



Projet de Fin d'Année – FIE3 Athletica

Suivi personnalisé des performances en course à pied

Réalisé par :

Ethan Cabanes – Chef de projet
Yasmine Belarbi – Responsable de la communication
Younes Nafaa – Contributeur

Encadrant pédagogique : Adrien Defossez
Client : Yohann Chasseray

Année universitaire 2024–2025

Table des matières

1 Analyse et étude d'opportunité	2
1.1 Introduction du sujet	2
1.2 Synthèse de l'analyse du besoin, de l'opportunité du projet, et de sa faisabilité	3
1.3 Résultats de l'analyse des risques	6
1.4 Plan de gestion des risques	8
1.5 Roadmap	9
2 Conception	11
2.1 Spécifications modélisées et commentées	11
2.2 Fonctionnalités	12
2.3 Modèle statique de données (diagramme de classes ou modèle de données)	14
2.4 Architecture logicielle	16
2.5 Éléments pertinents de la phase de génération de la solution .	17
3 Développement de l'application	23
3.1 Technologies et outils utilisés	23
3.2 Organisation et gestion de l'intégration et du versioning	25
4 Suivi de projet	27
4.1 Organisation et gestion ScrumBan	27
4.2 Répartition des rôles	27
4.3 Méthodes agiles et itérations	28
4.4 Suivi de la qualité	28
4.5 Recettage et déploiement	28
5 Conclusion	30

Chapitre 1

Analyse et étude d'opportunité

1.1 Introduction du sujet

Le projet **Athletica** consiste en la création d'une application web dédiée au suivi personnalisé des performances sportives, principalement orientée vers la course à pied. Ce projet a été mené dans le cadre de notre formation en ingénierie informatique pour la santé, en collaboration avec un client réel et sous l'encadrement d'un tuteur pédagogique.

L'objectif principal de cette application est d'offrir aux utilisateurs la possibilité d'enregistrer et de visualiser leurs sessions d'entraînement, tout en leur permettant d'ajouter des indicateurs contextuels personnalisés, tels que la météo, la fatigue, les douleurs musculaires, en complément des données classiques telles que la distance, le temps ou la vitesse. Cette personnalisation des indicateurs est l'élément différentiateur de notre application par rapport aux solutions existantes sur le marché.

1.2 Synthèse de l'analyse du besoin, de l'opportunité du projet, et de sa faisabilité

Analyse du besoin

Dès les premières semaines du projet, nous avons lancé une phase d'idéation pour mieux comprendre les besoins des utilisateurs finaux. Cette phase a permis de définir des profils utilisateurs. Deux principaux profils ont été retenus :

- **Les coureurs amateurs** : Ceux-ci cherchent un outil simple et adapté à leurs critères de performance personnels, sans complexité inutile.
- **Les coachs sportifs** : Ces utilisateurs souhaitent une plateforme qui leur permette de suivre les performances de plusieurs athlètes tout en personnalisant les indicateurs en fonction des objectifs individuels de chaque coureur.

Pour approfondir cette analyse, un questionnaire a été diffusé auprès d'une vingtaine de coureurs. Les résultats ont mis en évidence plusieurs besoins récurrents :

- Le suivi d'indicateurs contextuels comme la qualité du sommeil ou la douleur musculaire.
- La difficulté à centraliser les données provenant de différentes applications.
- Le manque de visibilité sur les tendances de progression des performances au fil du temps.

Parallèlement, des maquettes ont été produites pour donner une représentation visuelle des interfaces. Initialement, nous avons utilisé **MockFlow**, une solution intuitive pour concevoir des premiers prototypes. Cependant, son manque d'interaction et de personnalisation nous a poussés à migrer vers **Figma**, qui, bien que plus complexe à prendre en main, offrait des possibilités de conception plus larges, notamment pour l'interactivité essentielle lors des premières présentations client. Ces prototypes ont été validés et affinés au cours de focus groups avec des étudiants sportifs et des coachs, permettant de simplifier certaines vues et de repenser le parcours utilisateur. **Les maquettes finalisées seront présentées en détail dans la phase de conception afin d'illustrer les choix ergonomiques et fonctionnels retenus.**

Une difficulté majeure lors de cette phase fut la traduction des attentes qualitatives des utilisateurs en exigences fonctionnelles précises. Il a fallu trouver un compromis entre la richesse fonctionnelle souhaitée et la nécessité de maintenir l'application lisible et performante.

Analyse de l'opportunité

Le marché des applications de running est bien établi, avec des acteurs majeurs comme **Strava**, **Nike Run Club** et **Runkeeper**, qui proposent principalement des indicateurs standards avec une forte composante de suivi automatique via GPS.

Cependant, ces outils manquent de personnalisation en termes d'indicateurs contextuels, ce qui crée une véritable opportunité pour notre application. En intégrant la possibilité d'ajouter des données telles que la météo, la fatigue ou les douleurs musculaires, **Athletica** se positionne comme une alternative plus flexible et centrée sur l'utilisateur.

Critères	Strava	Nike Run Club	Runkeeper	Notre solution
Orientation principale	Tracking des performances sportives	Coaching personnalisé	Plans d'entraînement	Personnalisation et suivi global (météo, fatigue, etc.)
Mode de saisie	Automatique via GPS	Automatique via GPS	Automatique ou manuel	Manuel, GPS et indicateurs personnalisés
Gestion des indicateurs	Standard : vitesse, distance	Standard + coaching	Standard : vitesse, distance	Indicateurs personnalisés (météo, douleurs, fatigue)
Tarification	Gratuit, premium payant	Gratuit	Gratuit	Gratuit

FIGURE 1.1 – Comparaison des applications de running

Avantage concurrentiel

Le principal avantage concurrentiel de notre projet réside dans la liberté offerte à l'utilisateur pour personnaliser les critères de suivi, notamment en ce qui concerne les données contextuelles comme la météo, les douleurs musculaires ou le niveau de fatigue.

Cette personnalisation permet à chaque utilisateur de définir ses propres objectifs de performance, en fonction de ses besoins spécifiques, ce qui est très peu proposé par les applications concurrentes actuelles.

Une **analyse SWOT** a également été menée pour identifier les points forts, les faiblesses, les opportunités et les menaces associées au projet.

Une analyse SWOT a permis d'identifier les points forts du projet :

Catégorie	Soutien	Frein
Interne (équipe projet)	Strengths (Forces) : <ul style="list-style-type: none">- Bonne entente et collaboration au sein du groupe.- Compétences en gestion de projet grâce aux expériences précédentes.	Weaknesses (Faiblesses) : <ul style="list-style-type: none">- Compétences techniques en développement encore limitées.- Cours condensés limitant le temps disponible pour avancer efficacement sur le projet.
Externe (environnement du client)	Opportunities (Opportunités) : <ul style="list-style-type: none">- Intérêt croissant pour les applications de suivi sportif.- Possibilité de se différencier en proposant des indicateurs personnalisés et innovants.- Potentiel de partenariats avec des clubs de sport locaux ou des coachs.	Threats (Menaces) : <ul style="list-style-type: none">- Concurrence des applications déjà établies sur le marché (ex. Strava, Garmin, etc.).- Adoption incertaine par des utilisateurs fidèles à d'autres outils.- Risques liés à la collecte et la sécurité des données personnelles.

