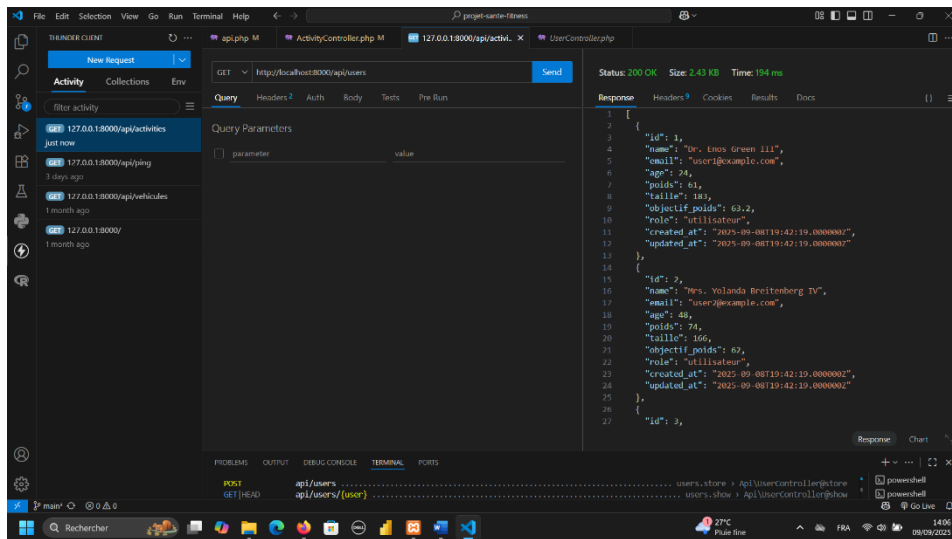
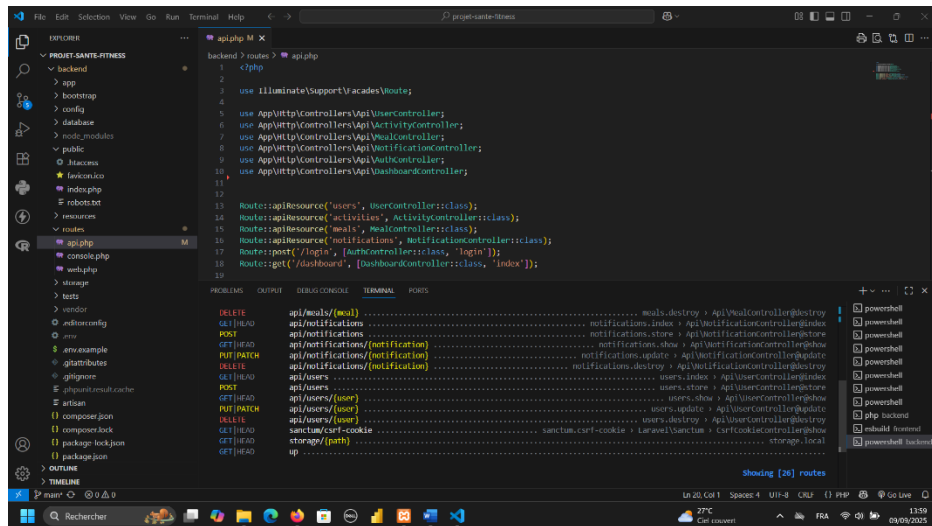
Le présent document a pour objectif de fournir une description claire du projet, accompagnée de captures d'écran de l'application et d'extraits de code source, afin d'illustrer la conception, les fonctionnalités et le fonctionnement global de la solution.

