Fit Buddy

Cahier des charges

29 NOVEMBRE 2025 – V2

1. Sommaire

1. Introduction

1.1. Contexte du Projet

Dans le cadre d'un projet entre l'ESILV et l'entreprise Fit Buddy, les équipes d'étudiants ingénieurs devront développer un coach sportif IA sur une application mobile. L'entreprise Fit Buddy vise à connecter chaque adhérent à ses objectifs personnels grâce à une expérience d'entraînement personnalisée, valorisante et motivante. Chez Fit Buddy, nous pensons que s'entraîner doit redevenir un plaisir, pas une contrainte. C'est pour ces raisons que nous souhaitons connecter tous les adhérents de salle de sport à leurs objectifs par une expérience personnalisée, valorisante et motivante. Pour répondre à cela, l'objectif est de proposer un service de coaching sportif avec la création de programme, le suivi et l'analyse des performances pour les sportifs de tous niveaux. Grâce aux dernières avancées sur l'intelligence artificielle et à des fonctionnalités avancées comme le LLM, l'application vise à :

- Fournir un accompagnement personnalisé (programmes sportif, accompagnement et ajustement de séances, conseils, réponses aux utilisateurs).
- Suivre en temps réel les performances et l'évolution de l'utilisateur.
- Offrir une expérience interactive et motivante (gamification, communauté).

1.2. Objectifs et Résultats Attendus

L'objectif dans un premier temps est de proposer un prototype se concentrant sur la création de programmes, l'accompagnement sur les séances de sport, le suivi simplifié et l'analyse des données. Il y aura 2 objectifs lors de la livraison du premier prototype. Le premier sera de tester la qualité et la précision du modèle de l'application avec des coachs professionnels et le second sera de fournir une expérience utilisateurs simple et fluide avec des sportifs de tous niveaux.

Plusieurs équipes travaillent sur le projet (3 équipes au total pour l'application complète). Il est essentiel de bien se coordonner entre les équipes car elles auront devront développer un prototype Web App commun puis un prototype chacun sur Android et iOS. Les fonctionnalités auront différents niveaux de priorité (3 au total dans un premier temps). Le niveau de priorité 1 devra être validé par l'entreprise afin de passer au niveau suivant. Le but du premier prototype est qu'au minimum le niveau 1 de priorité soit complété par toutes les équipes et que l'ensemble fonctionne de manière cohérente. La date de livraison des fonctionnalités de priorité 1 est le 19 décembre 2025.

L'objectif ensuite sera d'itérer selon les retours des experts et utilisateurs tout en développant les fonctionnalités de priorité 2 et ainsi de suite.

1.3. Personas

L'application Fit Buddy s'adresse à plusieurs profils de sportifs en salle, dont trois segments prioritaires pour le MVP. Chaque persona correspond à une combinaison typique de **niveau**, **motivation**, **disponibilité** et **objectif**. Le coach IA devra s'adapter dynamiquement à ces profils.

- Persona 1 : « Alex, le débutant motivé pour la prise de masse »

- Âge / profil : 18-28 ans, étudiant ou jeune actif, 0 à 1 an de pratique.

- Motivations : gagner en muscle et confiance, suivre un plan clair sans se perdre dans les machines.
- Freins : manque de méthode, peur du jugement, découragement rapide, peu de compréhension des charges et des tempos.
- Besoins clés :
    - guidance pas à pas avec explications simples
    - recommandations automatiques de charges et temps de repos
    - visualisation des progrès pour maintenir la motivation.

- Persona 2 : « Sofia, la débutante en quête de forme et perte de poids »

- Âge / profil : 18-28 ans, débute le fitness, veut perdre du poids et remodeler sa silhouette.
- Motivations : mieux se sentir dans son corps, suivre un programme adaptable à son rythme.
- Freins : incertitude sur la bonne exécution, peur du jugement, découragement rapide, peur des machines complexes.
- Besoins clés :
    - interface intuitive et rassurante
    - suivi simplifié de la progression (temps, calories, effort)
    - messages de renforcement positif du coach IA.

- Persona 3 : « Karim, le jeune actif pressé »

- Âge / profil : 28-38 ans, salarié ou entrepreneur, niveau intermédiaire, 2–3 séances/semaine.
- Motivations : maintenir sa forme, se sentir bien, optimiser son temps d'entraînement.
- Freins : emploi du temps chargé, oublis, frustration si séance non optimisée.
- Besoins clés :
    - programmes courts et adaptatifs selon ses préférences d'exercices,machines et temps disponible
    - suivi automatique de ses séances
    - retour rapide sur performance et recommandations ciblées

L'application devra également s'adapter à des utilisateurs dont l'objectif principal est l'amélioration de la performance, quel que soit leur niveau initial. Ces utilisateurs peuvent être débutants, intermédiaires ou avancés, mais partagent une même recherche d'amélioration de leurs capacités physiques, qu'il

s’agisse de l’endurance, de la force, de la puissance, de la mobilité, de la coordination ou de la gestion de l’effort. Fit Buddy devra leur proposer des programmes adaptés à leur objectif et un suivi structuré, des indicateurs clairs de progression avec une adaptation dynamique des programmes et séances en fonction des données de performances collectées. Par ailleurs, l’application devra aussi anticiper les besoins d’un autre profil : les utilisateurs experts ou sportifs confirmés, déjà autonomes dans leur entraînement et cherchant avant tout un outil de suivi automatique et d’analyse fine. Ces derniers disposent de leur propre routine et attendent surtout de Fit Buddy une fiabilité dans la mesure, une visualisation avancée des performances, et la possibilité d’exploiter leurs données pour optimiser leurs cycles d’entraînement. Bien que ces usages dépassent le périmètre du MVP, ils devront être pris en compte dans la conception modulaire de la base de données et de l’architecture IA afin de permettre, à terme, l’ajout de fonctionnalités d’analyse de performance et de personnalisation avancée sans refonte majeure du système.

