### 

| **TECHNIQUES DE DEVELOPPEMENT DES SYSTEMES** |
| --- |

### 

**Document d’analyse et de conception**

# **Plan du document**

[**Plan du document 2**](#_giaz436uut14)

[Titre du Projet : Application Web et Mobile de Suivi de Fitness 3](#_8f7wps69dspw)

[Introduction 3](#_3ntycw9d6g32)

[Analyse des Besoins 3](#_qjw3ampfbtff)

[1. Exigences Fonctionnelles 3](#_ra8bx7gsxrqr)

[2. Exigences Non Fonctionnelles 4](#_aqqo6obmr71e)

[Conception de l'Architecture 4](#_paaxhtvh4hqq)

[1. Architecture Globale 4](#_b1csca1nfowp)

[1. Frontend Mobile (Flutter) 5](#_k0q84e3fp661)

[2. Frontend Web (React.js/Angular) 6](#_itdftv105o3s)

[3. Backend (Node.js avec Express.js) 6](#_jgs6juvj8ve8)

[4. Bases de Données (Firestore et MongoDB) 7](#_c1tsnbg6tbkp)

[5. Services Firebase 7](#_pj8qgstlq0ba)

[Conception des Modules 8](#_vr7o777uexo5)

[1. Module d'Authentification 8](#_o5n223djwowa)

[2. Module de Suivi des Séances 8](#_n2j1zkjh1ktk)

[3. Module de Gestion des Objectifs 8](#_x45fhc62tnwr)

[4. Module de Visualisation des Statistiques 8](#_joz5kpp31x45)

[5. Module de Notifications 8](#_qesgfjs73wi)

[6. Interface Utilisateur 8](#_96wvals7q113)

[Méthodologie de Développement 9](#_3l7ptg8dcyw3)

[1. Approche Agile 9](#_9c5veo1jky8l)

[2. Intégration Continue et Déploiement Continu (CI/CD) 9](#_khlzzxay3cke)

[3. Documentation et Collaboration 10](#_itw663d1w81g)

[4. Maintenance et Amélioration Continue 11](#_huuamhn6xab6)

[Diagramme UML 11](#_s9y6374w7cc7)

[1. Diagramme des Cas d'Utilisation 11](#_nc97k1ft6zl8)

[Description UML du Diagramme de Cas d'Utilisation 11](#_6hbhrf9d0d3e)

[Acteurs : 12](#_evhltg3i4m0d)

[Cas d'Utilisation Principaux : 12](#_jl9islmqa6ll)

[1.0. Diagramme d’activité pour l’inscription/connexion : 13](#_occ8txvkt2pe)

[1.1. Diagramme d’activité pour le suivi des séances : 14](#_wcyckaa2lw60)

[1.2. Diagramme d’activité pour le suivi et la définition des objectifs: 15](#_eu8hay17jhnc)

[1.3. Diagramme d’activité de la visualisation des statistiques: 16](#_cmw5ct97985o)

[1.4. Diagramme de Séquence 16](#_e5755emkkjbt)

[Description du Diagramme de Séquence 17](#_dk07eu7fddg0)

[1.5. Modèle de Données 18](#_467snxzcxgf)

[Diagramme ERD (Entity-Relationship) 18](#_kij7gr654d8i)

[Plan de Tests et Validation 19](#_eke07x4v5j77)

[Conclusion 20](#_i8wbzh1ycmh4)

## 

## **Titre du Projet :** Application Web et Mobile de Suivi de Fitness

### **Introduction**

Ce document d'analyse et de conception détaille les aspects techniques et fonctionnels nécessaires à la réalisation de l'application de suivi de fitness. Il comprend l'architecture du système, la conception des modules, ainsi qu'une description détaillée des interfaces utilisateur.

### **Analyse des Besoins**

#### **Exigences Fonctionnelles**

* **Authentification et Gestion des Utilisateurs** :
  + Inscription/Connexion via email et mot de passe.
  + Authentification via Google et Facebook.
* **Suivi des Séances d'Entraînement** :
  + Enregistrement manuel des types d'exercice, de la durée et des calories brûlées.
  + Suivi automatique des activités physiques via GPS et accéléromètre.
  + Accès à l'historique des séances avec des filtres (type, date).
* **Définition et Suivi des Objectifs de Fitness** :
  + Paramétrage d'objectifs personnels comme courir 5 km ou perdre 5 kg.
  + Visualisation des progrès vers ces objectifs avec des graphiques et notifications.
  + Attribution de récompenses (badges) pour l'accomplissement d'objectifs.
* **Visualisation des Statistiques et des Progrès** :
  + Tableaux de bord pour afficher les progrès sur différentes périodes.
  + Analyse des performances et des tendances (meilleurs temps, etc.).
  + Comparaison des performances actuelles avec les périodes précédentes.
* **Notifications et Rappels** :
  + Envoi de notifications push pour rappeler les objectifs quotidiens et les séances planifiées.
  + Notifications pour informer de l'obtention de nouveaux badges.
* **Interface Utilisateur Attrayante et Intuitive** :
  + Design responsive et moderne.
  + Navigation intuitive via une barre de navigation ou un menu latéral.
  + Utilisation de graphiques et animations pour améliorer l'expérience utilisateur.
* **Fonctionnalités Sociales (Optionnel)** :
  + Partage des progrès sur les réseaux sociaux.
  + Défis et compétitions entre amis.
  + Encouragements et commentaires entre utilisateurs.