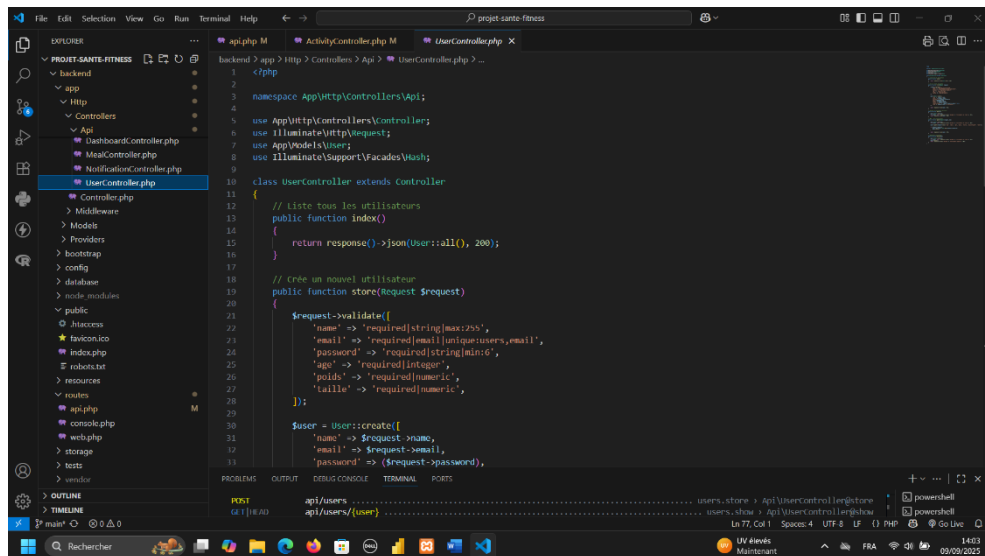
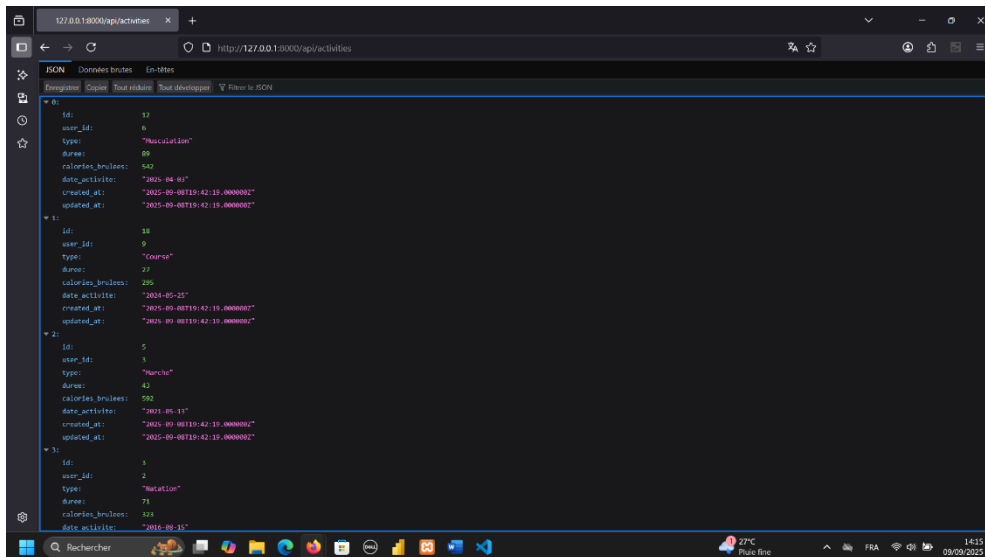
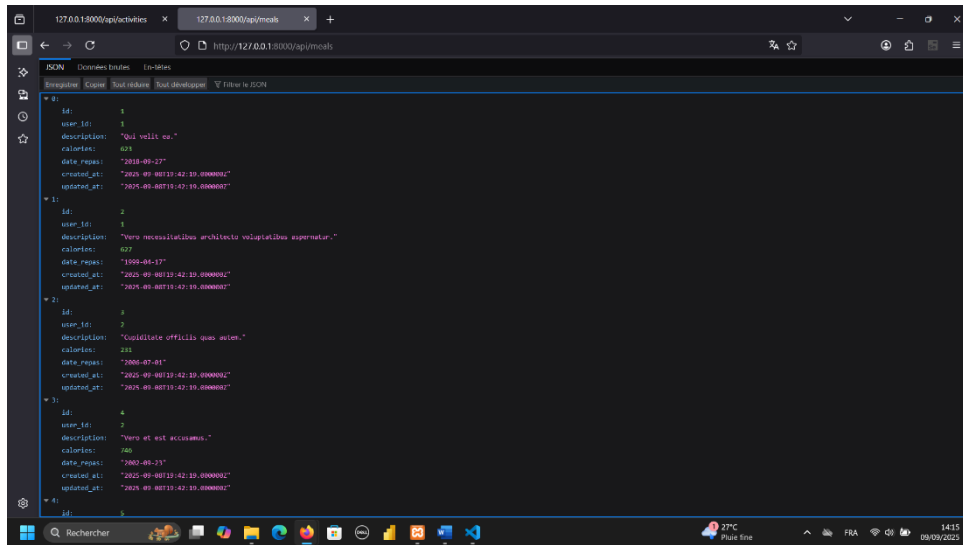
I. Technologies utilisées