FIGURE 1.2 – Analyse SWOT

Faisabilité du projet

Techniquement, le projet est réalisable en utilisant des technologies modernes. Le choix de **H2** comme base de données permet une gestion fiable des données en environnement local, facilitant le développement et les tests des fonctionnalités liées aux sessions et aux indicateurs.

De plus, des frameworks web comme **Vue.js** (pour le frontend) et **Spring Boot** (pour le backend) sont parfaitement adaptés pour la création de l'application, offrant à la fois réactivité, évolutivité et maintenabilité.

L'évaluation financière du projet, bien que montrant un coût modéré (environ **7 870 à 8 130 €**), reste maîtrisable grâce à une gestion optimisée des ressources humaines et techniques.

1.3 Résultats de l'analyse des risques

L'analyse des risques a permis d'identifier plusieurs défis potentiels qui pourraient affecter le bon déroulement du projet. Ces risques ont été classés en différentes catégories pour faciliter leur gestion et leur suivi.

1. Risques techniques

- **Problèmes d'intégration avec la base de données (H2)** : L'un des risques identifiés concerne les limitations potentielles de **H2** en termes de performance et de gestion de volumes importants de données (sessions de course, indicateurs personnalisés). Bien que H2 soit adapté au développement, des tests approfondis seront réalisés pour s'assurer de la fiabilité de l'intégration et anticiper une éventuelle migration vers un SGBD plus robuste pour la production.
- **Problèmes liés à l'importation des fichiers CSV** : L'importation des fichiers CSV provenant d'autres applications de suivi (comme Strava ou Nike Run Club) peut poser des problèmes, notamment en termes de formatage et d'erreurs de données. Des mécanismes de validation des fichiers seront mis en place pour garantir une importation fluide et sans erreurs.
- **Sécurité des données des utilisateurs** : Avec l'augmentation de la collecte de données personnelles, notamment les performances sportives et les informations de santé, la sécurité des données devient une priorité. Des protocoles de sécurité stricts, comme le chiffrement et l'authentification renforcée, seront mis en œuvre pour protéger ces informations sensibles.

2. Risques liés à l'utilisateur

- **Adoption incertaine** : L'un des risques notables est le manque d'adhésion des utilisateurs, notamment si l'application ne se différencie pas suffisamment des applications concurrentes comme **Strava** ou **Nike Run Club**. Il sera important de promouvoir les fonctionnalités uniques de personnalisation des indicateurs contextuels afin d'attirer les utilisateurs potentiels.
- **Difficulté de prise en main de l'application** : Bien que l'objectif soit de créer une application intuitive, certains utilisateurs pourraient trouver la personnalisation des indicateurs complexe. Un travail sur l'ergonomie et des tutoriels interactifs sera essentiel pour faciliter l'adoption.

3. Risques liés au planning et à la gestion du temps

- **Retards dans le développement** : Comme pour tout projet informatique, des retards peuvent survenir, notamment en raison de problèmes techniques imprévus ou de l'ampleur du travail de développement. Un suivi rigoureux du planning de développement sera effectué, avec des ajustements réguliers pour respecter les délais.
- **Manque de temps pour les tests finaux** : Le temps alloué aux tests utilisateurs et aux ajustements finaux pourrait être insuffisant. Il est donc crucial d'intégrer des phases de tests dès le début du développement pour éviter des retards dans cette phase critique.

4. Risques financiers

- **Surcoût en ressources humaines** : Le projet étant mené avec une équipe réduite, il existe un risque de dépassement des coûts estimés en cas de besoin accru en ressources humaines, notamment pour la phase de tests et le suivi post-livraison. Une gestion stricte des ressources et une planification détaillée permettront de minimiser ce risque.

1.4 Plan de gestion des risques

Dans le cadre du projet **Athletica**, un plan de gestion des risques a été défini afin d'anticiper les difficultés pouvant nuire à la réussite du projet. Ce plan repose sur une analyse croisée de la probabilité d'occurrence et de l'impact potentiel de chaque risque. Deux niveaux de priorité ont été établis : les **risques prioritaires**, qui nécessitent une surveillance constante et des actions préventives claires, et les **risques secondaires**, qui sont suivis de manière plus souple mais restent sous contrôle.

Parmi les risques prioritaires, l'intégration de la base de données **H2** constitue une préoccupation centrale. Bien que ce SGBD soit parfaitement adapté au développement, il peut présenter des limites en termes de gestion de volumes importants de données, en particulier lorsqu'il s'agit de sessions sportives enrichies d'indicateurs personnalisés. Pour prévenir ce risque, des jeux de données massifs ont été utilisés en test afin d'identifier d'éventuelles faiblesses. Le projet prévoit également une stratégie de migration vers **PostgreSQL** si des performances insuffisantes sont observées. Cette anticipation garantit une flexibilité technique pour un futur passage en production.

Un second risque critique concerne l'**adoption de l'application par les utilisateurs**. Le projet se positionne sur un marché concurrentiel dominé par des solutions bien établies comme **Strava** ou **Nike Run Club**. Pour que les utilisateurs perçoivent l'intérêt de notre application, il est essentiel de valoriser les fonctionnalités différenciantes, notamment la personnalisation des indicateurs contextuels. Des actions de communication ciblées et la mise en place de sessions de test avec des utilisateurs finaux ont été prévues pour renforcer l'adhésion. En cas de retours négatifs, des ajustements fonctionnels pourront être mis en œuvre rapidement grâce à une architecture modulable.

Le **respect du planning** est également un enjeu de premier ordre. La diversité des tâches, couplée à une équipe réduite, expose le projet à des risques de retard. Pour limiter cet impact, une organisation rigoureuse en **sprints** et un suivi via un outil *ScrumBan* ont été instaurés dès le début. Chaque tâche est suivie, priorisée et réévaluée régulièrement en fonction de sa criticité. Cette organisation permet de détecter rapidement les dérives et de recentrer les efforts sur les livrables les plus essentiels.

Parmi les risques secondaires, la **sécurité des données** occupe une place importante. Les données manipulées, bien que sportives, peuvent contenir des éléments personnels liés à la santé. Le projet intègre donc des protocoles de sécurisation dès la phase de développement : chiffrement, authentification sécurisée, audit régulier du code, et respect des bonnes pratiques RGPD.

Enfin, la **complexité d'utilisation** perçue par certains utilisateurs pourrait freiner la prise en main de l'application. Ce risque est pris en compte par la conception d'une interface simple et intuitive, associée à un système de tutoriels interactifs. L'objectif est d'offrir une expérience utilisateur fluide et accessible, même pour les moins technophiles.

Le suivi de ces risques est intégré dans le cycle de développement agile. À chaque fin de sprint, une revue des risques est effectuée avec l'équipe projet, permettant d'actualiser les priorités et d'adapter les stratégies mises en place. Cette démarche garantit une gestion dynamique et proactive des risques tout au long du projet.

1.5 Roadmap

La **roadmap** du projet se divise en plusieurs phases clés, correspondant aux étapes principales du cycle de vie de l'application.