#### **Exigences Non Fonctionnelles**

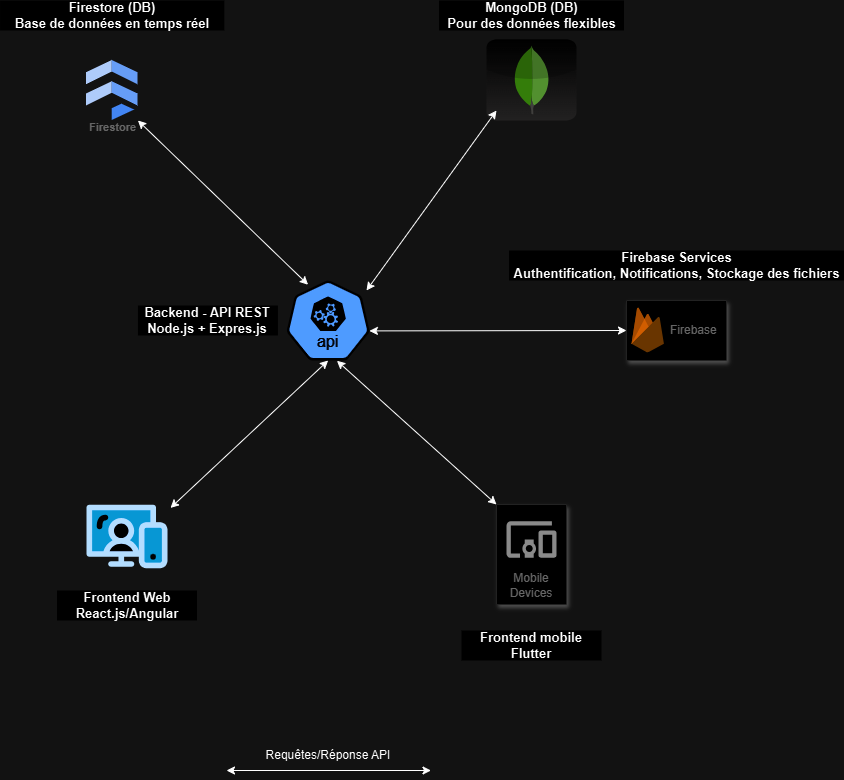
* **Performance** : Application réactive sans latence perceptible.
* **Sécurité** : Protection des données utilisateur et sécurité des authentifications.
* **Compatibilité** : Fonctionnement optimal sur iOS, Android, et navigateurs modernes.
* **Scalabilité** : Capacité à gérer un nombre croissant d'utilisateurs sans dégradation des performances.

### **Conception de l'Architecture**

#### **Architecture Globale**

* **Frontend Mobile** : Développement avec Flutter, communication via API REST.
* **Frontend Web** : Développement avec React.js ou Angular, communication via API REST.
* **Backend** : Utilisation de Node.js avec Express.js pour la logique serveur et les API.
* **Base de Données** : Firestore pour des données en temps réel et MongoDB pour une structure flexible.
* **Services Supplémentaires** : Firebase pour authentification, notifications, et stockage.

**Diagramme d'Architecture du système**



**Description du Diagramme d'Architecture du Système**

Le diagramme d'architecture présenté ci-dessus illustre l'infrastructure technique de l'application de suivi de fitness, comprenant les composants frontend, backend, les bases de données et les services auxiliaires.

#### **Frontend Mobile (Flutter)**

* **Rôle :** Cette composante représente l'application mobile développée avec Flutter. Elle permet aux utilisateurs d'interagir avec le système via leurs appareils mobiles.
* **Interaction :** Le frontend mobile communique avec le backend en envoyant des requêtes HTTP via des API REST pour récupérer ou envoyer des données (ex. enregistrement des séances d'entraînement, suivi des objectifs). Les réponses sont également reçues du backend pour afficher les informations pertinentes sur l'interface utilisateur mobile.

#### **Frontend Web (**[**React.js/Angular**](http://react.js/Angular)**)**

* **Rôle :** Le frontend web, développé avec React.js ou Angular, permet aux utilisateurs d'accéder à l'application via un navigateur web. Ce composant est responsable de l'affichage et de la gestion des interactions utilisateur sur le web.
* **Interaction :** De manière similaire au frontend mobile, le frontend web communique avec le backend via des API REST. Les requêtes sont envoyées pour diverses opérations telles que l'authentification, la consultation des statistiques, et la gestion des objectifs, avec des réponses renvoyées par le backend pour mise à jour dynamique de l'interface utilisateur.

#### **Backend (Node.js avec Express.js)**

**Rôle :** Le backend, développé avec Node.js et Express.js, est le cœur du système. Il gère la logique métier, les règles d’authentification, le traitement des données, et la communication avec les bases de données et les services externes.