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

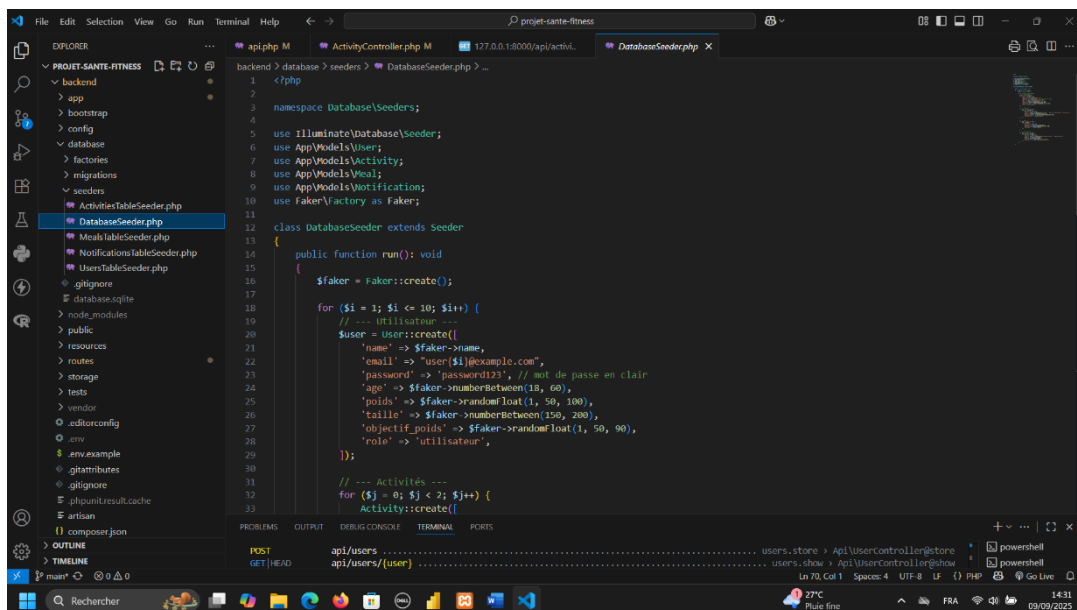
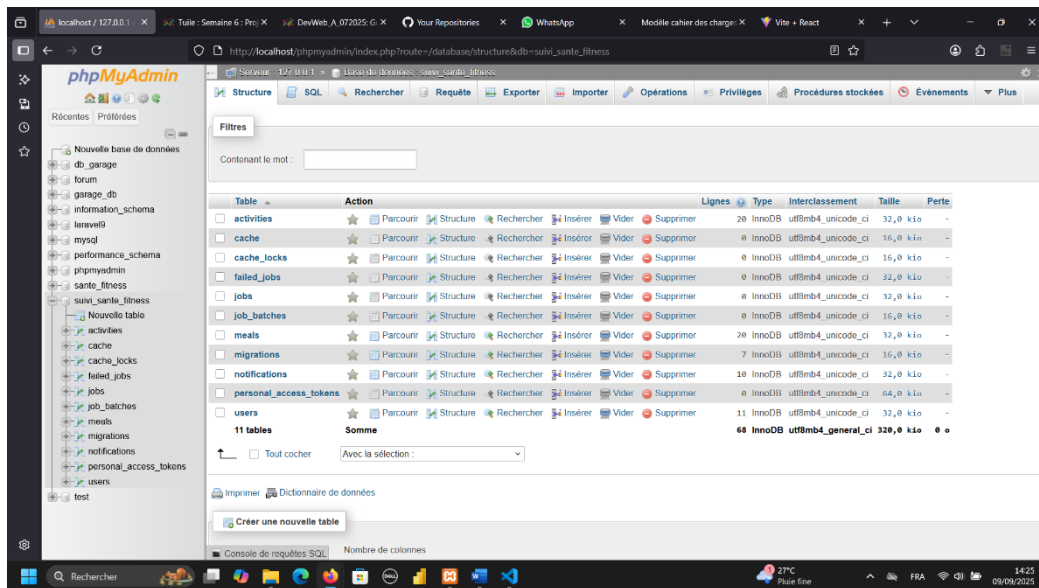
Le back-end de l'application est développé avec Laravel 12.28.1. Il met en place une API REST structurée pour gérer l'ensemble de la logique métier et les échanges de données. Cette API propose des endpoints CRUD pour les utilisateurs, les activités, les repas et les notifications, centralisant ainsi 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.





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, activités, 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.



Affichage des lignes 0 - 10 (total de 11, traitement en 0,0003 seconde(s))

SELECT * FROM "users"