2. Organisation du Projet

2.1. Équipes Rôles et Responsabilités

Les équipes travailleront en autonomie, chaque équipe devra maintenir son propre **dépôt GitHub** (un par équipe) et définir des **contrats d’API (Swagger/OpenAPI)** clairs avant tout développement.

Priorité Absolue : Le déploiement d’une **Web App (MVP)** construite avec des primitives **React Native** (Expo) pour faciliter la transition ultérieure vers les applications natives.

- Equipe 400 (référent Gaspard Lebreton) : Front et IA
  - gaspard.lebreton@edu.devinci.fr
  - maxence.bothorel@edu.devinci.fr
  - johan.kobana@edu.devinci.fr
  - mathieu.maury@edu.devinci.fr
- Equipe 453 (référent Gaspard Breton) : Data et API
  - gaspard.breton@edu.devinci.fr
  - ayhan.bolu@edu.devinci.fr
  - luka.genot@edu.devinci.fr
  - henri.passani@edu.devinci.fr (support documentation RAG et logique métier avec les 2 autres équipes)
  - jules.wejroch@edu.devinci.fr
- Equipe 531 (référent Lucas Barrez) : Capteurs et logique métier
  - lucas.barrez@edu.devinci.fr
  - Gabriel.carlotti@edu.devinci.fr
  - Nassim.lamnini@edu.devinci.fr
  - Ugo.monneau@edu.devinci.fr
  - Harrisson.tevoedjre@edu.devinci.fr
  - Gregoire.woroniak@edu.devinci.fr
  - lucas.lopes@edu.devinci.fr

Voici la répartition des responsabilités du projet pour la web app :

- **Mission de l'équipe Front et IA** : Construire l'interface utilisateur et fournir toute la matière textuelle (prompts et documentation) qui sert de "cerveau" au coach IA.
  - **Intégration UI/UX (Pixel Perfect)** : Développement des écrans (Screens) en respectant strictement les tokens Figma (couleurs, typos) et gestion de la navigation (Expo Router).
  - **Player de Séance (Cœur du Front)** : Développement du chronomètre, de la logique de validation des séries (cocher les cases), et de l'affichage des médias (images/vidéos) .
  - **Interface Chatbot (Stateless)** : Création de la fenêtre de chat (bulles, input) et affichage des réponses de l'IA envoyées par le Backend.
  - **Gestion d'État (Affichage) :** Utilisation de React Query/Context pour afficher les données reçues (Programme, Profil) sans effectuer de calculs métier (Architecture Server-Driven).
  - **Prompt Engineering (Contenu)** : Définition des "System Prompts" (la personnalité du coach) et des templates de questions à envoyer au Backend. Note : Il s'agit **uniquement des textes** du prompt, mais ne codent pas l'appel API vers le LLM.
  - **Documentation RAG (mémoire du coach)** : rédaction et structuration des fichiers markdown qui serviront de base de connaissances :
    - principes de programmation sportive,
    - fiches détaillées d'exercices,
    - méthodes d'entraînement,
    - explications des métriques (volume, intensité, RPE…),
    - synthèse des études / bonnes pratiques.

    Il est attendu un fichier RAG propre prêt à être ingéré par l'équipe backend.
  - **Documentation des besoins API :** L'équipe Front doit fournir un dossier /api-spec-front/ contenant un fichier Markdown par écran/flow (ex. home.md, profile.md, program.md, session.md, chat.md, stats.md) décrivant les données nécessaires, la structure JSON attendue, les actions/mutations attendues, les erreurs à gérer et les endpoints nécessaires sous forme de contrats de besoins. L'équipe Data et API reste libre de l'implémentation interne tout en garantissant la conformité à ces contrats.

- **Mission de l'équipe Data et API** : être la colonne vertébrale qui sécurise les données, gère les utilisateurs et connecte les 3 équipes.
  - **API Gateway & Routing : Point d'entrée unique.** Redirige les requêtes du Front soit vers la BDD, soit vers le service LLM, soit vers les services de l'équipe capteur.
  - **Authentification & User Management :** Gestion de l'inscription/connexion (Email/Password uniquement pour la V1) et sécurisation des routes (JWT).
  - **Pipeline de création IA :**
    - Ingestion des documents RAG, construction du megaprompt et appel à l'API
    - Gère les clés API du LLM et les appels techniques.
    - Crucial : Reçoit le JSON brut de l'IA, le vérifie/nettoie (via Zod ou Typebox par exemple) pour s'assurer qu'il ne fera pas planter l'appli, puis l'insère proprement en base de données.
  - **Base de Données (PostgreSQL) :** Conception du schéma de BDD selon besoins projet et autres équipes et maintenance du schéma relationnel (Users, Programs, Exercises, Sessions).
- **Mission de l'équipe logique métier et capteurs :**
  - **Rattachement des données capteurs :** récupérer, valider et stocker les données des capteurs et associer ces données à l'utilisateur, la séance et l'exercice corrects dans l'application. Pour le premier prototype, des données simulées seront utilisées.