Phase de conception

Cette phase inclura :

- L'étude des besoins des utilisateurs.
- La définition des fonctionnalités principales.
- La modélisation des données.
- La conception des maquettes de l'application.

Phase de développement

Le développement se concentrera sur :

- L'implémentation des fonctionnalités principales, telles que la création d'indicateurs personnalisés, le suivi des données et la gestion des sessions de course.
- La réalisation de tests réguliers pour garantir la qualité du code et les performances de l'application.

Phase de déploiement

Une fois l'application développée :

- Des tests utilisateurs seront menés pour recueillir des retours et affiner l'expérience.
- L'application sera déployée auprès des utilisateurs cibles.
- Une phase de maintenance sera mise en place pour assurer son bon fonctionnement sur le long terme.

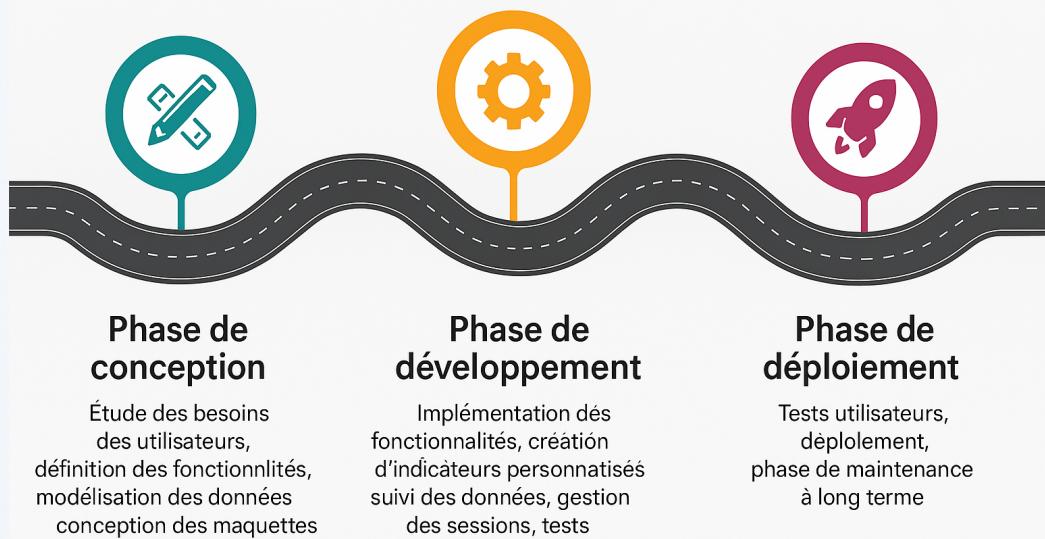


FIGURE 1.3 – Roadmap

Chapitre 2

Conception

Dans cette section, nous allons détailler la conception de l'application, en abordant les différents éléments essentiels à sa réalisation, tels que les spécifications, les fonctionnalités, les modèles de données, l'architecture logicielle, ainsi que les décisions prises pendant la phase de génération de la solution.

2.1 Spécifications modélisées et commentées

Exigences de la solution

L'application a pour but de permettre aux utilisateurs de suivre leurs performances sportives, principalement liées à la course à pied. Voici les exigences principales de la solution :

- **Enregistrement des sessions de course** : Les utilisateurs peuvent enregistrer des sessions de course, en entrant des informations telles que le temps, la distance parcourue et les indicateurs liés à la course.
- **Personnalisation des indicateurs** : Les utilisateurs peuvent créer des indicateurs personnalisés, en plus des indicateurs globaux (tels que la météo, la fatigue, etc.), pour suivre leur performance de manière plus spécifique.
- **Ajout de mesures** : Les utilisateurs peuvent ajouter des mesures associées à chaque indicateur (par exemple, température, douleur musculaire) pendant ou après la session.
- **Visualisation des progrès** : Les utilisateurs peuvent consulter graphiquement leurs progrès sur les différents indicateurs, afin de suivre leur évolution au fil du temps.

2.2 Fonctionnalités

Les fonctionnalités de l'application sont les suivantes :

1. **Enregistrement de sessions** : L'utilisateur peut enregistrer des informations relatives à une session de course.
2. **Ajout d'indicateurs globaux et de session** : L'utilisateur peut ajouter des indicateurs globaux et spécifiques à chaque session.
3. **Ajout de mesures** : Une fois les indicateurs ajoutés, l'utilisateur peut y associer des mesures.
4. **Affichage des progrès** : L'utilisateur peut visualiser les données sous forme de graphiques pour suivre l'évolution de ses performances.

Diagramme de cas d'utilisation (Use Cases)

Voici un diagramme de Use Case qui illustre les interactions entre l'utilisateur et le système.

- **Utilisateurs** : L'utilisateur enregistre une session, ajoute des indicateurs, y associe des mesures, et consulte les graphiques de ses progrès.
- **Système** : Le système enregistre les données, calcule les progrès, et affiche les résultats sous forme graphique.

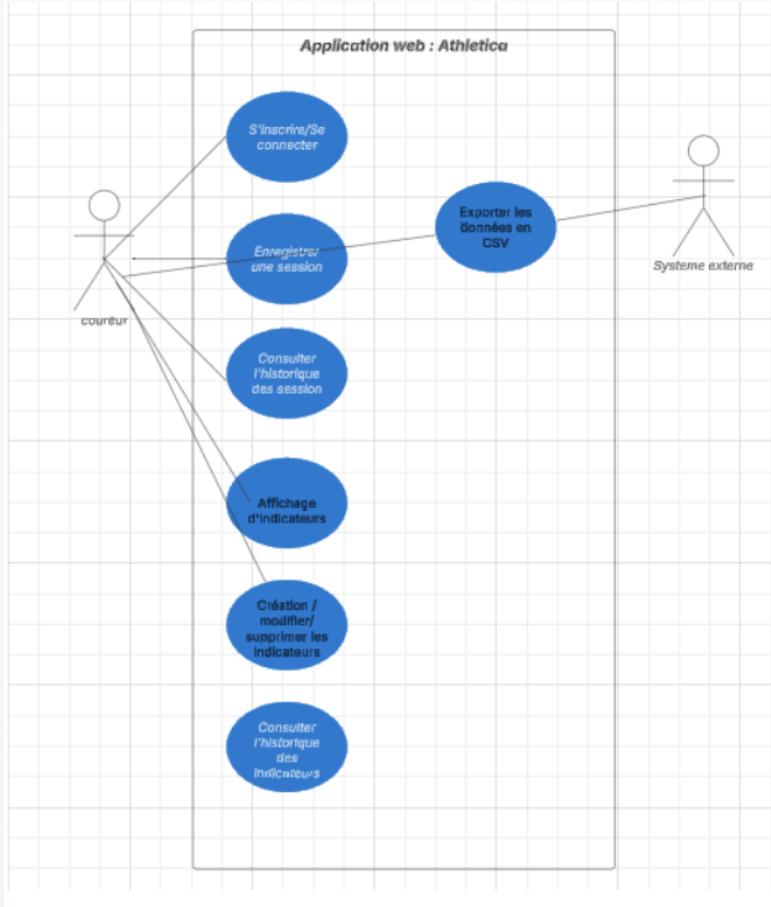


FIGURE 2.1 – Diagramme de cas d'utilisation de l'application Athletica

2.3 Modèle statique de données (diagramme de classes ou modèle de données)

Le modèle statique des données représente les entités et leurs relations dans la base de données. Il permet de visualiser les liens entre les différentes entités telles que **Session**, **Indicateur Global**, **Indicateur Session**, **Mesure**, et **Utilisateur**.