Options supplémentaires

| | id | name | email | password | age | poids | taille | objectif_poids | role | created_at | updated_at |
|--------------------------|----|-----------------------------|--------------------|-------------|-----|-------|--------|----------------|-------------|---------------------|---------------------|
| <input type="checkbox"/> | 1 | Dr. Enos Green III | user1@example.com | password123 | 24 | 61 | 183 | 63.2 | utilisateur | 2025-09-08 19:42:19 | 2025-09-08 19:42:19 |
| <input type="checkbox"/> | 2 | Mrs. Yolanda Breitenberg IV | user2@example.com | password123 | 48 | 74 | 166 | 62 | utilisateur | 2025-09-08 19:42:19 | 2025-09-08 19:42:19 |
| <input type="checkbox"/> | 3 | Osborne Pollich | user3@example.com | password123 | 31 | 51.8 | 176 | 60.6 | utilisateur | 2025-09-08 19:42:19 | 2025-09-08 19:42:19 |
| <input type="checkbox"/> | 4 | Mr. Federico Borer II | user4@example.com | password123 | 54 | 68.7 | 173 | 68.2 | utilisateur | 2025-09-08 19:42:19 | 2025-09-08 19:42:19 |
| <input type="checkbox"/> | 5 | Karelle Ermsr | user5@example.com | password123 | 54 | 58.2 | 177 | 83.2 | utilisateur | 2025-09-08 19:42:19 | 2025-09-08 19:42:19 |
| <input type="checkbox"/> | 6 | Miss Emmalee Denesik | user6@example.com | password123 | 30 | 50.5 | 176 | 78.7 | utilisateur | 2025-09-08 19:42:19 | 2025-09-08 19:42:19 |
| <input type="checkbox"/> | 7 | Haylee Schumm | user7@example.com | password123 | 29 | 78.7 | 163 | 72.3 | utilisateur | 2025-09-08 19:42:19 | 2025-09-08 19:42:19 |
| <input type="checkbox"/> | 8 | Gretchen Kihn | user8@example.com | password123 | 45 | 99.8 | 191 | 86.3 | utilisateur | 2025-09-08 19:42:19 | 2025-09-08 19:42:19 |
| <input type="checkbox"/> | 9 | Amani White | user9@example.com | password123 | 46 | 82.8 | 173 | 53.2 | utilisateur | 2025-09-08 19:42:19 | 2025-09-08 19:42:19 |
| <input type="checkbox"/> | 10 | Donna Simonis | user10@example.com | password123 | 38 | 69.7 | 185 | 73 | utilisateur | 2025-09-08 19:42:19 | 2025-09-08 19:42:19 |

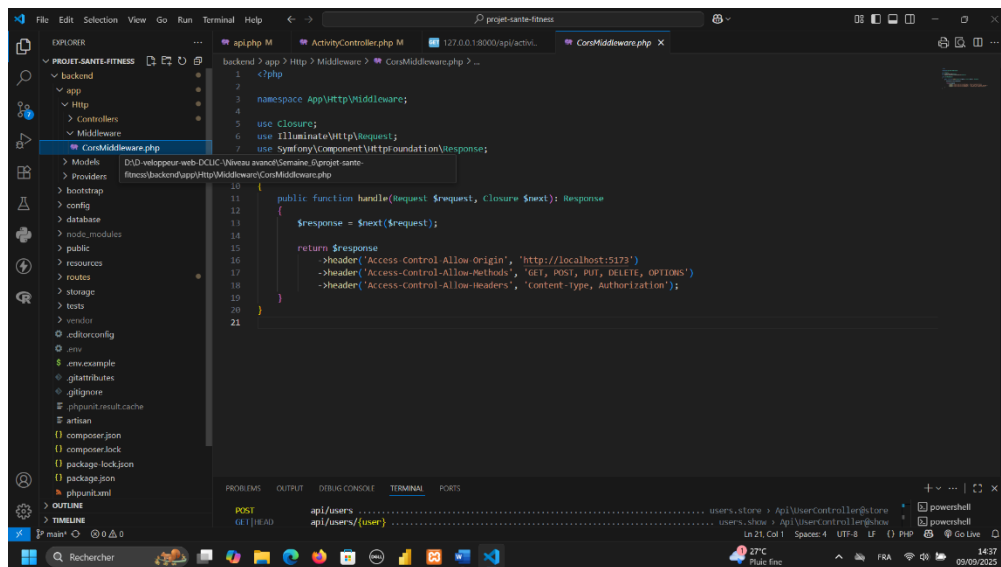
c. Communication Front/Back : CORS

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.

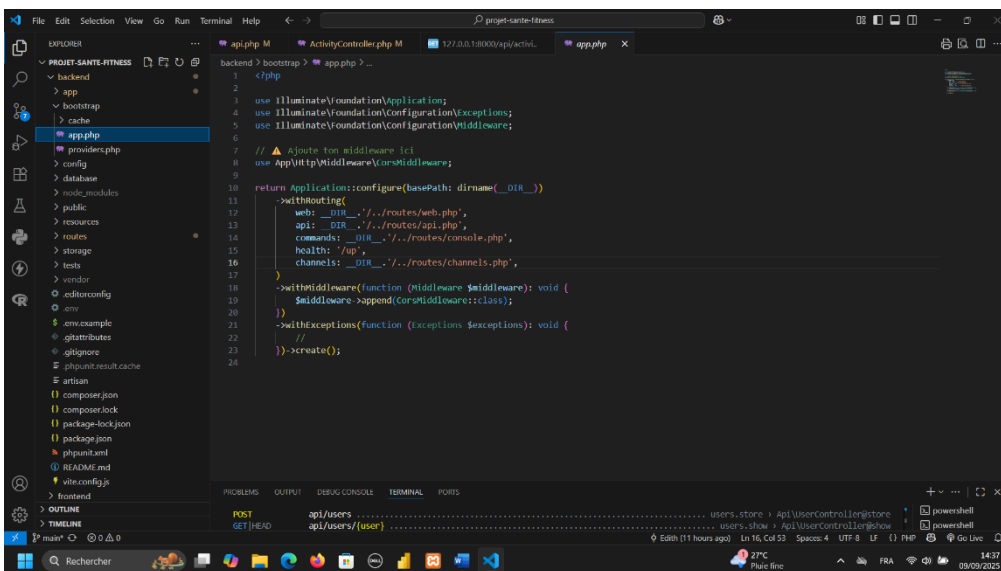
```

1 import axios from "axios"
2
3 const api = axios.create({
4   baseURL: "http://localhost:8080/api",
5   headers: { "Content-Type": "application/json" }
6 })
7
8 // Inject token si présent
9 api.interceptors.request.use((config) => {
10   const token = localStorage.getItem("token")
11   if (token) config.headers.Authorization = `Bearer ${token}`
12   return config
13 })
14
15 export default api
16