- **Modèle de prédiction attente :** Développer le modèle d'estimation du temps d'attente des machines
- **Ajustement Dynamique de Séance** : Implémenter la logique de modification de la séance en cours en fonction de la disponibilité des machines et du temps d'attente estimé par le modèle.
- **Service d'Interopérabilité :** Fournir une API propre pour que l'équipe Data et API puisse interroger la logique d'ajustement et de prédiction d'attente car **le front n'interrogera que l'API Gateway développé par l'équipe Data et API.**
- **Ajustement du Programme & des Séances (Logique Métier) :** Implémentation des algorithmes simples pour mettre à jour les séances et programmes selon les performances de l'utilisateur ou la disponibilité/l'attente estimée des machines (logiques métiers en dur, sans appel LLM).
- **Définition des contrats de données :** Pour toutes les fonctionnalités d'ajustement de séance et de programme (en fonction des performances de l'utilisateur et de la disponibilité/attente estimée des machines), l'équipe Logique Métier & Capteurs est responsable de définir précisément les données nécessaires qui doivent être exposées par l'API de l'équipe Data & API.

2.2. Outils de Collaboration (GitHub, Teams, Excel Template)

- **GitHub** : gestion du code, issues, pull requests. Attention veillez à utiliser votre compte pour ajouter du code afin de faire un suivi sur la contribution du projet pour votre CTS.
- **Microsoft Teams** : points, réunions, canaux de discussion.
- **Template Excel** : Un fichier template sera utilisé pour centraliser les questions de chaque équipe afin que vous ayez tous le même niveau d'information par la suite. Les questions sont à envoyer avant les réunions pour optimiser le temps et la clarté.

- **Notion** : créer un projet unique qui centralise le pilotage : lots & sous-objectifs, répartition des tâches (RACI), Kanban + calendrier des jalons, suivi risques/dépendances, décisions, CR de réunion et dépôt des livrables/ressources. Chaque membre met à jour ses tâches avant le point prévu chaque 2 semaines.

2.3. Méthodologie

Le projet s'organisera sous forme de sprints de 2 semaines. Un calendrier commun détaillant les objectifs pour chaque fonctionnalité et groupe technique lors de chaque de chaque sprints est à mettre en place au début du projet sur Notion (possible de proposer un autre outil en début de projet, à faire valider par Fit Buddy). Les avancements, livrables et rétrospectives seront présentés à chaque fin de sprint jusqu'à la livraison du projet.

2.4. Dates Clés (Livraison du Prototype, Versions suivantes)

- **19 décembre** : livraison du **premier prototype fonctionnel en web app** (minimum priorités 1 achevées).
- Versions suivantes : développement des priorités 2, roadmap détaillée à définir après retours internes et bêta-tests.

3. Fonctionnalités de l'Application

3.1. Création du Profil Utilisateur

- **Création de compte / Authentification** : e-mail et mot de passe
- **Questionnaire adaptable :**
    - Définir environ 10 questions obligatoires et communes à tous les utilisateurs puis définir environ 10 à 15 autres questions possibles à passer (individuellement ou complètement) adaptées selon les objectifs des utilisateurs (5 catégories : prise de masse, perte de poids, amélioration de la force et/ou endurance, bien être et remise en forme, préparation à un sport ou une compétition).
    - Fichier excel en annexe proposant des questions types à utiliser. Le choix des questions devra se faire en accord avec les modèles de coach virtuel créés afin de définir les questions ayant le plus d'impact sur le programme et d'adapter les questions à poser selon les réponses de l'utilisateur. La liste complète des questions devra être validée par Fit Buddy avant développement.
- **Reprise du questionnaire**
    - Dans le cas où l'utilisateur ne répond pas à la totalité des questions. Il sera proposé d'y répondre plus tard. Les questions posées seront toujours adaptées aux précédentes réponses. La proposition de reprise du questionnaire devra revenir sous forme de fenêtre pop-up.

3.2. Home page

Selon l'utilisateur, la page d'accueil devra afficher la salle de sport auquel le sportif est rattaché. Cette page d'accueil devra afficher les informations suivantes :

- Informations sur la salle (Nom, horaires, adresse, photo).
- Objectif principal de l'utilisateur et information clé de son suivi par rapport à l'objectif
- Phrase motivante/conseil adapté proposé par le RAG prompting du coach virtuel
- Calendrier de la semaine
- Encart sport avec résumé de la séance du jour (objectif de la séance, durée)

3.3. Création Automatique d'un programmes sportif

Le développement des fonctionnalités autour du coaching sportif s'inspirera d'applications comme Zing, Freeletics, FitnessAI ou équivalent. L'objectif sera de proposer des programmes sportifs adaptés à chaque type de profils et objectifs. Les différences principales avec les applications existantes seront une adaptabilité large et précise sur les sportifs de tous niveaux, la mise en relation avec les informations du profil utilisateur, l'ajustement du programme selon l'évolution et les données des machines de musculation connectée (connexion réelle avec les capteurs en priorité 2, à faire avec données simulée en priorité 1). L'application devra reprendre la totalité des informations du profil et des réponses au questionnaire puis générer des programmes sportifs adaptés à chaque utilisateur indiquant à minima les informations suivantes :

- **Planning Général**
  - Durée des phases(ou cycles) principales et sous-phases (en semaines ou mois).
  - Nom, objectifs et sous-objectifs des phases et sous-phases. (poids, force, endurance…)
  - Nombre de séances hebdomadaires et répartition (ex. 2 musculation, 2 cardio, 1 mobilité).
  - Progression : comment les charges, distances ou intensités doivent évoluer d'une semaine à l'autre.
  - Lieux et équipements nécessaires
- **Séances Détail**
  - Nom de la séance (ex. "Pectoraux - Épaules", "Endurance Course", etc.).
  - Liste des exercices :
    - Noms (Squats, Développé couché, Fractionné 30/30, etc.).
    - Temps global de la séance
    - Nombres de séries et d'exercices
    - Séries, répétitions, ou temps (pour le cardio) + charge approximative (%1RM ou intensité) + temps de repos.
    - Description de l'exercice
    - Objectifs et/ ou muscles visés lors de l'exercice
    - Points clés de la technique : consignes de sécurité, posture, alignement, etc.
  - Lieux et équipements nécessaires
- **Conseils de Récupération et Sécurité**
  - Jours de repos ou séances légères (yoga, mobilité).
  - Rappels sur l'échauffement, la prévention des blessures.