Premier MCD (avant révision)

Ce diagramme montre la première version du Modèle Conceptuel de Données (MCD), dans laquelle certaines relations étaient mal définies, ce qui pouvait entraîner des problèmes de cohérence comme des boucles infinies.

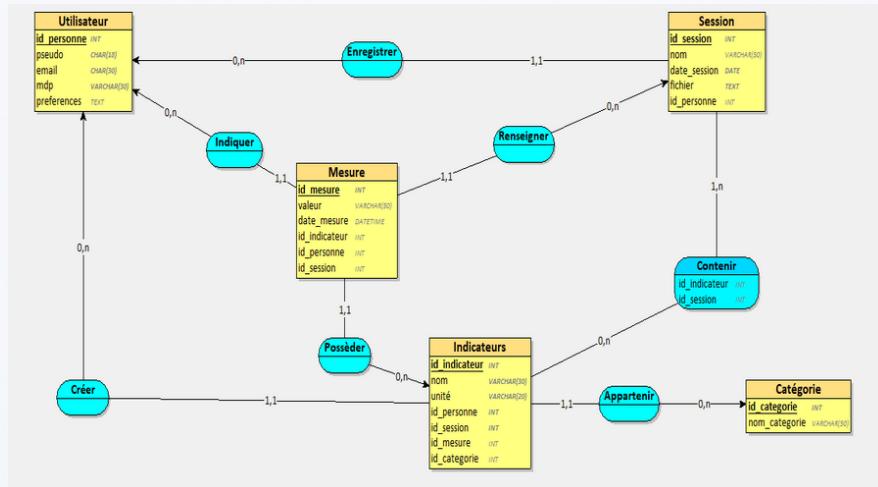


FIGURE 2.2 – Premier MCD – version initiale avec relations problématiques

MCD corrigé (version finale)

Après analyse, ce diagramme a été corrigé pour mieux représenter la structure relationnelle, en séparant clairement les entités et en évitant les dépendances circulaires problématiques.

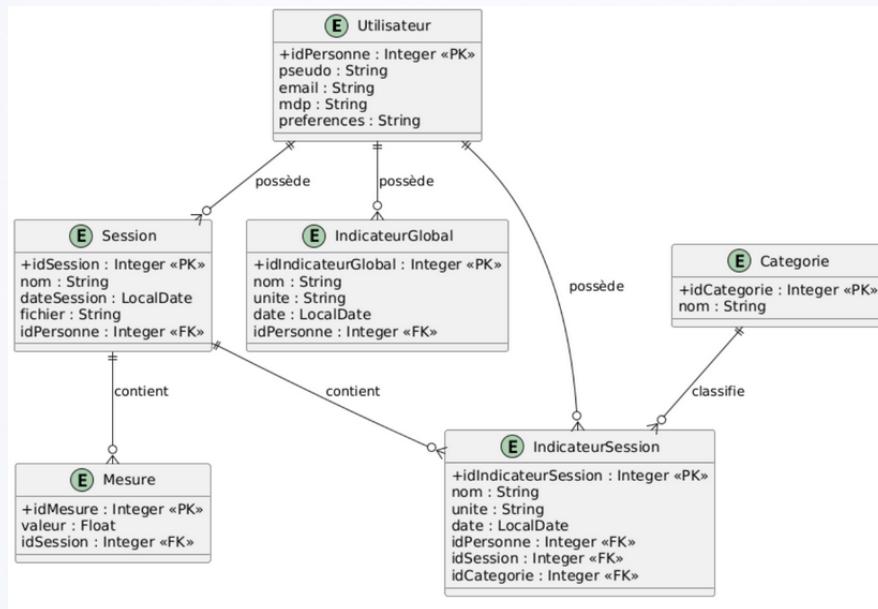


FIGURE 2.3 – MCD corrigé – version finale retenue pour le développement

2.4 Architecture logicielle

L'architecture de l'application repose sur un modèle **client-serveur**, structuré autour des technologies suivantes :

- **Frontend** : *Vue.js*, un framework JavaScript moderne qui permet de créer des interfaces utilisateur réactives et dynamiques.
- **Backend** : *Spring Boot*, un framework Java adapté pour créer des applications web robustes, sécurisées et évolutives.
- **Base de données** : *H2*, un SGBD relationnel léger et embarqué, idéal pour le développement et les tests. Il permet une gestion fiable des données en local (sessions, indicateurs, mesures), tout en facilitant une transition future vers une base plus robuste comme *PostgreSQL*.

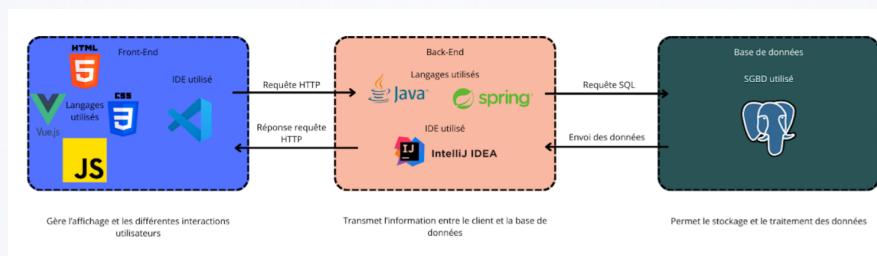


FIGURE 2.4 – Architecture logicielle

2.5 Éléments pertinents de la phase de génération de la solution

Plusieurs décisions importantes ont été prises lors de la phase de génération de la solution afin de garantir une application robuste, pertinente et centrée sur l'utilisateur.

Personas

Pour s'assurer que le produit réponde aux besoins des utilisateurs finaux, nous avons créé un **persona** représentant un utilisateur type. Ce persona détaille ses caractéristiques, ses objectifs, ses frustrations et ses attentes. Il a servi de référence tout au long du processus de conception, notamment pour guider les choix fonctionnels, les maquettes et l'ergonomie de l'interface.

