TourGuide

[TourGuide]

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 Schémas de structure technique

3.2 Diagramme de classes UML (Logique Métier)

3.3 Glossaire

3.4 Solutions techniques

3.5 Autres solutions non retenues

3.6 Annexes

1. Présentation du projet

TourGuide est une application ASP.NET Core Web API qui permet aux utilisateurs de localiser des attractions touristiques proches et d'accéder à des offres spéciales pour les séjours hôteliers et les billets de spectacles. Public cible : les voyageurs souhaitant découvrir des attractions et profiter de réductions touristiques. Avantages commerciaux : augmenter l'attrait et la satisfaction des clients en leur fournissant des recommandations personnalisées et des offres, ce qui encourage leur fidélisation.

1.1 Objectifs du projet

Optimiser les performances pour gérer efficacement une augmentation significative d'utilisateurs, passant de centaines à potentiellement 100 000 utilisateurs quotidiens.

Réduire les temps de réponse pour une meilleure expérience utilisateur, notamment ceux de GpsUtil et RewardsCentral.

Corriger les bugs dans les tests unitaires et garantir des recommandations touristiques adaptées pour tous les utilisateurs, quel que soit leur emplacement.

Mettre en place un pipeline d'intégration continue qui permettra de compiler, tester et construire les Dll qui seront téléchargeables sous la forme d'une archive Zip.

1.2 Hors du champ d'application

Développement d'une nouvelle interface utilisateur ou d'un front-end.

Ajout de nouvelles fonctionnalités majeures, autres que les corrections de bugs et l'optimisation des performances comme le développement d'itinéraire le plus rapide selon la position de l'utilisateur ou un système de note pour les différentes attractions avec des recommandations personnalisées selon les notes données.

1.3 Mesures du projet

Réduction significative des temps de réponse pour les services critiques, GpsUtil et RewardsCentral, en dessous des valeurs actuelles mesurées.

Validation de la capacité de l'application à gérer 100 000 utilisateurs simultanés grâce à une suite de tests de performances.

Amélioration des avis et commentaires des utilisateurs sur les performances et la pertinence des recommandations.

2. Spécifications fonctionnelles

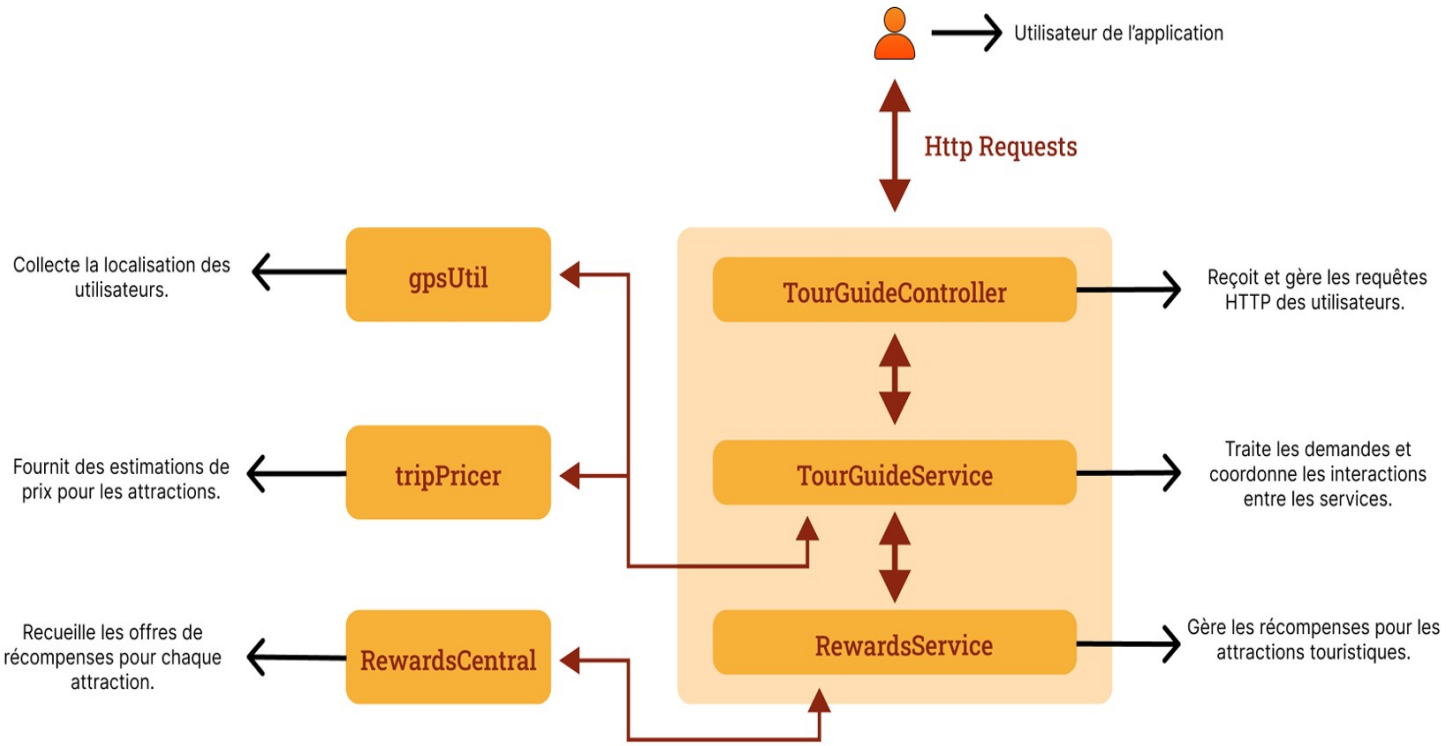
Localisation des attractions : Permettre aux utilisateurs de localiser des attractions touristiques à proximité en utilisant le service GpsUtil.

