Cours développement Web

Niveau approfondi

Modèle cahier des charges

Cahier des charges (DSF) pour la réalisation d'une Application Web

Suivi Santé et Fitness

---

1. Introduction

Ce document a pour but de définir les spécifications fonctionnelles, techniques et les exigences générales pour le développement d'une application web complète. L'application, intitulée « **Suivi Santé et Fitness** », a pour objectif d'aider les utilisateurs à garantir leur bien-etre physique en suivre leurs activités physiques, leur alimentation et leur progression personnelle en matière de santé et de bien-être.

Elle sera développée en utilisant **React** pour la partie front-end (interface utilisateur), permettant une expérience utilisateur interactive et dynamique, et **Laravel** pour la partie back-end (API et gestion des données), assurant la sécurité, la fiabilité et la gestion efficace des informations relatives aux utilisateurs et à leurs activités.

L'application offrira aux utilisateurs la possibilité de suivre leurs objectifs de santé, d'enregistrer leurs activités quotidiennes, de consulter des statistiques personnalisées et de recevoir des recommandations adaptées pour améliorer leur forme physique et leur alimentation.

2. Objectifs du Projet

L'application « **Suivi Santé et Fitness** » vise à aider les utilisateurs à mieux gérer leur santé et leur bien-être. Elle offre un outil interactif pour suivre les activités physiques, l'alimentation et la progression personnelle. Plus précisément, elle :

- Permet de suivre facilement les activités quotidiennes (marche, course, musculation, etc.) et la consommation alimentaire.
- Fournit une vue globale et synthétique de la progression grâce à des statistiques personnalisées.
- Aide à atteindre les objectifs de santé et de forme physique avec un suivi régulier et motivant.

Cette application présente plusieurs avantages : un suivi personnalisé pour chaque utilisateur, des rappels et notifications pour encourager des habitudes saines, ainsi qu'une interface intuitive et accessible même aux personnes peu familières avec les technologies.

Elle a également une finalité pédagogique : elle permet de mettre en pratique des compétences techniques (React pour le front-end, Laravel pour les API, gestion des bases de données), de comprendre la conception d'une application complète et de simuler un cas d'usage réel en respectant les bonnes pratiques du développement web.

En résumé, ce projet allie utilité pratique pour l'utilisateur et apprentissage complet pour le développeur, en créant un outil motivant et fonctionnel pour le suivi de la santé au quotidien.

3. Portée du Projet

Cette première version de l'application se concentre sur les fonctionnalités essentielles pour le suivi santé et fitness :

- Gestion des utilisateurs : inscription, connexion et déconnexion sécurisées.
- Profil utilisateur : saisie et mise à jour des informations personnelles (nom, prénom, email, âge, poids, taille, objectifs de santé).
- Suivi des activités physiques et de l'alimentation : ajout, modification, suppression d'activités et enregistrement des repas avec calcul simple des calories.
- Tableau de bord : visualisation des données sous forme de graphiques simples.
- Notifications simples : rappels pour enregistrer activités et repas.
- API sécurisée avec Laravel pour les opérations CRUD.

Fonctionnalités exclues pour cette version : recommandations personnalisées, suivi avancé avec analyse prédictive, intégration avec appareils externes, partage social et gamification.

Utilisateurs et rôles :

- Utilisateur standard : gère son compte, enregistre et consulte ses données, accède à son tableau de bord.
- Administrateur : supervise les comptes et les données globales.

4. Exigences Fonctionnelles

L'application **« Suivi Santé et Fitness »** comporte plusieurs modules fonctionnels. Côté **front-end avec React**, l'utilisateur peut créer et gérer son compte en renseignant son nom, adresse e-mail, mot de passe et informations personnelles de base (âge, poids, taille, objectifs de santé), se connecter et se déconnecter de manière sécurisée, réinitialiser son mot de passe via un lien envoyé par e-mail, et consulter ou modifier ses informations personnelles et objectifs. Il peut également enregistrer ses activités physiques quotidiennes (type d'activité, durée, calories brûlées), les modifier ou les supprimer, et consulter l'historique de ses activités sous forme de liste ou de tableau. Pour le suivi de l'alimentation, l'utilisateur peut ajouter ses repas

avec description et estimation des calories consommées, modifier ou supprimer des repas, et consulter l'historique alimentaire. Le tableau de bord lui permet de visualiser ses progrès à travers des graphiques (poids, calories consommées, activités réalisées) et de suivre l'avancement de ses objectifs personnels. L'utilisateur reçoit aussi des notifications et alertes personnalisées pour l'encourager à enregistrer ses activités ou repas et pour signaler l'atteinte d'objectifs ou le dépassement de certaines limites. Enfin, un module de recherche optionnel lui permet de filtrer ses activités et repas par date, type ou calories.