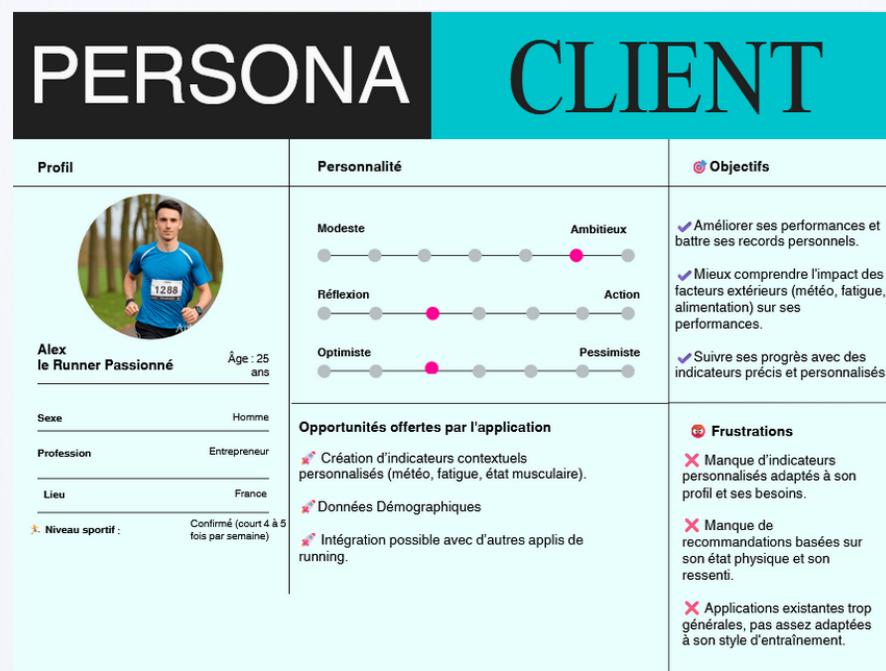


FIGURE 2.5 – Persona

Expérience Map

L'Expérience Map a été utilisée pour visualiser l'ensemble du parcours utilisateur, en identifiant les émotions, besoins et actions à chaque étape de l'utilisation de l'application.

Cette démarche a permis d'affiner l'interface utilisateur et de garantir que chaque fonctionnalité réponde aux besoins exprimés par notre persona principal, *Alex*, ainsi que par d'autres utilisateurs ayant un profil similaire.

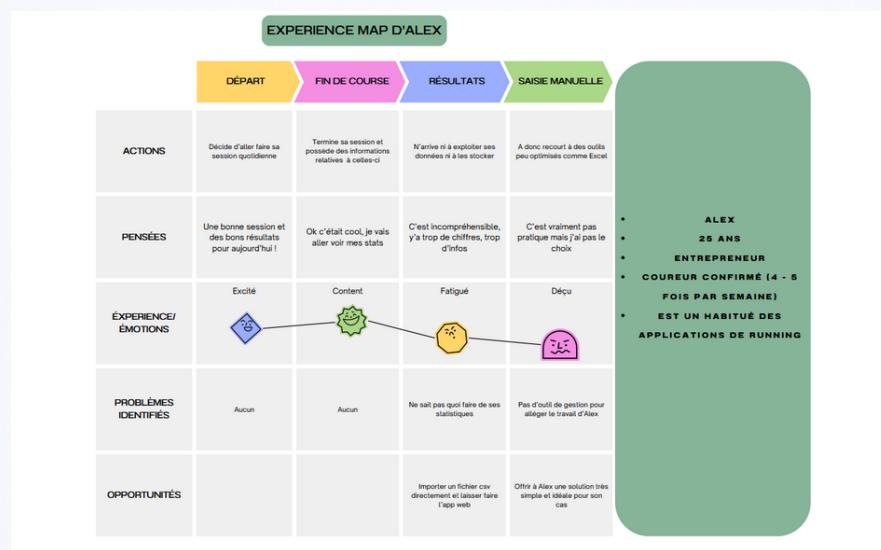


FIGURE 2.6 – Expérience Map

Story Mapping

Le **Story Mapping** a été utilisé pour organiser les fonctionnalités du projet par priorité. Ce processus a permis de structurer l'ensemble des *user stories* en fonction de leur importance pour l'utilisateur final et de leur apport au produit.

Cette méthode a été essentielle pour planifier les itérations de développement, en veillant à ce que les fonctionnalités principales soient développées en priorité. Elle a également facilité le suivi des tâches et la coordination entre les membres de l'équipe.

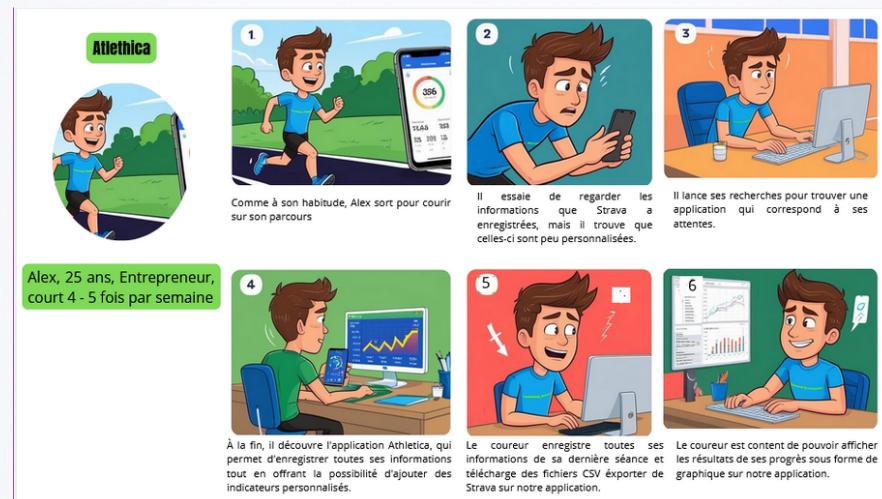


FIGURE 2.7 – Story Mapping

Maquettes

Des maquettes ont été réalisées avec **Figma** afin d'illustrer l'interface utilisateur et la navigation au sein de l'application. Ces maquettes ont servi de support tout au long du développement pour aligner les choix techniques avec les attentes des utilisateurs.

Parmi les écrans conçus, on retrouve notamment :

- Une **maquette d'accueil** avec l'option d'enregistrement d'une nouvelle session.