**Interaction :** Le backend reçoit les requêtes des frontends via l'API REST, traite ces requêtes en interagissant avec les bases de données (Firestore et MongoDB) et les services Firebase, puis renvoie les réponses appropriées aux frontends.

#### **Bases de Données (Firestore et MongoDB)**

* **Firestore :**
  + **Rôle :** Firestore est utilisé comme base de données en temps réel, idéal pour les fonctionnalités nécessitant des mises à jour immédiates, telles que le suivi en direct des activités physiques.
  + **Interaction :** Le backend interagit avec Firestore pour stocker et récupérer des données en temps réel, facilitant des mises à jour instantanées côté frontend.
* **MongoDB :**
  + **Rôle :** MongoDB est utilisé pour stocker des données structurées et non structurées, offrant flexibilité et scalabilité pour le stockage des informations utilisateur, des statistiques, et des historiques d'entraînement.
  + **Interaction :** Le backend interagit avec MongoDB pour les opérations de lecture et d’écriture des données complexes, telles que les historiques des utilisateurs et les configurations personnalisées.

#### **Services Firebase**

* **Rôle :** Firebase est utilisé pour des services essentiels tels que l'authentification des utilisateurs, les notifications push, et le stockage de fichiers.
* **Interaction :** Le backend envoie des requêtes à Firebase pour des opérations d’authentification, l’envoi de notifications aux utilisateurs, et la gestion des fichiers (comme les photos de profil des utilisateurs). Les réponses sont ensuite traitées par le backend et relayées aux frontends.

En résumé, le diagramme d'architecture montre une vue d'ensemble claire et concise des composants et de leurs interactions dans l'application de suivi de fitness. Les connexions bien définies entre les frontends, le backend, les bases de données, et les services Firebase assurent une communication efficace et une expérience utilisateur optimisée

### **Conception des Modules**

#### **Module d'Authentification**

* Gestion des inscriptions, connexions, et authentifications via Google et Facebook.
* Sécurisation des sessions utilisateur via Firebase Authentication.

#### **Module de Suivi des Séances**

* Enregistrement manuel et automatique des données d'entraînement.
* Intégration avec les capteurs (GPS, accéléromètre) pour un suivi précis des activités.

#### **Module de Gestion des Objectifs**

* Définition des objectifs et suivi des progrès avec rappels via notifications.
* Attribution de badges pour la motivation des utilisateurs.

#### **Module de Visualisation des Statistiques**

* Création de tableaux de bord dynamiques avec graphiques (courbes, histogrammes).
* Analyse des performances avec comparaisons et tendances.

#### **Module de Notifications**

* Implémentation de notifications push via Firebase Cloud Messaging pour rappels et récompenses.

#### **Interface Utilisateur**

**Conception des Interfaces Utilisateur**

* **Mobile** :
  + **Écran d'Accueil** : Présentation des objectifs actuels et des dernières séances.
  + **Écran de Suivi** : Interface intuitive pour enregistrer ou suivre une activité en temps réel.
  + **Écran de Statistiques** : Affichage des progrès avec des graphiques dynamiques et responsives.
  + **Navigation** : Barre de navigation en bas de l'écran pour un accès rapide aux sections principales (Accueil, Suivi, Statistiques, Profil).
* **Web** :
  + **Dashboard Principal** : Vue d'ensemble des activités et des objectifs, avec des graphiques interactifs.
  + **Page d'Historique** : Liste filtrable des séances avec détails sur chaque activité.
  + **Page de Paramètres** : Gestion du profil utilisateur, des préférences, et des paramètres de notification.
  + **Navigation** : Menu latéral pour une navigation fluide entre les différentes sections (Dashboard, Historique, Objectifs, Paramètres).

### **Méthodologie de Développement**

#### **Approche Agile**

* **Méthodologie** : Utilisation de **Scrum** pour le développement itératif de l'application. Les fonctionnalités sont développées en **sprints**, généralement de deux semaines, avec des **rétrospectives** régulières pour ajuster le travail en fonction des feedbacks des utilisateurs et des parties prenantes. Cette approche favorise l'amélioration continue et l'adaptabilité aux changements de besoins.
* **User Stories** : Création de **user stories** pour décrire les fonctionnalités du point de vue de l'utilisateur final. Chaque story est accompagnée de critères d'acceptation clairs. Par exemple, "En tant qu'utilisateur, je veux pouvoir enregistrer une séance d'entraînement pour suivre mes progrès." Cette méthode assure que le développement reste centré sur l'utilisateur et ses besoins.
* **Product Backlog et Priorisation** : Le backlog du produit est géré en continu, avec une priorisation basée sur la valeur ajoutée pour l'utilisateur et les objectifs du projet. Cela permet de garantir que les fonctionnalités les plus importantes sont développées en premier.