Côté **back-end avec Laravel**, l'application assure la gestion sécurisée des utilisateurs et des opérations CRUD pour les activités et repas. L'API permet de gérer l'inscription, la connexion, la réinitialisation des mots de passe et la mise à jour des profils. Elle traite également l'enregistrement, la modification, la suppression et la consultation des activités et repas, et fournit les données nécessaires au tableau de bord et aux statistiques. Pour le rôle administrateur, le back-end permet de consulter, activer, désactiver ou supprimer des comptes utilisateurs, ainsi que de visualiser des statistiques globales sur l'utilisation de l'application.

5. Modèle de Données / Structure des Données

L'application manipulera plusieurs entités principales pour gérer les utilisateurs, leurs activités et leurs repas, avec des relations conçues pour faciliter le suivi des données et la génération de statistiques.

L'entité **Utilisateur** comprend un identifiant unique, le nom, l'email (unique), le mot de passe hashé, l'âge, le poids, la taille, un objectif de poids optionnel, le rôle (utilisateur ou administrateur), ainsi que les dates de création et de modification du compte. Chaque utilisateur peut avoir plusieurs activités et repas, et recevoir plusieurs notifications.

L'entité **Activité** contient un identifiant unique, une clé étrangère vers l'utilisateur, le type d'activité (marche, course, musculation, etc.), la durée en minutes, les calories brûlées, la date de l'activité ainsi que les dates de création et de modification. Chaque activité appartient à un seul utilisateur.

L'entité **Repas** inclut un identifiant unique, une clé étrangère vers l'utilisateur, une description du repas, le nombre de calories consommées, la date du repas et les dates de création et modification. Chaque repas est lié à un seul utilisateur.

L'entité **Notification** (optionnelle pour cette version) contient un identifiant unique, une clé étrangère vers l'utilisateur, le type de notification (ex : rappel activité, rappel repas), le message, l'état (lu/non lu) et la date d'envoi. Chaque notification est associée à un seul utilisateur.

Ainsi, les relations principales sont : un utilisateur peut avoir plusieurs activités, plusieurs repas et plusieurs notifications, tandis que chaque activité, repas ou notification appartient à un seul utilisateur.

6. Exigences Non Fonctionnelles

L'application doit être performante, avec un chargement des pages en moins de 2 secondes, une interface React réactive et des API Laravel répondant en moins de 500 ms. La **sécurité** sera assurée par le hashage des mots de passe, l'authentification sécurisée, et la protection contre CSRF, XSS et injections SQL. L'**ergonomie et convivialité** seront garantis grâce à une interface intuitive, des transitions fluides et une cohérence visuelle. Elle sera **compatible** avec les principaux navigateurs et responsive sur tous les écrans. Le code sera **modulaire et scalable**, optimisé pour la montée en charge, et respectera l'**accessibilité** (textes alternatifs, navigation clavier, contraste adapté). Enfin, les ressources seront **optimisées et éco-responsables** : images compressées, cache intelligent et algorithmes efficaces.

7. Architecture Technique

L'application sera conçue selon une architecture Client-Serveur avec une SPA (Single Page Application). Le front-end, développé avec React, offrira une interface dynamique et réactive, tandis que le back-end, développé avec Laravel, fournira une API REST pour gérer les données et la logique métier. Cette séparation garantit modularité, maintenabilité et évolutivité de l'application.

Côté front-end, React.js sera utilisé comme framework principal avec JavaScript (ou TypeScript pour un typage statique). Les bibliothèques additionnelles incluent React Router pour la navigation, Context API ou Redux pour la gestion de l'état global (données utilisateur, activités, repas), et une UI Library comme Material UI pour accélérer le développement et assurer une interface cohérente. L'objectif est de créer des composants réutilisables et une interface responsive et ergonomique.

Côté back-end, Laravel (version 10 ou supérieure) sera utilisé avec PHP et une base de données MySQL. L'API REST sera organisée autour des ressources principales : /users, /activities, /meals et /notifications, offrant les opérations CRUD correspondantes. Les routes seront sécurisées par l'authentification et l'autorisation basées sur les rôles.