```



```
1 <?php
2
3 namespace App\Http\Middleware;
4
5 use Closure;
6 use Illuminate\Http\Request;
7 use Symfony\Component\HttpFoundation\Response;
8
9
10 public function handle(Request $request, Closure $next): Response
11 {
12     $response = $next($request);
13
14     return $response
15         ->header('Access-Control-Allow-Origin', 'http://localhost:5173')
16         ->header('Access-Control-Allow-Methods', 'GET, POST, PUT, DELETE, OPTIONS')
17         ->header('Access-Control-Allow-Headers', 'Content-Type, Authorization');
18 }
19
20
21
```

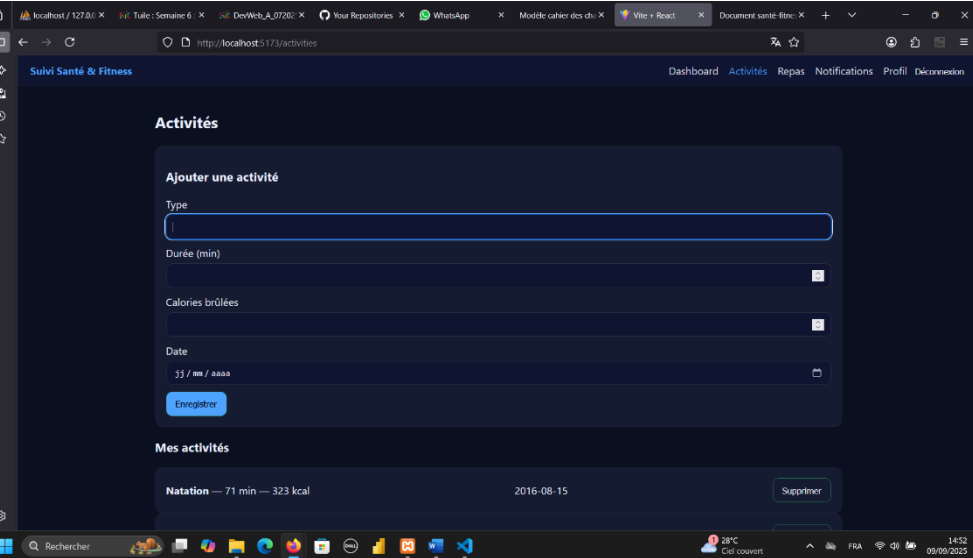
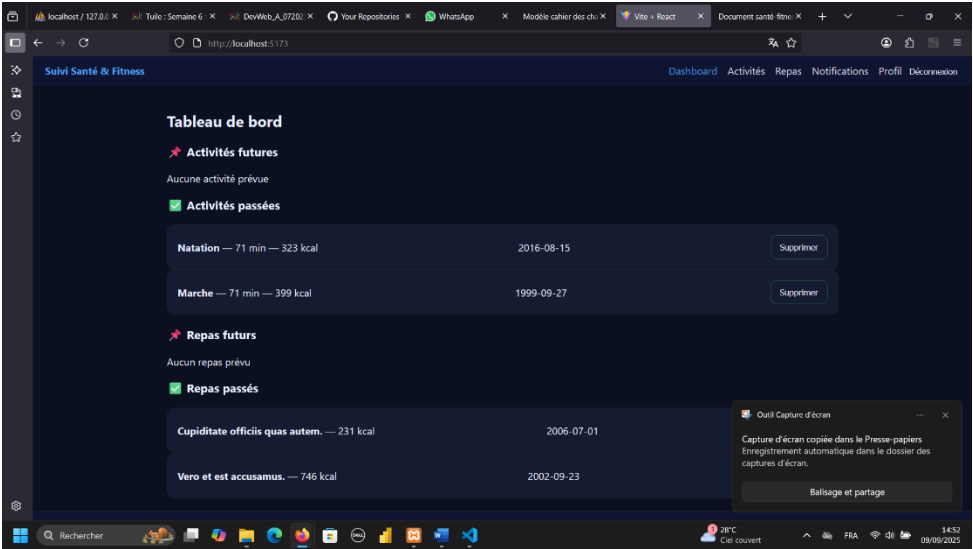
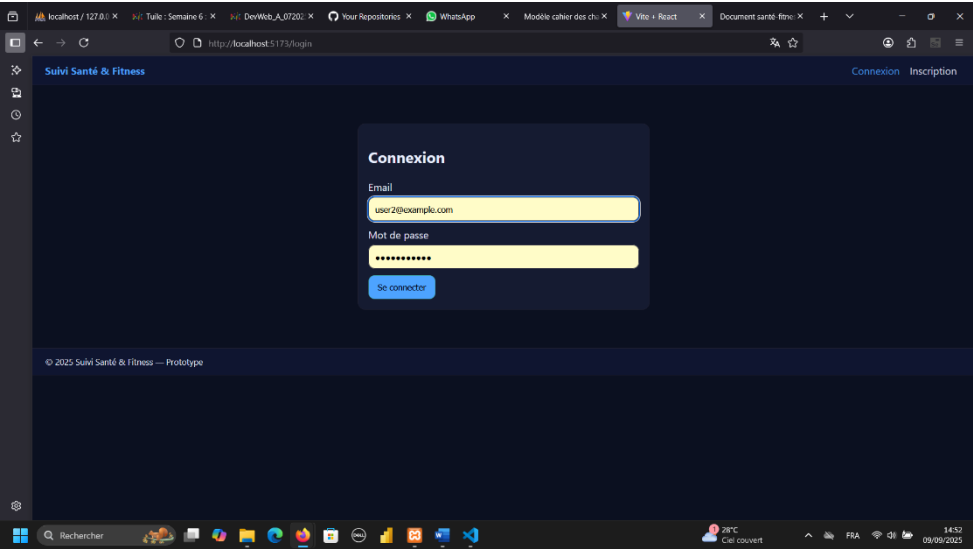