Afin de simplifier le développement du prototype, la création de programme se fera en suivant la méthodologie suivante :

1. **Templates de programmes par objectif (template développé avec LLM en amont)**

- 1 template "prise de masse débutant"
- 1 "perte de poids débutant"
- 1 "intermédiaire prise de masse"
  etc.
  → stockés en JSON ou dans la BDD.

2. **LLM utilisé uniquement pour :**

- mettre des mots :
  - nom de phase
  - nom de séance
  - texte d'explication
  - recommandations de progression

- ajuster légèrement les paramètres (par ex. jouer sur 2–3 variables : nombre de séances, volume total, répartition haut/bas)

3. **Structure JSON du programme définie à la main (schema TypeScript + Zod)**
   → le LLM **ne construit pas la structure** : il remplit des “blanks” et donnes des conseils, recommandations et phrases motivantes lors des différentes phases (création de programme, séance du jour, adaptation du programme ou de la séance…).

3.4. Accompagnement sur séance de sport en direct

La Home page présentera synthétiquement l’entraînement du jour. En cliquant sur cet encart, la page dédié au sport présentera avec un niveau de détail plus avancé le sous-objectif actuel et la séance du jour. Dans cette fonctionnalité, l’utilisateur accède à un “Entraînement du Jour” présenté dans la section sport pour connaître tout le détail. Une synthèse présentera synthétiquement la séance avec les données suivantes :

- Durée estimée de la séance et titre (ex. “Séance Force Bas du Corps”),
- Image illustrant les principaux muscles ciblés,
- Aperçu rapide des exercices (liste brève : noms, nombre total d’exercices, objectifs),
- Lieu (salle de sport, extérieur, domicile),
- Message de bienvenue et d’encouragement,
- Bouton de personnalisation pour adapter la séance aux besoins.

Personnalisation/modification de la séance

En cliquant sur le bouton de personnalisation de l’entraînement, l’utilisateur peut modifier les paramètres suivants :

- Durée totale de la séance (choisir 30 min, 45 min, 60 min...),
- Muscles visés (garder la même sélection ou ajouter/enlever certains groupes musculaires),
- Douleurs
- Intensité
- Équipements disponibles (haltères, élastiques, machines, etc.),
- Lieu (extérieur, salle, domicile).

Exemple : si l’utilisateur sélectionne “30 min” et “haut du corps”, l’application réorganise la séance (réduction du nombre d’exercices, intensité adaptée) tout en conservant les objectifs initiaux (force ou endurance) et le volume d’entraînement recommandé. La logique de modification et d’adaptation est toujours dans le backend, pas dans le front.

Détails de la séance

Après avoir cliquer sur l’encart sport sur la home page et sur sa séance du jour dans la page sport ou après l’avoir modifié, l’utilisateur peut accéder à la fiche complète de la séance :

- Liste exhaustive des exercices (avec image illustrative),
- Nombre de séries, répétitions et temps de repos pour chacun,
- Poids conseillé (ou intensité) et consignes de sécurité,

- Équipements nécessaires (ex. haltères, banc, tapis),
- Lieu recommandé (p. ex. “en salle” si présence de machines).

Chaque élément (exercice, séries, temps de repos, charges, etc.) est modifiable pour s’ajuster aux contraintes de l’utilisateur (blessures, progression, matériel manquant...). Lorsque l’utilisateur modifie ces paramètres en détail, l’application propose des conseils ou des variations (ex. remplacer un squat lourd par une fente), afin de préserver l’objectif global (durée, muscles ciblés, intensité, etc.).

Lancement et suivi en direct

Une fois la séance finalisée, l’utilisateur lance la séance :

- Vidéo explicative ou démonstrations pour chaque exercice,
- Chronomètre intégré pour gérer le temps de travail et de repos,
- Instructions textuelles (points clés de la technique, encouragements),
- Possibilité de valider l’exercice et passer au suivant ou de modifier en temps réel le nombre de répétitions, la charge utilisée, la durée du repos.

Cette validation en direct alimente la progression et permet d’ajuster automatiquement la séance en cours. Les séances futures sont également adaptées si l’utilisateur ressent des difficultés ou, au contraire, de la facilité.

Ce mode devra fonctionner en mode hors ligne (une fois la séance finale connue). Le modèle devra être capable d'ajuster le nombre de répétitions, le poids et le temps de repos et/ou d’activité selon les retours de l’utilisateur. (Fonctionnalité offline en priorité 2)

Dans un second temps (priorité 2), l’application devra se connecter à des machines de musculation connectées. La connexion entre l’application et les machines se fera par NFC ou par un QR Code que l’utilisateur scannera depuis l’application. Il devra être possible d’avoir accès à cette fonctionnalité depuis la page sport et depuis la partie accompagnement en direct de la séance. Les machines fourniront les informations suivantes :

- Type de machine et identification précise de la machine
- Nombre de répétitions de l’utilisateur
- Vitesse, temps et amplitude à chaque mouvement (montée/descente).
- Temps de repos entre chaque répétition et chaque série.
- Notes sur le niveau de difficulté de chaque série et répétition.
- Selon les machines, il sera également possible de connaître les asymétries de l’utilisateur pendant le mouvement avec le détail entre la partie gauche et droite avec les différences de vitesses, positions et amplitudes différentes.