La communication front-end/back-end s'effectuera via HTTP/HTTPS, avec Axios (ou fetch) côté client pour envoyer les requêtes vers les endpoints Laravel. Les données seront échangées au format JSON. Par exemple, lorsqu'un utilisateur saisit une activité dans React, Axios envoie un POST vers /api/activities, Laravel traite la requête, enregistre l'activité en base de données et

renvoie un JSON confirmant la création, puis React met à jour le tableau de bord avec les nouvelles données.

L'environnement de développement inclut Node.js avec Create React App ou Vite pour le front-end, et un serveur Laravel local via Artisan ou Docker pour le back-end. La base de données sera locale (MySQL ou PostgreSQL) via XAMPP, MAMP ou Docker. Les outils complémentaires comprennent VS Code comme IDE, Postman pour tester l'API et Git/GitHub pour la gestion de version.

8. Interface Utilisateur et Expérience Utilisateur (UI/UX)

Le design de l'application sera moderne et minimaliste, afin de garantir une lisibilité optimale et une expérience fluide sur tous les supports. La palette de couleurs reposera sur des tons sobres et apaisants, avec un bleu clair utilisé pour les actions principales, du gris clair pour les arrière-plans et du blanc pour les zones de contenu. La typographie sera basée sur une police sans-serif moderne telle que *Roboto* ou *Open Sans*, favorisant la clarté et l'accessibilité.

Le style visuel suivra plusieurs principes directeurs : une utilisation équilibrée de l'espace blanc pour aérer l'interface, une mise en avant des informations essentielles (tableau de bord, suivi des activités et repas), ainsi qu'une cohérence stricte des couleurs, icônes et boutons sur l'ensemble des écrans.

La navigation reposera sur une structure de type Single Page Application (SPA), gérée par React Router. Les principales vues incluront :

- la page d'accueil / tableau de bord,
- la page de profil utilisateur,
- les pages d'ajout et de modification des activités,
- les pages d'ajout et de modification des repas,
- ainsi qu'une page de notifications.

Un menu latéral ou une barre supérieure permettra d'accéder rapidement à l'ensemble des fonctionnalités, et sera présent sur toutes les pages. Les actions effectuées par l'utilisateur (ajout, modification, suppression) seront accompagnées de retours visuels immédiats, tels que des messages de confirmation ou d'erreur, afin d'améliorer le feedback utilisateur.

L'interface reposera sur des composants UI réutilisables : des boutons standardisés avec un style uniforme pour toutes les actions principales, des cartes d'information pour présenter les activités, repas et statistiques, ainsi que des formulaires dynamiques dotés d'une validation intégrée. Pour la visualisation des données, des graphiques et tableaux interactifs seront proposés, en utilisant des bibliothèques telles que *Chart.js* ou *Recharts*. Enfin, des alertes et

notifications réutilisables permettront de rappeler les événements et d'informer l'utilisateur des actions réussies.

Le parcours utilisateur sera pensé pour être simple et intuitif. Le tableau de bord présentera un résumé des activités et repas récents, accompagné de graphiques de progression et d'accès rapide aux actions principales. La page de profil offrira la possibilité de consulter et modifier les informations personnelles et objectifs. Les pages dédiées aux activités et aux repas incluront des formulaires de saisie, des listes d'historique ainsi que des filtres par date ou type. Enfin, la page notifications affichera l'ensemble des rappels et alertes, avec la possibilité de les marquer comme lus.

Pour compléter cette section, des maquettes (wireframes) pourront être intégrées en annexe afin de représenter visuellement l'interface et les parcours utilisateurs. Elles montreront notamment l'emplacement des menus, des cartes d'information, des formulaires et des graphiques sur les différentes pages de l'application.

9. Flux de Navigation

Cette section décrit les parcours principaux de l'utilisateur à travers l'application, en détaillant les étapes nécessaires pour accomplir les actions essentielles liées au suivi des activités physiques, de l'alimentation et des objectifs personnels.

Scénario 1 : Inscription et connexion

Un nouvel Utilisateur peut créer un compte en accédant à la page d'accueil et en sélectionnant l'option « S'inscrire ». Il renseigne ses informations personnelles (nom, e-mail, mot de passe, âge, poids, taille, objectifs) puis valide le formulaire. Le compte est créé et l'utilisateur est automatiquement connecté ou redirigé vers la page de connexion. Il peut ensuite se connecter avec ses identifiants et accéder à son tableau de bord personnalisé.