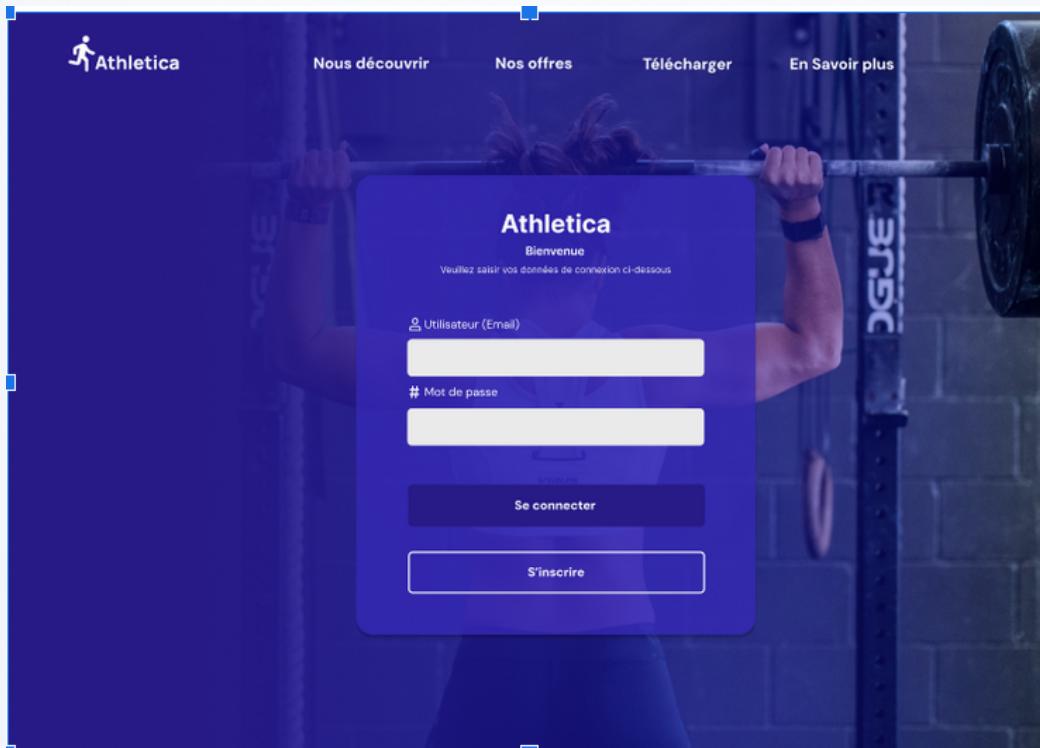


FIGURE 2.8 – Maquette de la page d'accueil

- Des vues permettant d'ajouter ou consulter des indicateurs.

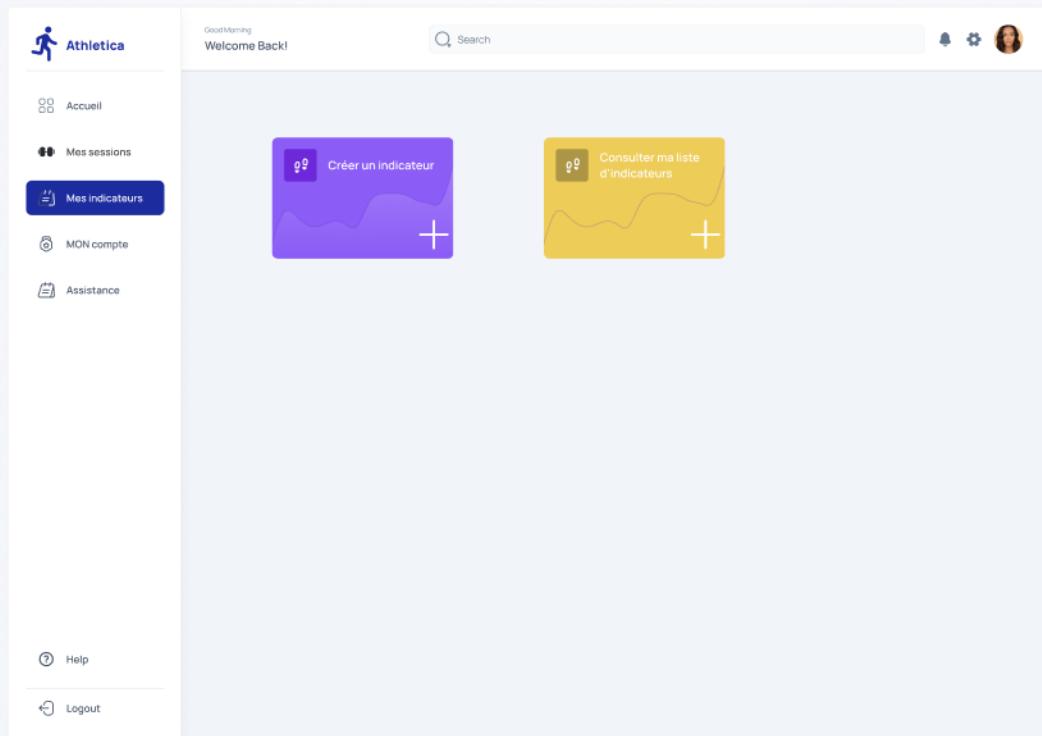


FIGURE 2.9 – Maquette de gestion des indicateurs

- Un écran de visualisation des performances sous forme de progression graphique.

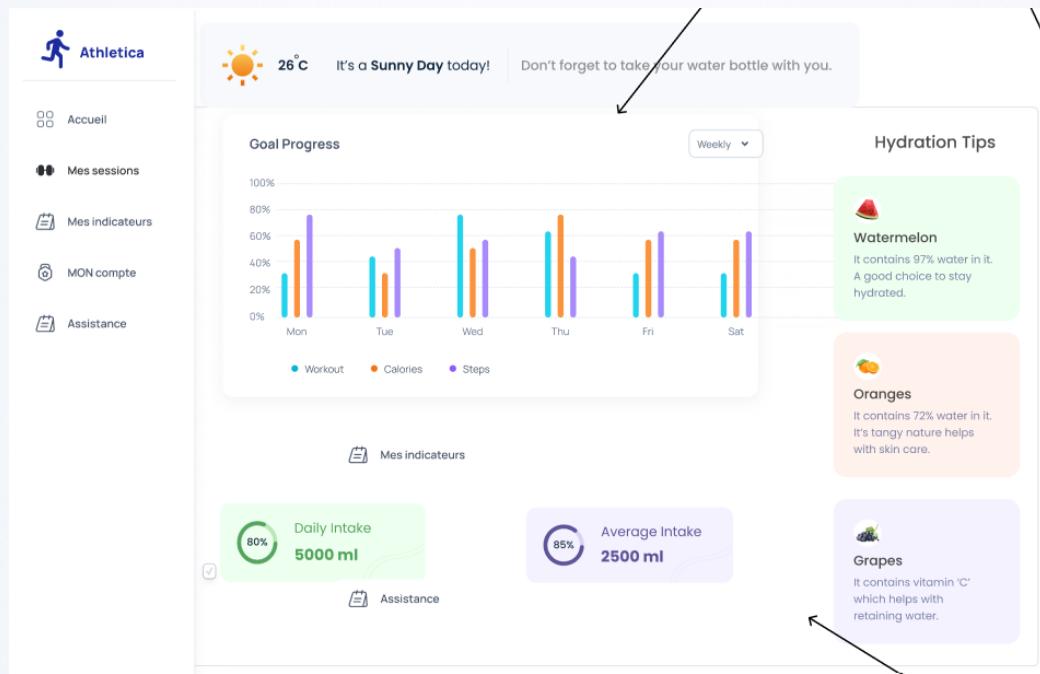


FIGURE 2.10 – Maquette de visualisation des progrès

Chapitre 3

Développement de l'application

3.1 Technologies et outils utilisés

Pour le développement de l'application, nous avons sélectionné des technologies éprouvées et adaptées aux besoins spécifiques du projet. Voici un aperçu des technologies et outils utilisés.

Frontend : Vue.js

Le frontend de l'application a été développé avec **Vue.js**, un framework JavaScript moderne permettant de créer des interfaces utilisateurs réactives et dynamiques. Vue.js a été choisi pour sa flexibilité, sa courbe d'apprentissage relativement courte, et la possibilité de créer des composants réutilisables, ce qui facilite la gestion et l'extension de l'interface utilisateur.

Backend : Spring Boot

Le backend a été réalisé avec **Spring Boot**, un framework Java permettant de développer des applications web performantes et sécurisées. Spring Boot a été choisi pour sa robustesse, sa facilité d'intégration avec des bases de données relationnelles, et la possibilité de rapidement déployer l'application en utilisant son écosystème complet. Ce choix a permis d'implémenter efficacement les API nécessaires pour la gestion des sessions, des indicateurs et des mesures.