Offres et récompenses : Obtenir et afficher les réductions et récompenses disponibles pour les attractions, fournies par le service RewardsCentral.

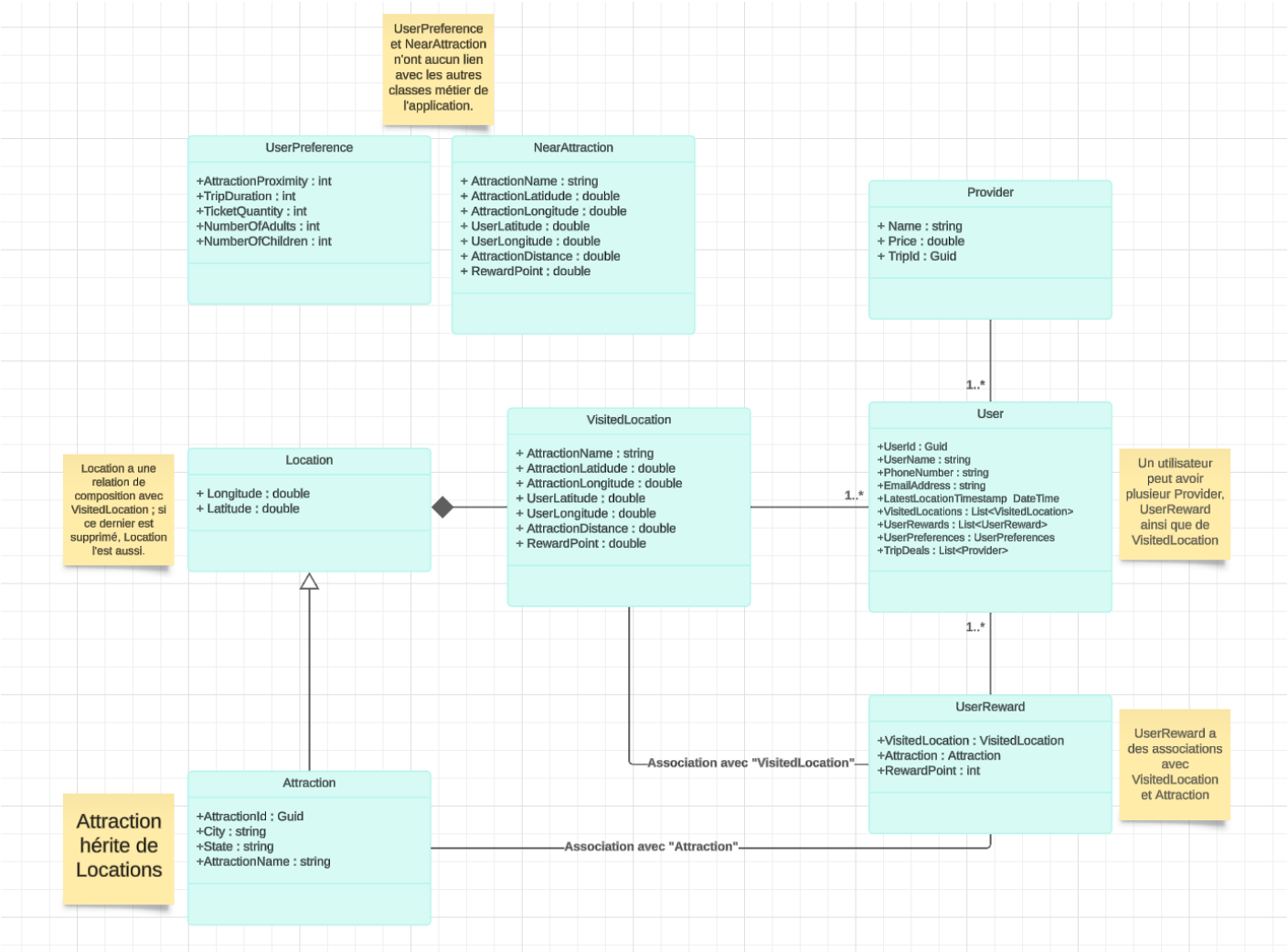
Performance et stabilité : Garantir la réactivité de l'application pour un nombre croissant d'utilisateurs, et stabiliser les tests unitaires qui échouent de manière intermittente.