Scénario 2 : Ajouter une activité physique

Depuis son tableau de bord, l'utilisateur accède à la section « Activités » et choisit l'option « Ajouter une activité ». Il complète le formulaire avec le type d'activité, la durée, les calories brûlées et la date correspondante. Une fois enregistré, l'historique est mis à jour et les statistiques affichées sur le tableau de bord reflètent immédiatement cette nouvelle activité.

Scénario 3 : Ajouter un repas

Pour assurer le suivi de son alimentation, l'utilisateur se rend dans la section « Repas » et sélectionne « Ajouter un repas ». Il renseigne la description ainsi que les calories consommées,

puis enregistre les informations. Le repas est ajouté à l'historique et les données du tableau de bord sont automatiquement mises à jour afin de suivre l'évolution des apports caloriques.

Scénario 4 : Consulter le tableau de bord

À tout moment, l'utilisateur peut consulter son tableau de bord afin d'avoir une vue d'ensemble sur ses activités physiques et son alimentation. Il y visualise les activités récentes et les calories brûlées, les repas enregistrés et les calories consommées, ainsi que sa progression par rapport aux objectifs fixés. Les notifications ou rappels y sont également accessibles et permettent un accès direct aux pages concernées.

Scénario 5 : Modifier le profil utilisateur

L'utilisateur a la possibilité de mettre à jour ses informations personnelles ou ses objectifs en se rendant dans la section « Profil ». Après avoir modifié les champs souhaités (poids, taille, objectifs, etc.), il enregistre les changements. Ces derniers sont immédiatement pris en compte et reflétés dans son tableau de bord et ses statistiques.

10. Échéancier et Plan de Travail

Semaine 1 : Conception et préparation

- Jours 1-2 : Choix de l'idée et brainstorming
  - Définition du projet, identification des besoins des utilisateurs.
  - Brainstorming sur les fonctionnalités essentielles et non fonctionnelles
- Jour 3 : Rédaction du Cahier des Spécifications Fonctionnelles (DSF)
  - Rédaction des sections : introduction, objectifs, portée, fonctionnalités principales.
- Jour 4 : Validation du DSF avec le tuteur
  - Présentation du DSF.
  - Ajustements des fonctionnalités et priorités selon les retours.
- Jours 5-7 : Conception détaillée
  - Modèle de données : Création des migrations pour les entités Utilisateur, Activité, Repas, Notification.
  - Architecture Front-end : Structure des composants React, routes avec React Router, choix de bibliothèques UI.
  - Architecture Back-end : Routes API, contrôleurs, sécurité et gestion des rôles.
  - Wireframes / maquettes : Conception des écrans principaux (tableau de bord, activités, repas, profil, notifications).

Semaine 2 : Développement et finalisation

- Jours 8-9 : Développement Front-end
  - Création des composants React pour les modules : Gestion des utilisateurs, Activités, Repas, Tableau de bord, Notifications.
  - Mise en place de la navigation avec React Router.
  - Intégration initiale des appels API via Axios (sans logique complète côté serveur).
- Jours 10-11 : Développement Back-end
  - Mise en place des migrations et modèles Laravel.
  - Création des contrôleurs et routes API pour gérer les opérations CRUD sur Utilisateur, Activité, Repas, Notifications.
  - Implémentation de la logique métier (calcul des calories, gestion des statistiques).
- Jour 12 : Intégration Front-End / Back-End
  - Connexion finale entre React et Laravel.
  - Vérification que les données circulent correctement entre les composants et l'API.
  - Gestion de l'état global dans React (Context API ou Redux).
- Jour 13 : Tests
  - Tests fonctionnels : Parcours utilisateur complets (ajout activité, ajout repas, consultation tableau de bord).
  - Tests d'intégration : Vérification des échanges Front <-> Back.
  - Identification et correction des bugs.
- Jour 14 : Optimisation & Sécurité
  - Optimisation des performances Front-end et Back-end (réduction des requêtes, lazy loading, cache).
  - Vérification des aspects sécurité : authentification, validations, protection contre XSS/CSRF/SQL Injection.
  - Revue des points d'éco-responsabilité (optimisation des images, réduction des appels API inutiles).
- Jour 15 : Finalisation & Présentation
  - Préparation de la démo et des captures d'écran.
  - Documentation rapide sur l'installation, l'utilisation et les principales fonctionnalités.
  - Dernières corrections avant livraison.