#### **Intégration Continue et Déploiement Continu (CI/CD)**

* **CI/CD Pipeline** : Mise en place d'un pipeline **CI/CD** pour automatiser les tests (unitaires, d'intégration, de régression) et le déploiement sur les environnements de développement, de test et de production. Cette automatisation réduit les risques d'erreurs humaines et permet de livrer les nouvelles fonctionnalités plus rapidement et de manière plus fiable.
* **Outils de CI/CD** : Utilisation d'outils tels que **Jenkins, GitLab CI/CD** ou **GitHub Actions** pour automatiser le pipeline de déploiement. Ces outils permettent d'intégrer les tests automatisés et de déclencher des déploiements en continu à chaque modification du code source.
* **Docker et Kubernetes** :
  + **Docker** : Containerisation des services à l'aide de Docker, garantissant que l'application fonctionne de manière cohérente dans différents environnements (développement, test, production). Cela simplifie la gestion des dépendances et l'évolutivité de l'application.
  + **Kubernetes** : Orchestration des conteneurs à grande échelle avec **Kubernetes**, permettant une gestion dynamique des déploiements, une mise à l'échelle automatique, et une gestion simplifiée des défaillances (self-healing). Kubernetes est particulièrement utile pour déployer l'application dans des environnements de production complexes.
* **Surveillance et Logging** : Intégration d'outils de **monitoring** (comme **Prometheus** ou **Grafana**) et de **logging** (comme **ELK Stack**) pour surveiller les performances de l'application, détecter les anomalies et analyser les logs en temps réel. Cela assure une maintenance proactive et rapide en cas de problème.
* **Tests Automatisés et Qualité du Code** :
  + **Tests automatisés** : Mise en place de tests automatisés couvrant l'ensemble des fonctionnalités critiques, garantissant ainsi que chaque modification du code ne casse pas les fonctionnalités existantes.
  + **Analyse de la qualité du code** : Utilisation d'outils comme **SonarQube** pour analyser la qualité du code, détecter les mauvaises pratiques, et assurer que le code respecte les standards définis.

#### **Documentation et Collaboration**

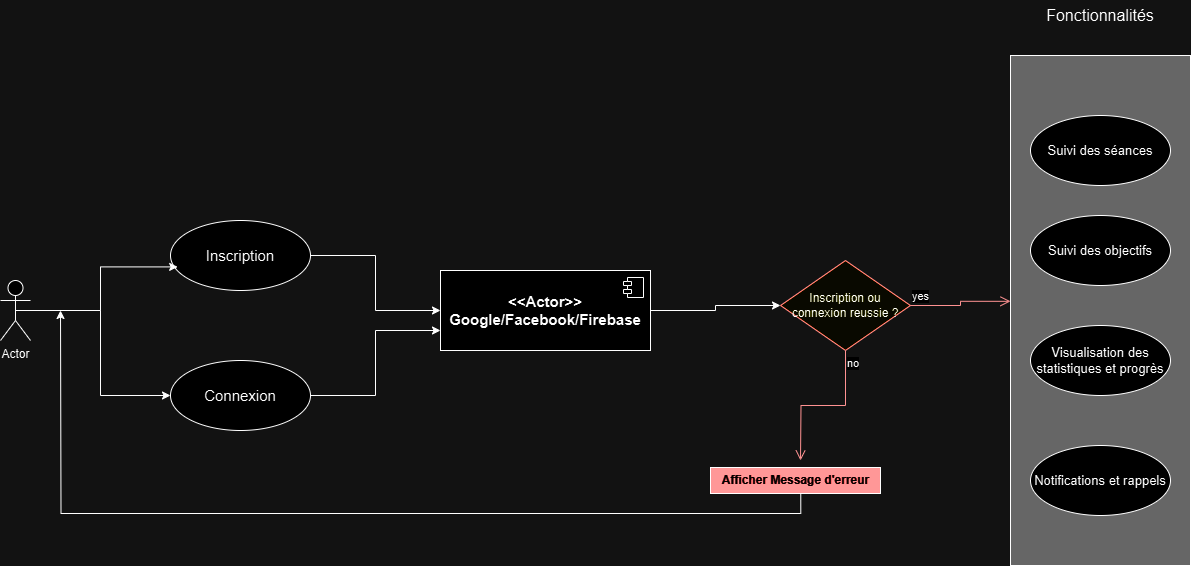
* **Documentation** : Utilisation de **tools de documentation** tels que **Confluence** pour centraliser la documentation du projet. Chaque sprint est accompagné d'une documentation mise à jour qui décrit les nouvelles fonctionnalités, les API, et les schémas de données.
* **Collaboration** : Utilisation de **Jira** ou **Trello** pour la gestion des tâches et la collaboration entre les équipes. Ces outils permettent de suivre l'avancement du projet et d'assurer une communication efficace entre les développeurs, les testeurs, et les parties prenantes.

#### **Maintenance et Amélioration Continue**

* **Feedback Utilisateur** : Collecte continue des retours d'expérience des utilisateurs pour identifier les opportunités d'amélioration et prioriser les évolutions du produit.
* **Mises à jour régulières** : Mise en place d'un processus de **release management** pour publier des mises à jour régulières, incluant des corrections de bugs, des améliorations de performance et de nouvelles fonctionnalités.

### **Diagramme UML**

#### **Diagramme des Cas d'Utilisation**



#### **Description UML du Diagramme de Cas d'Utilisation**

Le diagramme UML des cas d'utilisation pour l'application de suivi de fitness représente les interactions entre les acteurs principaux (les utilisateurs) et le système. Chaque cas d'utilisation correspond à une fonctionnalité principale de l'application.

