

# **Améliorez votre application avec des systèmes distribués**

# **Documentation fonctionnelle et technique**

## **Sommaire**

**1. Présentation du projet**

**1.1 Objectifs du projet**

**1.2 Hors du champ d’application**

**1.3 Mesures du projet**

**2. Spécifications fonctionnelles**

**3. Spécifications techniques**

**3.1 Schéma de conception technique**

**3.2 Glossaire**

**3.3 Solutions techniques**

**3.4 Autres solutions non retenues**

1. **Présentation du projet**

L'application TourGuide, en pleine expansion, accueille actuellement plus de 30 000 utilisateurs connectés simultanément, avec une prévision de 100 000 utilisateurs dans un avenir proche. Les objectifs actuels incluent l'amélioration de la plateforme pour répondre aux besoins croissants des utilisateurs et de l'équipe de développement. L'accent est mis sur l'optimisation de l'expérience utilisateur, la gestion efficace de l'expansion et la simplification de la maintenance continue de l'application. Cette initiative vise à garantir la satisfaction des utilisateurs tout en soutenant la croissance de l'application.

### Objectifs du projet

Les objectifs actuels incluent la résolution des problèmes de lenteur lors des appels à gpsUtil et RewardsCentral, ainsi que l'optimisation des tests unitaires et la mise en place d'un pipeline d'intégration continue. Ces efforts visent à garantir une expérience utilisateur fluide et à renforcer la qualité et la fiabilité de l'application TourGuide.

### Hors du champ d’application

Dans une optique d'amélioration, une piste envisagée concerne l'optimisation de l'accès à la liste d'attractions. L'idée serait de charger ces attractions une fois par jour, évitant ainsi les appels répétés au service GpsUtil à chaque utilisation. Une telle démarche pourrait réduire la dépendance de l'application à des services externes tout en renforçant ses performances globales.

### Mesures du projet

Les métriques du projet seront harmonisées avec le guide des étapes clés, nous permettant ainsi de suivre de près l'évolution globale tout en progressant à travers les différentes étapes et points de vigilance définis dans le guide correspondant. Cette approche nous assure une trajectoire claire et nous guide vers le succès du projet.

La validation des différents tests de performance à l'aide d'une durée limite de temps, fournit une métrique précise à suivre. Cette approche garantit la qualité et la fiabilité de l'application tout au long de son développement.

## **Spécifications fonctionnelles**

Les différentes fonctionnalités de l'application incluent :

* Récupérer la localisation de l'utilisateur.
* Récupérer les attractions touristiques à proximité de l'utilisateur.
* Obtenir des réductions sur les séjours à l'hôtel / les billets de spectacle.
* Calculer les récompenses de l'utilisateur en fonction des attractions touristiques visitées.
* Estimer les récompenses potentielles des cinq attractions à proximité.

## **3. Spécifications techniques**

### 

### 3.1 Schémas de conception technique

Une image contenant texte, capture d’écran, Police, ligne

Description générée automatiquement

**Services externes :**

**GpsUtil :**

Renvoie la localisation de l’utilisateur.

Renvoie la liste des attractions partenaires.

**TripPricer :**

Fournit des réductions sur différents séjours/attractions en fonction des points de l’utilisateur.

**RewardsCentral :**

Détermine les points de récompense en fonction de l’attraction

**Services & Controller internes :**

**TourGuideController :**

Gère les requêtes HTTP pour la localisation, les attractions, les récompenses et les réductions.

**TourGuideService :**

Fournit une gamme de méthodes permettant d'appeler la localisation de l'utilisateur, de récupérer des attractions à proximité et d'obtenir les réductions des séjours

**RewardsService :**

Fournit des méthodes de calcul pour attribuer les récompenses aux utilisateurs.

### 3.2 Glossaire

**Localisation :** Emplacement définit par sa latitude et sa longitude.

### 3.3 Solutions techniques

***Tests unitaires qui échouent :***

L'utilisation de la classe « CopyOnWriteArrayList » est préconisée pour éviter les exceptions liées à la concurrence.

**Les 5 attractions les plus proches :**

Création de DTO « data transfer objects » pour récupérer les données nécessaires.

**Amélioration des performances :**

La mise en place de CompletableFuture pour implémenter le multithreading sur les méthodes nécessitant plus de ressources.

**Pipeline d’intégration continue :**

La mise en place d’un pipeline d’intégration via GitHub Action pour garantir une intégration continue et efficace.

### 3.4 Autres solutions non retenues

**Tests unitaires qui échouent :**

L’utilisation des blocs ou des méthodes synchronized a été envisagée pour résoudre les problèmes de concurrence. Cependant, cette approche n'a pas été retenue en raison de ses limitations en matière d'accès multithread.

***Améliorez les performances :***

La mise en place de CompletableFuture avec un Executor : newSingleThreadExecutor.

La problématique étant une consommation de ressource trop importante sur le pipeline d’intégration en continue (GitHub Action) empêchant la validation du workflow.