L’accompagnement sur la séance d’entraînements devra s’adapter selon ses retours en ajustant le poids, nombre de répétitions, de séries et les temps de repos. Le programme complet devra se mettre à jour en fin de séance ensuite selon les performances durant la session du jour. Les ajustements (pas la création) se feront via des algorithmes qui suivront des logiques métiers simples.

Ajout manuel de séances (priorité 2)

Dans l'onglet sport et suivi, l'utilisateur pourra saisir manuellement des entraînements non prévus par FitBuddy (ex. une sortie en vélo ou un cours de danse). Ces données s'intègrent dans son programme hebdomadaire, permettant un suivi global de l'activité. Cette fonctionnalité est de priorité 2.

Programme Hebdomadaire et navigation

Pour chaque semaine, une vue récapitulative affiche :

- Les séances planifiées (dont l'Entraînement du Jour),
- Les séances déjà effectuées (dont celles ajoutées manuellement),
- Les séances manquées,
- Les objectifs et sous-objectifs,
- Une navigation jour par jour pour consulter le contenu détaillé ou le modifier au besoin.

Synthèse et évolution

L'ensemble de ces séances (modifiées ou validées) se synchronise avec le programme global afin de mettre à jour les prochaines séances, la progression de l'utilisateur et, le cas échéant, de rééquilibrer les objectifs (durée totale, muscles ciblés, charge de travail) en cours de cycle. Les ajustements (pas la création) se feront via des algorithmes qui suivront des logiques métiers simples.

Ainsi, même si l'utilisateur personnalise fortement une séance (lieu, durée, matériel), FitBuddy veille à maintenir la cohérence avec les objectifs sportifs initiaux (perte de poids, prise de masse, performance, etc.) et l'adapte aux performances réellement réalisées.

3.6. Mode Conversationnel

Un RAG prompting d'un coach sportif est à développer afin de créer les programmes, ajuster les séances et afin de répondre à toutes les questions sur les entraînements et la récupération. Il permettra également de répondre aux questions concernant l'application (fonctionnalités et informations disponibles). Il prendra la même forme qu'un LLM "classique" entraîné du type ChatGPT ou Gemini avec des suggestions de sujet ou question. Les réponses seront adaptées au profil de chaque utilisateur. Il sera également possible d'avoir l'historique des échanges.

3.7. Suivi des Données (priorité 2)

- **Page centrale pour le suivi globale des données**

Une page dédiée aux suivi des données viendra centraliser la totalité des données et offrir des analyses complètes sur les évolutions. Ces fonctionnalités s'inspireront d'applications comme Withings, Google Fit, Apple Health ou équivalent. Comme préciser au début du paragraphe, une page principale devra centraliser la totalité des données de l'utilisateur. La page dédiée au sport auront quelques données de suivi clés uniquement. Une section de texte pour donner des analyses et les données et évolutions seront prévues pour la page dédiée au sport et au suivi ainsi que de la documentation sur les différentes données pour former les sportifs sur ces sujets. Les données seront présentées sous forme de graphiques dans un premier temps à l'exception des muscles travaillés qui seront représentés sur un schéma de corps humain avec des pourcentage associés. Il sera possible de suivre les données représentées sur une journée, une semaine, un mois, un trimestre et une année.

- **Données de la page principale**
    - Poids
    - Composition corporelle (muscle, gras,os)
    - IMC
    - Nombre d'entraînements avec les séries d'entraînement (type flamme Duolingo)
    - Cumul de charge d'entraînement effectué vs prévu
    - Nombre de pas ou distance parcourue (à définir)
    - Score sommeil : durée, interruptions, temps d'endormissement, temps de réveil, profondeur (priorité 3)
    - Progression sur un exercice phare (répétition max et charge cumulée) avec la possibilité de modifier l'exercice phare ou d'en ajouter
    - Muscles entraînés (affichage sur un schéma de corps humain avec pourcentages affichés)
    - Streak entraînements effectués vs prévus
    - Distance ou temps cardio (uniquement pour ceux ayant un objectif lié) (priorité 3)
    - VMA/VO2 Max (priorité 3)
- Données page sport
    - Muscle travaillés durant la semaine (schéma corps humain avec pourcentage)
    - Charge et progression performance sur un exercice phare
    - Streak des entraînements

En priorité 3, il est attendu de développer une fonctionnalité permettant de faire des estimations sur les performances à venir et les résultats sur le profil de l'utilisateur selon ses habitudes afin de motiver au maximum les sportifs à suivre leur programme d'entraînement et à se surpasser.

3.8. Gamification (priorité 1 en partie)

Afin de motiver les utilisateurs au maximum, l'application intégrera des principes de gamification du type "Duolingo" ou équivalent. Le but sera de créer des mini défis quotidiens liés aux données suivis et objectifs et sous-objectifs du programme (ex : faire un entraînement). L'objectif de priorité 1 sera de donner des points à l'utilisateur lorsqu'il finalisera des exercices et finalisera une séance. L'utilisateur devra pouvoir connaître son nombre de points total et voir les récompenses qu'il pourra obtenir grâce à ces points (type Rev points Revolut). La priorité sera uniquement de connaître le nombre total de points de l'utilisateur et d'avoir un affichage simple non dynamique des récompenses possibles à obtenir. Ces points serviront à obtenir des récompenses du type goodies ou codes promotionnels.