#### **Acteurs :**

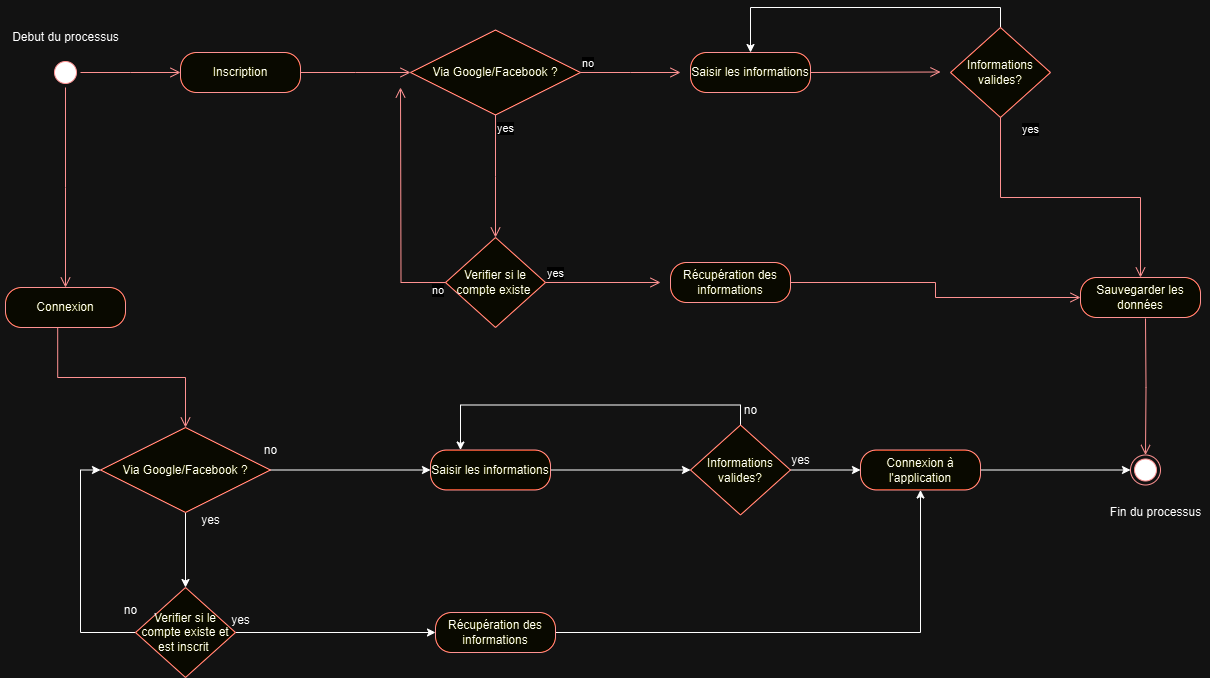
1. **Utilisateur :**
   * Un utilisateur standard de l'application, qui utilise les fonctionnalités pour suivre ses activités de fitness, définir des objectifs, consulter des statistiques, et recevoir des notifications.
2. **Services Externes :**
   * **Google/Facebook :** Services d'authentification utilisés pour permettre aux utilisateurs de se connecter à l'application.
   * **Firebase :** Services utilisés pour l'authentification, les notifications push et le stockage de fichiers.

#### **Cas d'Utilisation Principaux :**

*N.B : Dans les diagrammes suivants , on suppose que l’utilisateur est connecté à l’application*

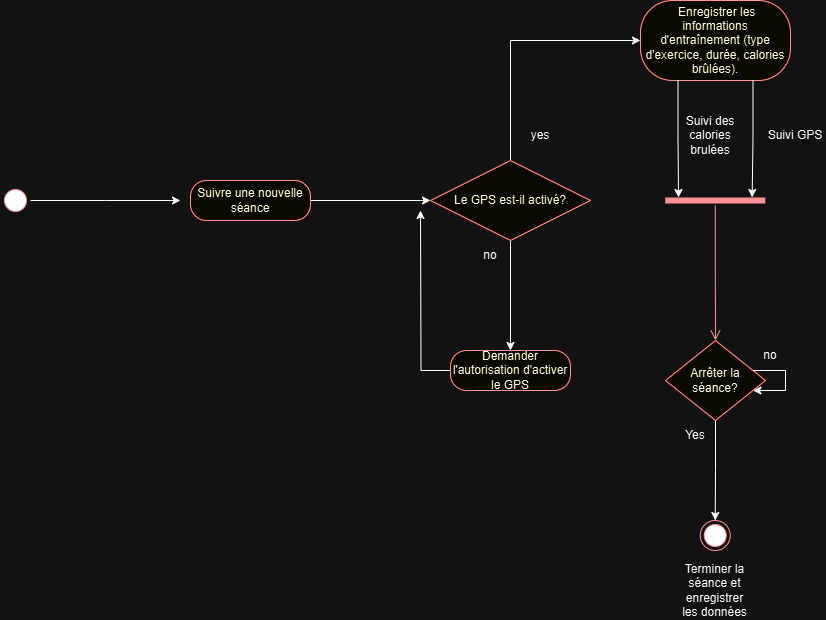
1. **Inscription/Connexion :**
   * **Description :** Permet à l'utilisateur de créer un compte ou de se connecter avec des identifiants traditionnels (email et mot de passe) ou via Google/Facebook. Ce cas d'utilisation est essentiel pour accéder aux autres fonctionnalités de l'application.
   * **Acteur Principal :** Utilisateur.
   * **Extensions :** Gestion des erreurs lors de la saisie des informations de connexion.