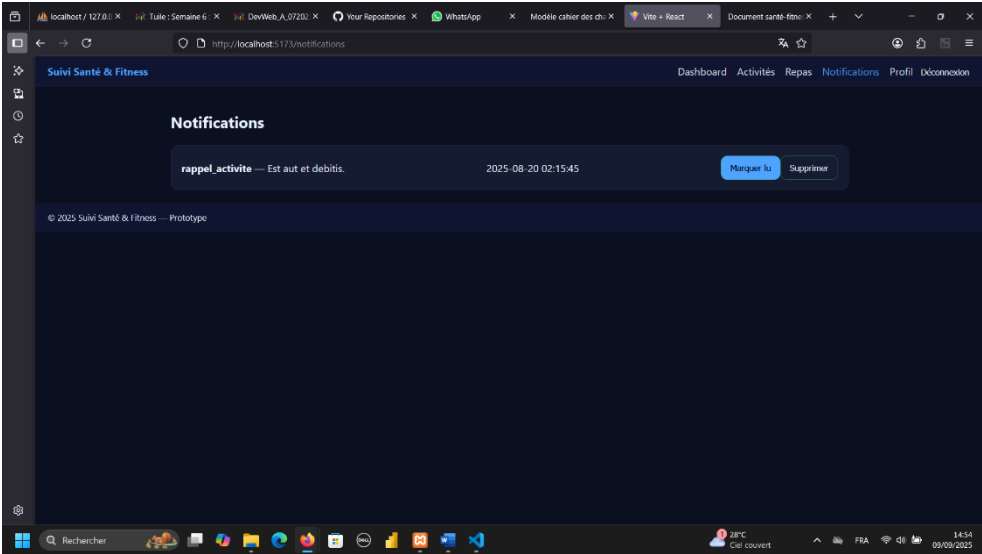
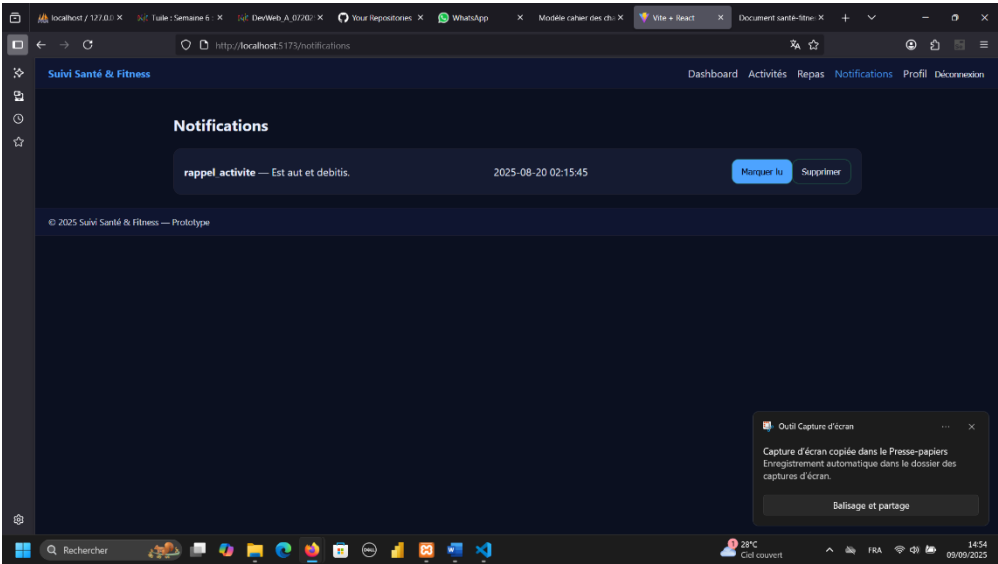
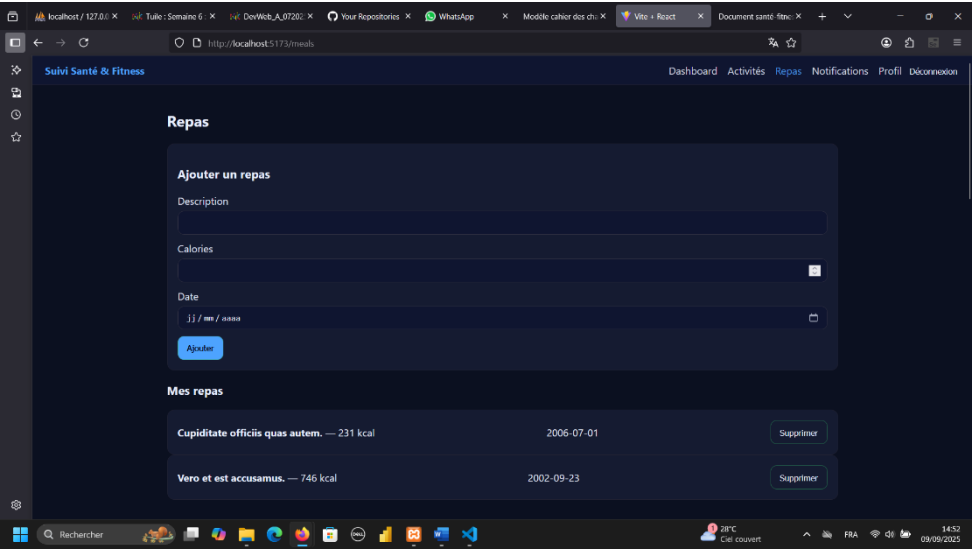
```
1 <?php
2
3 use Illuminate\Foundation\Application;
4 use Illuminate\Foundation\Configuration\Exceptions;
5 use Illuminate\Foundation\Configuration\Middleware;
6
7 // Ajoute ton middleware ici
8 use App\Http\Middleware\CorsMiddleware;
9
10 return Application::configure(basePath: dirname(__DIR__))
11     ->withRouting(
12         web: __DIR__ . '/../routes/web.php',
13         api: __DIR__ . '/../routes/api.php',
14         commands: __DIR__ . '/../routes/console.php',
15         health: '/up',
16         channels: __DIR__ . '/../routes/channels.php',
17     )
18     ->withMiddleware(function (Middleware $middleware): void {
19         $middleware->append(CorsMiddleware::class);
20     })
21     ->withExceptions(function (Exceptions $exceptions): void {
22         //
23     })->create();
24
```