Dans un second temps (priorité 3), des badges seront également attribués pour féliciter les utilisateurs de certaines de leurs actions. Par exemple, lorsqu'un utilisateur fera x séances, il obtiendra un badge associé. Un avatar représentant le côté gamification du coach IA est également attendu. Il devra représenter les différentes émotions de l'application en montrant de la joie et de l'engouement lorsque l'utilisateur suivra correctement son programme et de la tristesse et potentiellement de la colère lorsque l'utilisateur commencera a délaissé le programme. Le but sera d'humaniser au maximum l'application et de sensibiliser les utilisateurs afin de les garder motivés. Il devra également évoluer "physiquement" selon les évolutions de l'utilisateur.

4. Architecture Technique

4.1. Frontend & Design

- **Framework** : React Native via Expo (SDK 50+).
- **Outil de Build** : Expo Application Services (EAS) pour la compilation iOS/Android dans le cloud.
- **Compatibilité** : Web (PWA) : Support Chrome et Safari mobile via Expo Router (gestion des URLs indispensable pour le web). Mobile : iOS 13+ et Android 9+.
- **Structure du code** : Architecture modulaire (Atomic Design ou Feature-based).
- **Intégration design** : Respect fidèle des maquettes Figma, utilisation de composants vectoriels (SVG) pour l'adaptabilité.

Les vidéos à utiliser seront fournies par l'entreprise FitBuddy. Les maquettes de l'application Fit Buddy seront réalisées sur Figma par le designer de l'entreprise FitBuddy. Elles serviront de **référence unique** pour la mise en œuvre du front-end. Les développeurs front devront **suivre strictement la méthodologie et les éléments définis dans ces maquettes**, sans modifications arbitraires de styles, de couleurs, de typographies, d'espacements ou de proportions, sauf demande explicite du designer. Dans le cas où il y a des zones floues entre les maquettes et le cahier des charges, les maquettes sont la référence prioritaires.

Pour assurer une évolution fluide vers les stores (iOS/Android) sans réécriture de code, les équipes de développement devront se conformer aux exigences suivantes, en particulier l'équipe Frontend :

1. **Primitives UI :** Utilisation exclusive des composants React Native (<View>, <Text>). L'utilisation de balises HTML standards (,) est **strictement interdite**.

2. **APIs de Navigateur :** Interdiction d'utiliser des APIs spécifiques au Web (ex: window, document) qui crashent sur mobile ; utiliser les wrappers et modules d'Expo/React Native.

3. **Architecture "Server-Driven" :** La logique d'ajustement et de recommandation doit être gérée exclusivement au niveau du Backend. L'Équipe Front ne fait qu'afficher l'état dicté par l'API (pas de logique métier dans le Front).

Le designer de FitBuddy qui produira les maquettes utilisera la méthode Figma Tokens, permettant de centraliser tous les styles du design system (couleurs, typographies, espacements, ombres, radius, etc.) sous forme de variables réutilisables.
Ces tokens seront exportés en format JSON afin de garantir une correspondance directe entre le design et le code, facilitant ainsi l'intégration et la cohérence visuelle entre les écrans.

Les maquettes comprendront :

- une page "Style Guide" regroupant les couleurs, polices, icônes et composants réutilisables ;

- une page "Components" listant les éléments UI standards avec leurs différents états (normal, actif, erreur, désactivé, etc.) ;

- une page "Screens" contenant l'ensemble des écrans finaux, nommés et structurés selon la hiérarchie de navigation de l'application.

Le fichier Figma sera partagé en mode Inspect afin que les développeurs puissent accéder directement aux valeurs de design (dimensions, marges, couleurs, variables) et exporter les assets nécessaires (SVG, PNG). Toute divergence constatée entre l'interface développée et les maquettes devra être signalée et validée avant mise à jour du code.

L'objectif de cette méthodologie est d'assurer une cohérence visuelle et fonctionnelle totale entre le design et le développement, de réduire les erreurs d'intégration et de garantir la maintenabilité du code front sur l'ensemble du cycle de vie du projet.

4.2. Backend – Data et API

- **Langage** : Javascript avec typage Typescript
- **Framework** : Hono.js
- **Architecture API** : Restful API
- **Documentation :** OpenAPI / Swagger (Obligatoire avant tout développement pour que le Front et le Back se synchronisent).

4.3. Backend – Équipe récupération données capteurs et ajustement séance

- **Langage :** Python
- **Framework :** FastAPI
- **Architecture API :** Restful API (ne parle jamais au front, uniquement à la partie Data et API)
- **Documentation :** OpenAPI / Swagger (Obligatoire avant tout développement pour que le Front et le Back se synchronisent).

4.4. Base de Données

- **Technologie** : Supabase
- **Système de gestion** : PostgreSQL

4.5. Infrastructure IA pour création de programmes et conseils

- **Technologie :** Genkit.AI
- **LLM :** Gemini 3.0 mais doit être adaptable à n'importe quel autre LLM

4.6. Stockage Local et Synchronisation (priorité 2)

- **Technologie :** TinyBase ou TanStack Query (React Query) + AsyncStorage
- **Fonctionnement hors ligne** : enregistrement des données critiques (programmes, stats récentes).
- **Synchronisation :** envoi automatique des données au backend quand la connexion est rétablie.