Base de données : H2

Pour la gestion des données, nous avons utilisé **H2**, un système de gestion de base de données relationnelle open source, léger et embarqué. H2 est particulièrement adapté aux phases de développement grâce à sa simplicité de configuration et sa rapidité d'exécution. Il a permis de structurer les données efficacement en local et de tester les relations entre les entités dans un environnement de développement, avant un éventuel déploiement sur un SGBD plus robuste comme PostgreSQL.

Outils de développement : IntelliJ IDEA et Visual Studio Code

- **IntelliJ IDEA** a été utilisé pour le développement du backend avec Spring Boot. Il offre des outils puissants pour Java et Spring, permettant une productivité accrue grâce à ses fonctionnalités de complétion de code, de gestion des dépendances, et de débogage.
- **Visual Studio Code** a été utilisé pour le développement du frontend avec Vue.js. Cet éditeur léger et modulaire, enrichi d'extensions adaptées, facilite le développement avec Vue.js tout en offrant une expérience fluide pour la gestion des composants, des routes et des styles CSS.

Outils de gestion de version : Git et GitHub

Le projet a été géré avec **Git** comme système de contrôle de version, et les répertoires de code ont été hébergés sur **GitHub**. Cette configuration a permis de :

- Faciliter la gestion des versions du code source.
- Assurer une collaboration efficace entre les membres de l'équipe.
- Gérer les branches de développement de manière structurée.

Des branches spécifiques ont été créées pour chaque fonctionnalité ou amélioration, permettant une organisation claire du travail et limitant les conflits lors des *merges*. Cette méthode a contribué à un développement fluide, même en parallèle sur plusieurs tâches.

3.2 Organisation et gestion de l'intégration et du versioning

Une bonne gestion de l'intégration continue (**CI**) et du versioning a été essentielle pour maintenir la qualité du code et faciliter la collaboration tout au long du développement. Voici l'organisation mise en place :

Flux de travail Git

Nous avons adopté une stratégie de branches structurée pour garantir une organisation claire et éviter les conflits lors de l'intégration :

- **Branch master** : Représente la version stable et prête à être déployée en production. Chaque nouvelle version stable est fusionnée ici.
- **Branches dev/front et dev/back** : Spécifiquement dédiées au développement du frontend et du backend respectivement.
- **Branch démo** : Utilisée pour les démonstrations client. Dès que des fonctionnalités importantes sont prêtes, elles sont fusionnées dans cette branche afin de fournir une version fonctionnelle présentable lors des réunions.

Pull Requests (PR)

Les **Pull Requests** ont été utilisées pour la relecture de code et l'intégration des nouvelles fonctionnalités. Chaque branche de fonctionnalité était revue, testée, puis fusionnée dans **dev**, **master**, ou **démo**, garantissant un haut niveau de qualité du code partagé.

Intégration Continue (CI)

L'intégration continue a été mise en place afin d'automatiser les processus de test et de déploiement :

- **GitHub Actions** a été utilisé pour lancer automatiquement des tests unitaires et d'intégration à chaque push sur les branches principales.
- Cela a permis de détecter rapidement les erreurs, éviter les régressions, et garantir un code toujours fonctionnel.

Gestion des versions

- Les versions stables de l'application ont été étiquetées à l'aide de **tags Git** (ex : v1.0.0), facilitant le suivi de l'évolution.
- Un **changelog** a été maintenu pour documenter les ajouts, les améliorations et les corrections de bugs de chaque version.

Déploiement et Environnement

- **Environnement de développement** : Chaque membre de l'équipe disposait de son propre environnement local, avec des bases de données isolées pour éviter les interférences.
- **Environnement de production** : Le déploiement s'appuie sur **Kubernetes**, orchestrant les **containers Docker** pour gérer les microservices. Kubernetes assure la haute disponibilité, la scalabilité et la résilience nécessaires à une mise en production fiable.

Chapitre 4

Suivi de projet

4.1 Organisation et gestion ScrumBan

Le projet a suivi une méthodologie **ScrumBan**, une combinaison des pratiques de *Scrum* et de *Kanban*, permettant de bénéficier de la flexibilité de Kanban tout en conservant la structure de Scrum pour gérer les itérations et suivre l'avancement du projet.

ScrumBan Board

Nous avons utilisé **Trello** comme outil principal pour organiser notre projet et suivre l'avancement. Les tâches étaient organisées en colonnes : *À faire*, *En cours*, *En révision*, et *Terminé*, permettant à toute l'équipe de visualiser en temps réel l'état des tâches.

Les user stories ont été découpées en petites unités de travail pour faciliter leur gestion. Les sprints étaient organisés autour de ces stories, avec des objectifs précis à réaliser.

Lien vers le board Trello : <https://trello.com/invite/b/67cf03dbc0fbc002d38aa521/ATTIf83bd88d5cc6830f7e75d4b1541eef609A701C1C/running>

4.2 Répartition des rôles

- **Backend** : Yasmine Belarbi a développé le backend avec Spring Boot, assurant la gestion des sessions, utilisateurs, indicateurs et leur persistance en base.
- **Frontend** : Ethan Cabanes a développé le frontend avec Vue.js, en concevant les composants, les vues et en intégrant les appels API.
- **Gestion de projet** : L'ensemble du groupe a assuré la gestion de projet avec Trello pour planifier et suivre les tâches de manière agile.
- **Tests et validation** : Yasmine Belarbi et Ethan Cabanes ont pris en charge les tests (unitaires, d'intégration et fonctionnels).

4.3 Méthodes agiles et itérations

Implication du client

Le client a été impliqué tout au long du projet à travers des réunions régulières. Ces rencontres ont permis de présenter l'avancement, récolter des retours, ajuster les fonctionnalités et renforcer la satisfaction du client.

Sprints et itérations

Le projet a été découpé en **sprints de deux semaines**. À chaque sprint, des user stories ont été sélectionnées, découpées en tâches et intégrées dans Trello. Une revue de sprint clôturait chaque itération avec une démonstration au client.

Approche agile

Grâce à une gestion agile, les priorités ont été ajustées en continu. Les demandes du client ont été intégrées progressivement, permettant une amélioration continue de l'application et une meilleure réponse aux besoins réels.

4.4 Suivi de la qualité

Tests manuels

Des tests manuels ont été réalisés à chaque itération sur un environnement de staging configuré avec Kubernetes, simulant un environnement de production.

Suivi des bugs

Les bugs et régressions ont été suivis via **GitHub Issues** et **Trello**. Chaque problème détecté était immédiatement traité, documenté, et affecté à un membre de l'équipe.

4.5 Recettage et déploiement

Le recettage de l'application a été effectué à la fin de chaque sprint. Une fois les tests validés, les nouvelles fonctionnalités étaient fusionnées dans la branche **démo** puis présentées au client.

Une phase de **maintenance** est prévue après la livraison pour assurer la continuité et l'évolution du projet.

Diagramme de Gantt

Le diagramme de Gantt ci-dessous présente la planification globale du projet Athletica, en illustrant les différentes phases, les tâches associées et leur répartition dans le temps.