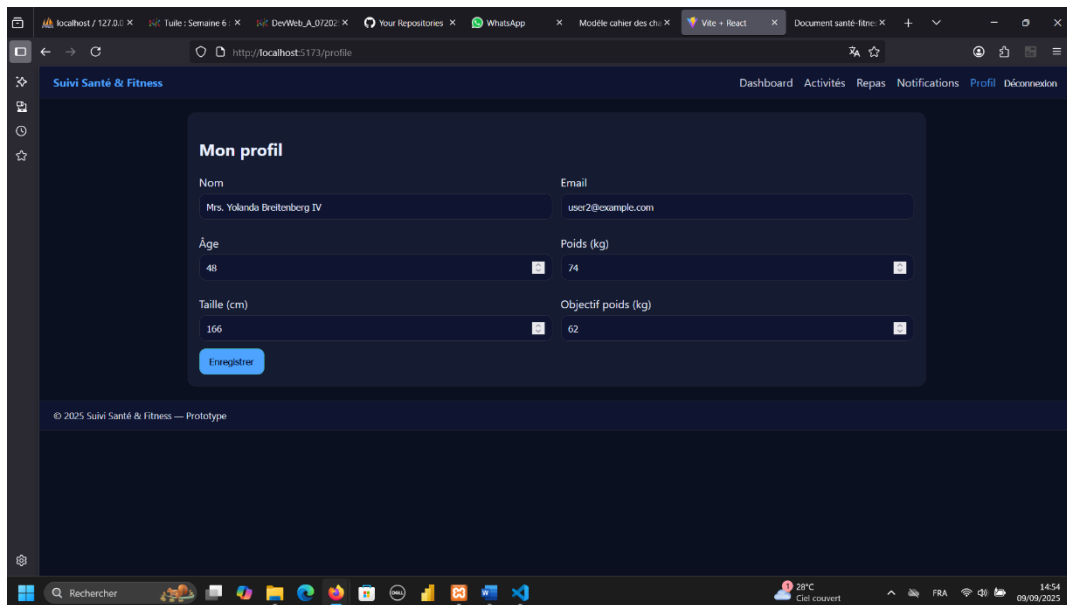
d. Front-end : React.js

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.