11. Plan de Validation et Tests

Afin de garantir que l'application fonctionne correctement et respecte toutes les exigences définies dans le DSF, plusieurs types de tests seront mis en place selon une approche progressive. Les tests unitaires permettront de vérifier le bon fonctionnement des composants React isolés, tels que les formulaires d'ajout d'activité ou l'affichage des cartes repas, ainsi que des fonctions PHP et méthodes des contrôleurs Laravel, comme le calcul des calories ou la validation des données reçues.

Les tests d'intégration vérifieront la communication entre le front-end et le back-end, par exemple lorsqu'un composant React envoie une requête POST pour ajouter une activité et reçoit la réponse JSON attendue de Laravel, tout en s'assurant que les données sont correctement enregistrées en base et affichées à l'utilisateur.

Les tests fonctionnels et systèmes simuleront des parcours utilisateurs complets, incluant l'inscription, la connexion et la déconnexion, l'ajout, la modification et la suppression d'activités et de repas, la consultation du tableau de bord et des statistiques, ainsi que la modification du profil utilisateur. Ces tests permettront de vérifier que toutes les fonctionnalités respectent les exigences définies dans le DSF.

La stratégie de test combinera des tests manuels, réalisés selon les parcours utilisateurs définis, et des tests automatisés réalisés avec Selenium pour simuler et valider les interactions de l'utilisateur. Des outils comme Jest ou React Testing Library pourront être utilisés pour tester les composants React, et PHPUnit pour tester les contrôleurs et modèles Laravel. Les anomalies détectées seront documentées et corrigées avant l'intégration finale.

La validation du DSF consistera à associer chaque fonctionnalité définie dans le cahier des charges à un ou plusieurs tests fonctionnels. La validation vérifiera que toutes les fonctionnalités prioritaires sont implémentées et opérationnelles, que les exigences non fonctionnelles (performance, sécurité, UI/UX, compatibilité) sont respectées, et que les parcours utilisateurs se déroulent correctement sans erreurs bloquantes. Une checklist finale permettra de cocher chaque exigence validée afin de s'assurer que le produit livré correspond exactement au cahier des charges.

---

12. Déploiement et Environnement d'exécution

L'application sera exécutée et présentée dans un environnement de production simulé afin de faciliter la démonstration de ses fonctionnalités. Pour cette version initiale, elle pourra être déployée localement sur un ordinateur ou sur un serveur distant simple, ce qui permettra de

montrer l'ensemble des modules, du tableau de bord aux notifications, dans des conditions proches de la production.

Les prérequis techniques pour exécuter l'application incluent l'installation des outils et environnements nécessaires pour le front-end et le back-end. Côté front-end, Node.js et un gestionnaire de paquets comme npm ou yarn seront requis pour lancer le serveur de développement React (via Create React App ou Vite) et gérer les dépendances. Côté back-end, PHP et Composer seront nécessaires pour exécuter Laravel, ainsi qu'une base de données MySQL ou PostgreSQL pour stocker les utilisateurs, activités, repas et notifications.

Pour la démonstration, l'environnement pourra être configuré avec Laravel Artisan pour lancer le serveur local du back-end, et Node.js pour le front-end, garantissant ainsi une interaction complète entre React et Laravel. L'application sera accessible via un navigateur web moderne, permettant de présenter de manière interactive les différents parcours utilisateurs et fonctionnalités implémentées.

13. Références

Pour l'élaboration du projet santé et fitness décrit précédemment, plusieurs ressources ont été consultées pour s'assurer de la conformité aux bonnes pratiques de développement, d'architecture et d'ergonomie. Voici les principales sources d'inspiration, tutoriels suivis, bibliothèques externes utilisées et documentations consultées :

- **D-CLIC – Organisation internationale de la Francophonie (OIF)** : Cours et documentations
- Tutoriel suivi : https://www.youtube.com/watch?v=_ZOcTXsdWn8
- Cours linkedInhttps://www.linkedin.com/learning/rediger-un-cahier-des-charges-de-site-internet/definir-un-cahier-des-charges?autoSkip=true&resume=false&u=69962338