4.7. Infrastructure d'Hébergement

- **Supabase** : héberge la base de données, l'authentification et les embeddings (pgvector)
- **Backend Data & API** : déployé sous forme de conteneur Docker sur une plateforme de conteneurs managée (ex. Railway, Render, Fly.io).

- **Service Capteurs & Logique Métier** : déployé également sous forme de conteneur Docker sur une plateforme managée (même service que l'équipe Data & API si possible).
- **Front-End** : déployé sur une plateforme web dédiée (Vercel, Netlify ou Expo Hosting), indépendamment du backend.
- **API Capteurs Externe (hors périmètre projet)** : fournie par une VM Azure existante, consommée uniquement par le service Logique Métiers.
- **Développement local** : utilisation de **Docker Compose uniquement en local** pour lancer plusieurs services en même temps, chaque service restant déployé indépendamment en production.

4.8. Modules logiques

- **Front & IA :**
  - Module Interface (UI/UX)
    - Implémentation des screens selon le design Figma (pixel perfect, tokens de couleurs/typos).
    - Gestion de la navigation (Expo Router).
    - Affichage des médias (images / vidéos).
  - Module Player de Séance
    - Chronomètre de séance et de séries.
    - Validation des séries
    - Affichage du déroulé de la séance (exercices à venir, progression).
  - Module Interface Chatbot (stateless)
    - Fenêtre de chat, bulles, champ texte, états de chargement.
    - Affiche les réponses envoyées par le backend (mode conversationnel).
    - Ne gère pas l'historique ni la logique métier du chat (géré côté backend).
  - Module Gestion d'État (client)
    - Utilisation de React Query/Context pour stocker et afficher les données reçues (profil, programme, séances, stats simples).
    - Pas de logique métier lourde : le front se contente d'afficher l'état dicté par l'API ("server-driven").

- Module Communication API (client HTTP)
    - Centralisation des appels réseaux vers l'API Gateway (Data & API).
    - Gestion des erreurs d'appel (messages utilisateur, re-try simple si nécessaire).
    - Injection des tokens d'auth récupérés via le flow Auth (JWT, etc.).
    - Important : ce module n'implémente pas la logique d'appel LLM, il consomme des endpoints backend déjà prêts.

- **API & Data :**
    - API Gateway & Routing
        - Point d'entrée unique pour la web app.
        - Expose les endpoints publics (Auth, Profil, Programme, Séances, Chat, Stats, etc.).
        - Route les appels vers :
            - la base de données,
            - les services IA (RAG + LLM),
            - les services de l'équipe Capteurs & Logique métier.
    - Module Authentification & Gestion des Utilisateurs
        - Inscription / connexion (Email + mot de passe en V1).
        - Gestion des tokens (JWT ou session) et des refresh tokens si nécessaire.
        - Protection des routes (middlewares d'authentification, vérification des droits).
    - Module Base de Données & Modélisation (PostgreSQL)
        - Conception et maintenance du schéma relationnel et de la base de données
        - Écriture des requêtes (ORM/SQL), transactions critiques, indices, intégrité référentielle.
    - Pipeline RAG (Ingestion & Recherche)
        - Ingestion des fichiers Markdown fournis par l'équipe Front & IA (doc RAG).
        - Vectorisation des documents et stockage (Postgres + pgvector ou autre).
        - Service interne pour rechercher les passages pertinents en fonction d'une question utilisateur ou d'un besoin contexte (profil, programme, historique).

- Fournit des données prêtes à être injectées dans le prompt (titre, extrait, type de ressource).

- Module LLM (Client IA)
  - Gestion des appels vers le fournisseur LLM (Gemini, OpenAI, etc.).
  - Gestion des clés API, des quotas et des éventuelles limites de taux.
  - Gestion des erreurs réseau / timeouts / retries.
- Pipeline de Création de Programme IA
  - Construction du megaprompt pour la création de programme en combinant :
    - les données utilisateur (profil, objectifs, contraintes),
    - le contexte d'entraînement (historique, matériel disponible),
    - les extraits RAG pertinents,
    - les "system prompts" définis par l'équipe Front.
  - Appel du Service LLM.
  - Validation stricte du JSON de sortie (Zod/Typebox), gestion des cas de sortie invalide (retry, prompt de correction, etc.).
  - Persistance du programme validé en base de données.
- Pipeline Mode Conversationnel
  - Gestion du mode chat :
    - construction du prompt conversationnel (historique + profil + contexte + RAG + system prompt)
    - gestion historique conversationnel (10-20 derniers messages en DB)
    - appels au Service LLM.
  - Validation / filtrage des réponses (sécurité, cohérence, format).
  - Renvoi d'une réponse propre au front (texte, éventuellement suggestions d'actions).
- Module de Validation & Stockage des Programmes et Séances
  - Garantir que toutes les modifications générées par l'IA ou par la logique métier (capteurs) respectent les contraintes de schéma.

- Centraliser la persistance des programmes/séances (création, mise à jour, archivage).
- Jouer les migrations éventuelles ou transformations nécessaires.

- Module d'Intégration Capteurs (Interop backend)
  - Client HTTP/gRPC vers le Service d'Interopérabilité de l'équipe Capteurs & Logique métier.
  - Récupère :
    - les données agrégées des capteurs,
    - les estimations de temps d'attente,
    - les propositions d'ajustement de séance/programme.
  - Transforme ces données au format interne de la BDD FitBuddy et les enregistre via le Service Base de Données.