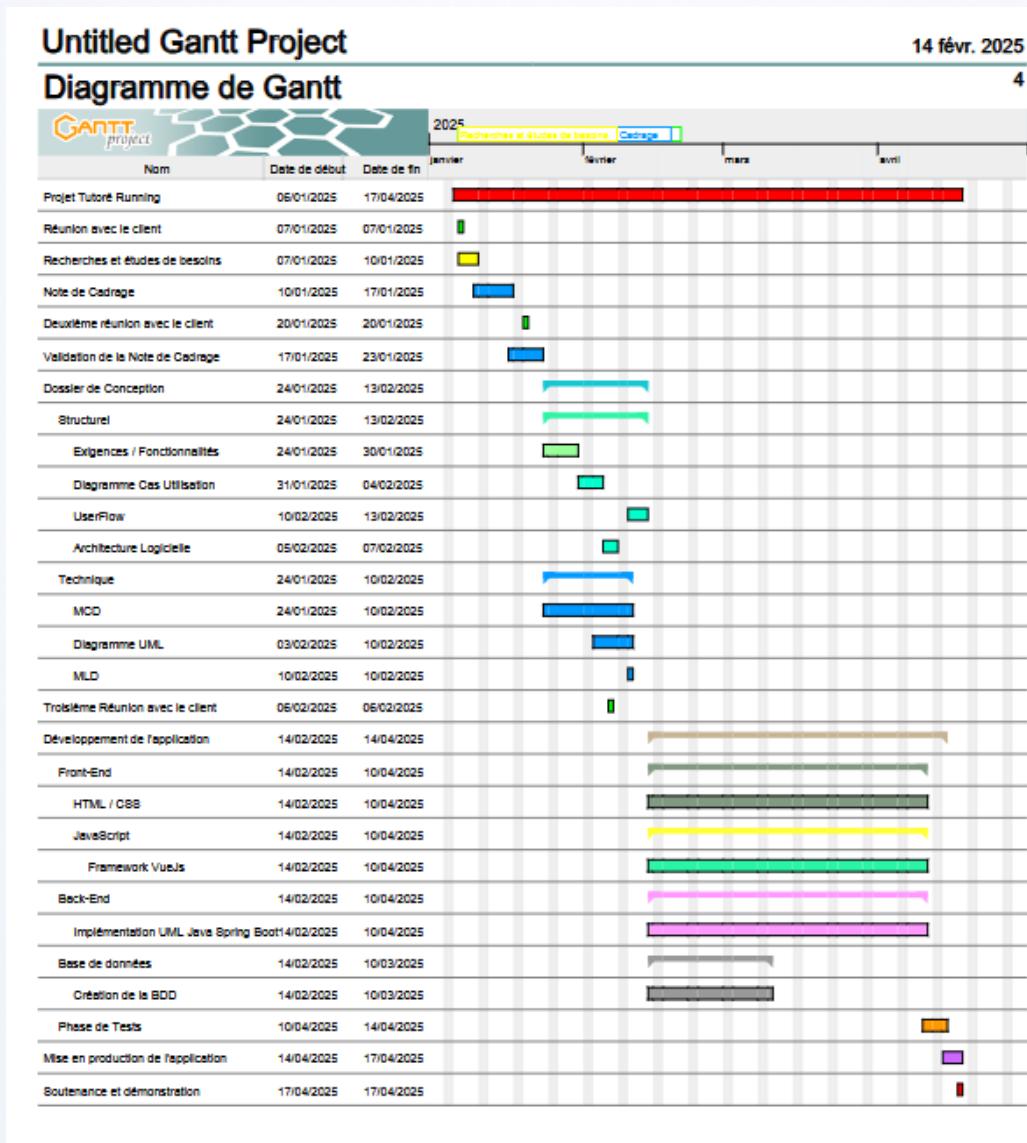


FIGURE 4.1 – Diagramme de Gantt du projet Athletica

Chapitre 5

Conclusion

Analyse de ce qui a été mis en place

Tout au long de ce projet, nous avons mis en place une solution robuste et évolutive pour permettre aux utilisateurs de suivre et d'analyser leurs performances sportives, principalement orientées vers la course à pied.

L'application, développée avec **Vue.js** pour le frontend et **Spring Boot** pour le backend, utilise **H2** pour la gestion des données, ce qui facilite le développement local et permet une gestion efficace des informations durant les phases de test et de prototypage.

Nous avons suivi une méthodologie **ScrumBan**, ce qui nous a permis de gérer le projet de manière flexible tout en restant alignés avec les attentes du client. La communication régulière avec le client a été primordiale pour ajuster les fonctionnalités selon les retours et garantir que l'application répondait aux besoins exprimés.

Difficultés rencontrées et solutions

Problèmes d'intégration des données : Lors de l'implémentation de la gestion des données et de la mise en relation des entités, nous avons rencontré des boucles infinies dans notre modèle de données. Cela a nécessité une révision complète du MCD. Une fois la structure des relations corrigée, l'intégration des données s'est déroulée sans encombre.

Gestion des indicateurs personnalisés : L'ajout de la fonctionnalité permettant aux utilisateurs de personnaliser leurs indicateurs a posé des défis techniques, notamment au niveau de la gestion dynamique des données. Nous avons surmonté ces difficultés en modifiant le modèle de données et en intégrant une gestion efficace des indicateurs globaux et de session.

Difficultés liées à la mise en production : Le choix de **Kubernetes** pour l'orchestration des containers et le déploiement de l'application a été une étape cruciale. Malgré quelques ajustements initiaux, Kubernetes a permis une gestion optimale des microservices, avec une scalabilité et une résilience accrues.

Paramétrage de Vuetify : La personnalisation de **Vuetify** pour obtenir un design moderne et épuré a représenté un défi important. L'intégration a nécessité plusieurs ajustements pour garantir que les composants s'adaptent à toutes les résolutions. Nous avons résolu ces problèmes en configurant les thèmes, breakpoints, et la disposition des composants.

Perspectives pour l'avenir

Partage avec les coachs : Permettre aux coachs de suivre les progrès de leurs athlètes et d'adapter les entraînements en conséquence.

Commentaires sur la page : Intégrer un système de commentaires pour que les utilisateurs puissent échanger des conseils et impressions sur leurs sessions.

Connecteurs pour mesures directes : Ajouter des connecteurs pour recueillir automatiquement les mesures via des appareils externes (montres connectées, capteurs, etc.).

État d'achèvement du projet et organisation de la livraison

Au moment du rendu de ce rapport, le projet est dans une phase **avancée de développement**. Toutes les fonctionnalités principales ont été implémentées et testées avec succès. L'application est **fonctionnelle** et prête à être utilisée.

Le **déploiement sur Kubernetes** a été effectué, garantissant une orchestration efficace des containers et des services. La dernière étape consiste à finaliser la documentation et organiser la **livraison finale** de l'application.

Un **test de validation global** sera réalisé pour s'assurer de la stabilité de l'application avant la mise en production. Enfin, une **réunion de clôture** avec le client est prévue pour présenter la version finale et recueillir les retours définitifs.