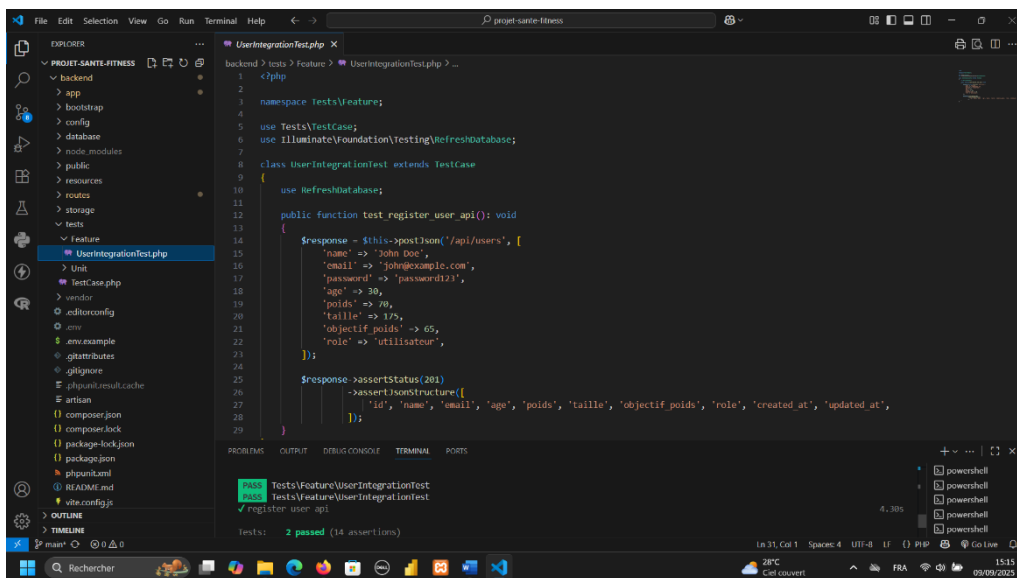
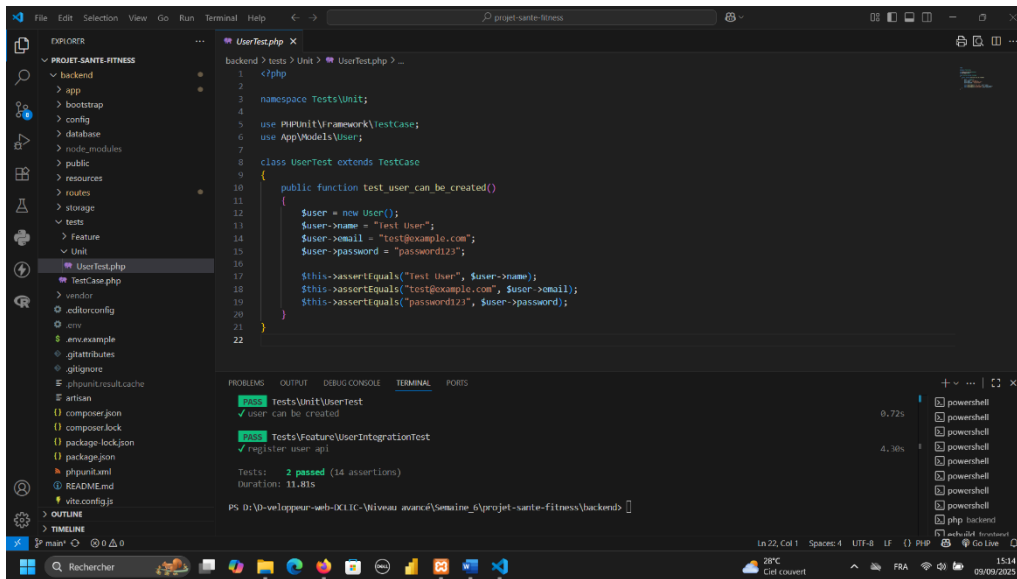
II. Tests Unitaires et d'Intégration

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.



III. Lien vers le dossier complet avec vidéo sur github et google drive

- <https://github.com/edithballogou/projet-sante-fitness>
- https://drive.google.com/drive/folders/1B5WR6akFr7_Ez6ZOip0AO8fUPjeCyZo?usp=sharing

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.

.