- **Capteurs et Ajustement séances et programmes :**
  - Module d'Ingestion des Données Capteurs
    - Va chercher les données capteurs traitées via une API externe. Pour le premier prototype, des données simulées seront utilisées. Il faudra utiliser un script python qui génère régulièrement des fausses données en fonction de l'heure de la journée et/ou jour de la semaine.
  - Module de Validation & Stockage Capteurs
    - Vérification et validation des données
    - Association aux bonnes entités : utilisateur, séance, exercice, machine.
    - Stockage dans une base adaptée (Postgres ou time-series DB).
  - Module d'Estimation du Temps d'Attente Machine
    - Implémentation du modèle d'estimation du temps d'attente par machine (règles heuristiques ou modèle ML simple dans un premier temps).
    - Exposition d'un endpoint interne
  - Module d'Ajustement Dynamique de Séance (Temps Réel)
    - Logique métier pour ajuster la séance en cours :

        - changement d’ordre des exercices,
        - proposition d’exercices alternatifs,
        - adaptation volume/intensité si la séance est trop longue.
    - Fournit des recommandations sous forme de JSON (diff/patch de séance).
- Module d’Ajustement de séance et programme (Hors séance)
    - Logique métier “en dur” (sans LLM) pour adapter les séances et programmes
- Module d’Interopérabilité (API interne)
    - Expose une API propre, documentée (OpenAPI/Swagger) pour l’équipe Data & API :
        - récupération des stats capteurs,
        - récupération des estimations d’attente,
        - propositions d’ajustement.
    - Cette API est consommée uniquement par l’API Gateway, jamais directement par le front.

5. Sécurité et Conformité RGPD

5.1. Collecte, Traitement et Stockage des Données Personnelles (priorité 1)

- **Données utilisateurs :** chiffrées, anonymisées et stockées de manière sécurisée.
- **Consentement explicite** : demander une acceptation claire des conditions d’utilisation de l'utilisateur avec une explication sur l’usage de ses données dans le détail des conditions.

5.2. Droits des Utilisateurs (Priorité 2)

Dans les paramètres de l’application, il devra être possible de gérer les demandes RGPD (accès, rectification et suppression des données).

5.3. Politique de Confidentialité (Priorité 2)

- **Documentation** : détails sur la finalité des collectes, durée de conservation, partage éventuel à des tiers.

5.4. Audit et Traçabilité (Priorité 2)

- **Log** des accès et manipulations de données sensibles.

6. Tests prototype

6.1. Création de compte / Authentification

- Vérifier la bonne inscription via e-mail & mot de passe
- À tester :
  - Création d'un compte classique (email + mot de passe).
  - Réinitialisation du mot de passe.

6.2. Questionnaire (Obligatoires + Optionnels)

- S'assurer que les 10 questions obligatoires s'enchaînent sans bug et que les 10-15 questions optionnelles s'adaptent bien aux réponses (prise de masse, perte de poids, etc.).
- À tester :
  - Remplir intégralement le questionnaire.
  - Sauter des questions pour voir si la relance (pop-up) fonctionne.
  - Vérifier l'adaptation automatique (si on répond "Objectif endurance", on ne voit pas les mêmes questions que "Objectif force").

6.3. Création Automatique du Programme Sportif

- Vérifier que les programmes générés (planning, séances, objectifs, progression) sont cohérents avec les réponses au questionnaire.
- À tester :
  - Plusieurs scénarios (objectif "Perte de poids", "Prise de masse").
  - Cas où le questionnaire est partiellement rempli (vérifier que le programme reste cohérent).

6.4. Accompagnement sur Séance de Sport en Direct ("Entraînement du Jour")

- C'est la fonctionnalité clé pour démarrer une séance, la modifier, valider l'exercice en temps réel.
- À tester :
  - Personnaliser la séance (durée, intensité, matériel).
  - Lancer la séance, modifier en direct (changer le nombre de répétitions, la charge, le temps de repos).

6.5. Chat conversationnel

- Vérifier que le chat répond de façon cohérente et utile aux questions sur l'entraînement/ la récupération/l'app
- Prend en compte le profil utilisateur + son programme
- Utilise bien la base RAG
- Ne modifie pas directement le programme en P1

7. Livrables

7.1. Livrables par Sprint

- README principal
  - Procédure d'installation et de lancement du projet.
  - Informations sur la configuration (variables d'environnement, dépendances).
  - Brève description de l'arborescence des dossiers.
- Code sur branche stable
  - Dernière version compilable et exécutable.
  - Conventions Git (branches, messages de commit) respectées.
- Qualité de code
  - Fichiers de configuration pour ESLint, Prettier et TypeScript.
  - Scripts npm run lint et npm run build fonctionnels.
- Tests unitaires
  - Au moins un test par fonctionnalité critique.
  - Script npm run test exécutable.
  - Parcourir une séance type (ex. "Séance Force Bas du Corps"), vérifier la synthèse (durée, muscles ciblés).
- Plan d'API
  - Liste des routes (méthodes, URL, params, réponses).
  - Schéma simplifié des tables principales (DB Supabase).

7.2. Livrable Final du Prototype

- Tag de version
  - Code final marqué (ex. v1.0-prototype).
- Release Notes
  - Fonctionnalités incluses, limitations et éventuels bugs connus.
- Documentation d'Architecture
  - Diagramme global (front React Native, backend Hono.js, BDD Supabase, etc.).
  - Instructions de déploiement (en local ou en production).
- Tests et Couverture
  - Niveau minimal de couverture maintenu, exécutable via npm run test.