##### **1.0. Diagramme d’activité pour l’inscription/connexion :**

****

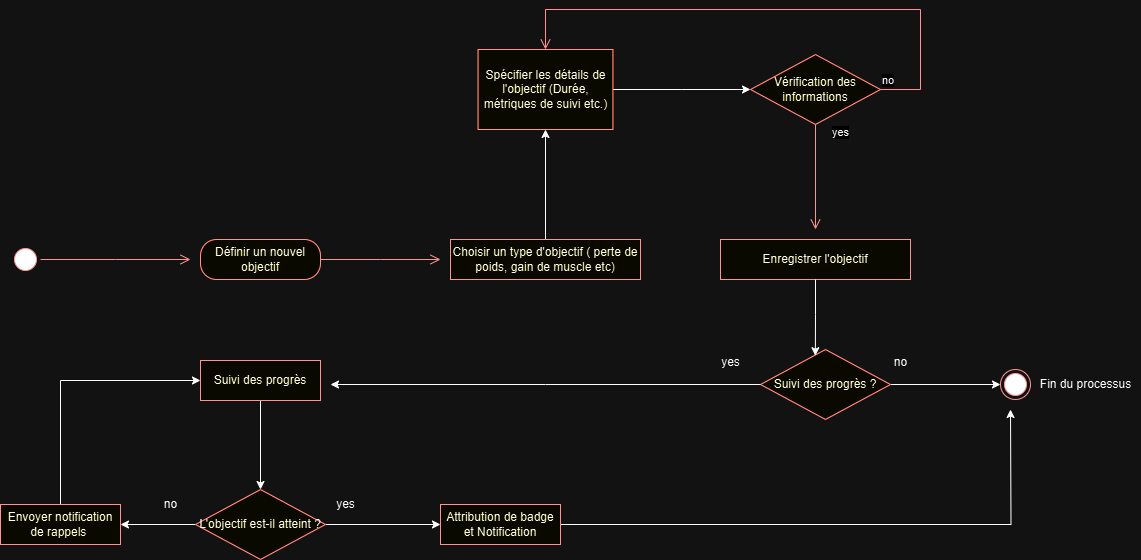
1. **Suivi des Séances d'Entraînement :**
   * **Description :** L'utilisateur peut enregistrer ses entraînements manuellement ou via des capteurs (GPS, accéléromètre). Les données sont ensuite stockées et affichées dans l'application.
   * **Acteur Principal :** Utilisateur.
   * **Extensions :** Correction manuelle des données si le suivi automatique échoue.

##### **Diagramme d’activité pour le suivi des séances :**

****

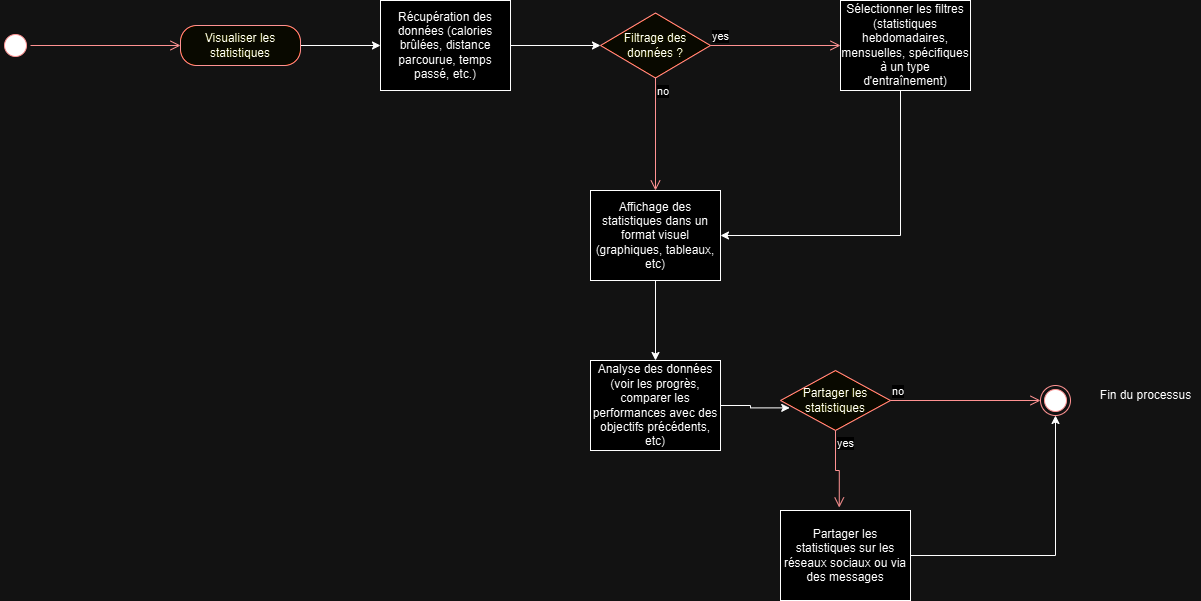
1. **Définition et Suivi des Objectifs :**
   * **Description :** L'utilisateur peut définir des objectifs personnels (ex : courir 5 km) et suivre ses progrès au fil du temps. Des rappels et des récompenses (badges) sont envoyés pour motiver l'utilisateur.
   * **Acteur Principal :** Utilisateur.
   * **Extensions :** Notifications pour les rappels d'objectifs.

##### **Diagramme d’activité pour le suivi et la définition des objectifs:**



1. **Visualisation des Statistiques :**
   * **Description :** Affiche un tableau de bord avec des graphiques montrant les progrès de l'utilisateur. L'utilisateur peut comparer ses performances sur différentes périodes et analyser ses tendances d'entraînement.
   * **Acteur Principal :** Utilisateur.
   * **Extensions :** Filtrage des statistiques pour obtenir des informations plus spécifiques.

##### **Diagramme d’activité de la visualisation des statistiques:**

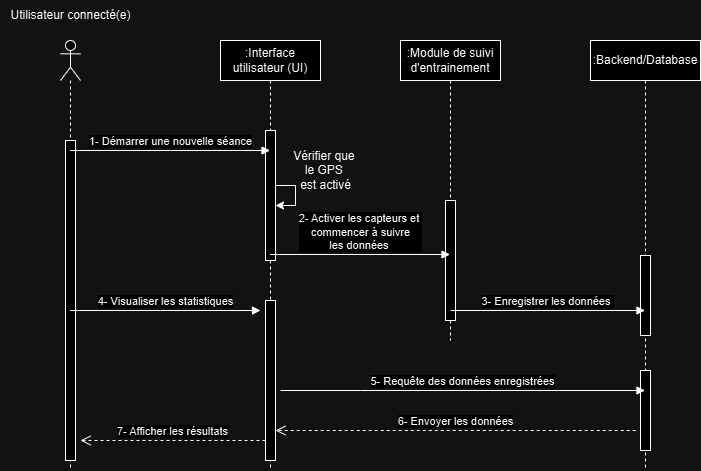
****

1. **Notifications et Rappels :**
   * **Description :** Le système envoie des notifications push pour rappeler à l'utilisateur ses objectifs quotidiens et l'informe de l'obtention de nouvelles récompenses.
   * **Acteur Principal :** Utilisateur.
   * **Extensions :** Désactivation des notifications.

##### **Diagramme de Séquence**

**Description** : Le diagramme de séquence décrit le flux d'une interaction typique entre l'utilisateur et le système, étape par étape. Par exemple, il montre comment un utilisateur démarre une séance d'entraînement, enregistre les données, et visualise les résultats.

**Diagramme de séquence pour le suivi d’une séance d'entraînement et la visualisation des statistiques**

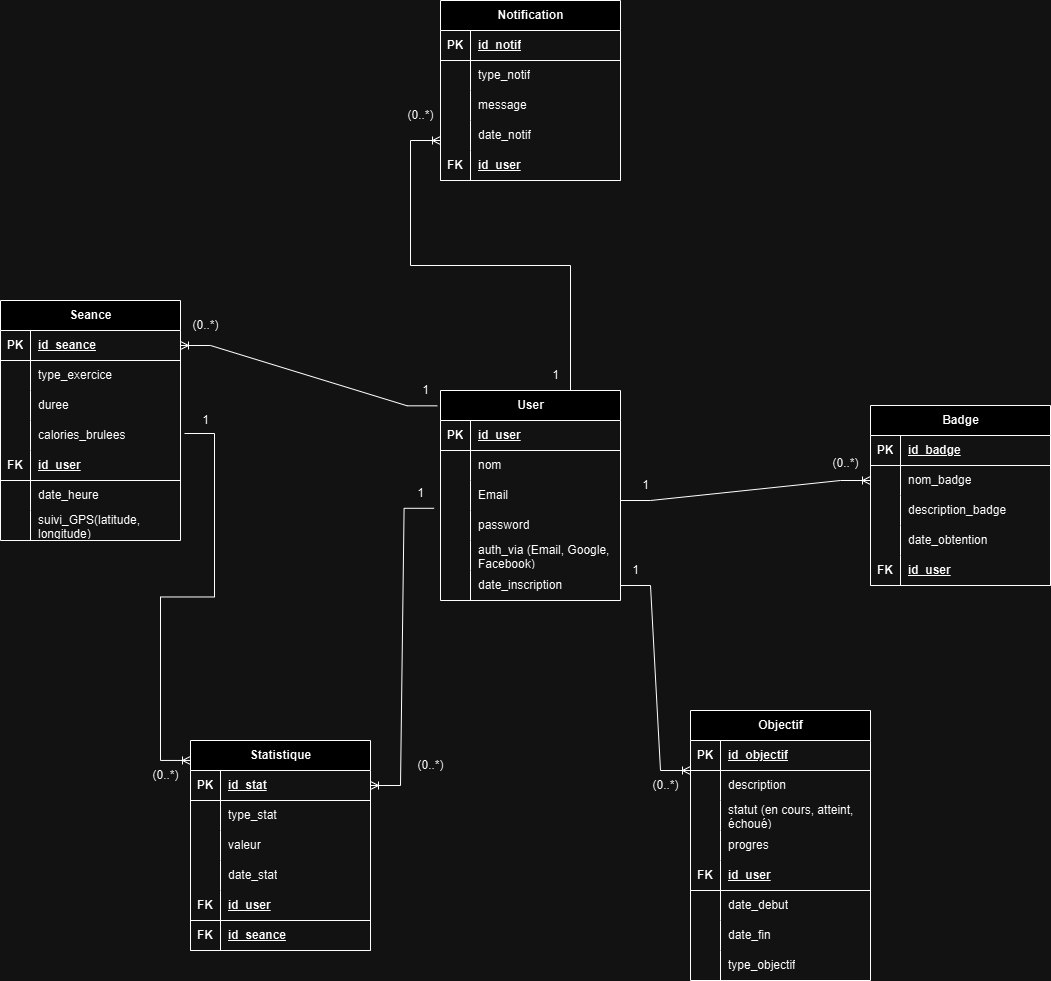
****

###### **Description du Diagramme de Séquence**

1. **Démarrage de la séance :**
   * L'utilisateur connecté(e) interagit avec l'interface utilisateur (UI) pour démarrer une nouvelle séance d'entraînement.
   * L'interface utilisateur (UI) vérifie si le GPS est activé avant de commencer le suivi de la séance.
2. **Activation des capteurs et suivi des données :**
   * Si le GPS est activé, l'interface utilisateur envoie une commande au module de suivi d'entraînement pour activer les capteurs et commencer à suivre les données d'entraînement (par exemple, la distance parcourue, les calories brûlées).
3. **Enregistrement des données :**
   * Le module de suivi d'entraînement enregistre les données collectées en temps réel et les envoie au backend ou à la base de données pour stockage.
4. **Visualisation des statistiques :**
   * Après la séance, l'utilisateur demande à visualiser les statistiques.
   * L'interface utilisateur envoie une requête au backend/base de données pour récupérer les données enregistrées.
5. **Affichage des résultats :**
   * Les données récupérées sont envoyées par le backend à l'interface utilisateur, qui affiche ensuite les résultats à l'utilisateur.

#### **Modèle de Données**

##### **Diagramme ERD (Entity-Relationship)**

* Le diagramme ERD détaillera les entités principales telles que **Utilisateurs, Séances, Objectifs, Statistiques, Badges, Notifications** et leurs relations. Les attributs des entités intégreront les détails spécifiques nécessaires pour chaque fonctionnalité (par exemple, type d'exercice, calories brûlées pour l'entité Séances)

### **Plan de Tests et Validation**

**Types de Tests :**

* **Tests Unitaires :** Chaque module et composant de l'application sera testé individuellement pour s'assurer qu'il fonctionne correctement. Par exemple, les tests unitaires vérifieront la logique derrière l'enregistrement des séances d'entraînement, le calcul des statistiques, etc.
* **Tests d'Intégration :** Ces tests s'assureront que les différents modules de l'application fonctionnent ensemble sans problème. Par exemple, vérifier l'intégration entre le module d'authentification et le suivi des séances.
* **Tests Fonctionnels :** Ils valident que chaque fonctionnalité de l'application répond aux exigences définies, telles que l'inscription, l'enregistrement des séances, la définition des objectifs, etc.
* **Tests de Performance :** Ils vérifieront la réactivité et la stabilité de l'application sous différentes charges, par exemple, en simulant plusieurs utilisateurs simultanés pour voir si l'application répond bien.
* **Tests de Sécurité :** Ils s'assureront que les données des utilisateurs sont protégées, en testant des scénarios tels que les tentatives de connexion non autorisées et la protection des données sensibles.

**Outils de Tests :**

* **JUnit:** Pour les tests unitaires automatisés.
* **Selenium:** Pour les tests d'interface utilisateur automatisés.
* **JMeter:** Pour les tests de performance.
* **OWASP ZAP:** Pour les tests de sécurité.

**Stratégie de Validation :**

* **Tests Alpha et Bêta :** Des versions alpha et bêta de l'application seront partagées avec des groupes restreints d'utilisateurs pour obtenir des retours avant le lancement public.
* **Validation des User Stories :** Chaque user story sera validée pour s'assurer que la fonctionnalité répond bien aux attentes de l'utilisateur final.

**Gestion des Bugs :**

* **Outil de Suivi des Bugs :** Jira sera utilisé pour suivre les bugs et les problèmes rencontrés durant les tests. Chaque bug sera classé par priorité et gravité pour être résolu rapidement.

### **Conclusion**

Le document d'analyse et de conception présente une vision claire et détaillée de la structure et des fonctionnalités de l'application de suivi de fitness. La conception est alignée avec les exigences fonctionnelles pour assurer une expérience utilisateur fluide et une gestion efficace des données. La prochaine phase implique l'implémentation de ces concepts en suivant le calendrier prévu.