3. Spécifications techniques

3.1 Schémas de structure technique



3.2 Diagramme de classes UML (Logique Métier)



3.3 Glossaire

Terme	Description
User	Représente un utilisateur dans l'application, avec des informations personnelles (ID, nom, email) et des localisations visitées (emplacements géographiques) ainsi que des récompenses collectées pour ces visites.
VisitedLocation	Objet qui enregistre un lieu visité par l'utilisateur, comprenant les coordonnées géographiques (latitude, longitude) et l'heure de la visite.
Attraction	Lieu d'attraction touristique avec des informations sur son nom, sa localisation et ses coordonnées géographiques.
RewardPoints	Points de récompense attribués à un utilisateur en fonction de ses visites d'attractions. Ces points peuvent être utilisés pour obtenir des réductions ou des offres spéciales sur des voyages.
GpsUtil	Service externe qui fournit des données de localisation utilisateur et

Terme	Description
	des attractions à proximité. Il utilise des méthodes pour obtenir l'emplacement actuel d'un utilisateur et une liste d'attractions disponibles.
RewardCentral	Service externe qui calcule les points de récompense associés à une attraction spécifique pour un utilisateur donné, basé sur son ID.
TripPricer	Service qui génère des offres de voyage en fonction de l'attraction visitée, du nombre d'adultes et d'enfants, de la durée du séjour, et des points de récompense disponibles.
Tracker	Composant qui suit l'activité des utilisateurs dans l'application, notamment leurs déplacements entre les attractions. Il peut également arrêter le suivi des utilisateurs lors de la fermeture de l'application.
Provider	Fournisseur d'offres de voyage, qui propose des prix pour un séjour en fonction des paramètres comme les récompenses, le nombre de personnes et la durée du séjour.
UserReward	Représente une récompense donnée à un utilisateur pour une attraction visitée, avec des détails sur l'attraction, le lieu visité et le nombre de points de récompense attribués.
NearAttraction	Représente une attraction proche de l'emplacement actuel d'un utilisateur, avec des informations sur la distance, le nom de l'attraction et les points de récompense disponibles.
Locations	Structure de données qui représente les coordonnées géographiques (latitude et longitude) d'un lieu.
ProximityBuffer	Distance (en mètres) autour de laquelle l'application considère qu'un utilisateur est proche d'une attraction et peut gagner des récompenses.
UserPreferences	La classe UserPreferences permet de définir les préférences d'un utilisateur pour ses voyages et ses visites d'attractions.

GitHub Actions	Plateforme d'automatisation intégrée à GitHub qui permet de créer des workflows pour automatiser des tâches comme les tests, le déploiement ou la gestion de versions.
Workflow	Fichier de configuration au format YAML qui définit une suite d'étapes automatisées (comme "build", "test", "deploy"). Ces fichiers se trouvent dans `.github/workflows/`.
Jobs	Ensemble de tâches exécutées dans un workflow. Chaque job s'exécute dans une instance isolée (machine virtuelle ou conteneur). Les jobs peuvent s'exécuter en parallèle.
Actions	Tâches réutilisables dans un workflow. Une action peut être créée par GitHub ou la communauté (ex. : "actions/checkout" pour récupérer le code).
Artifacts	Fichiers générés par un workflow (ex. : résultats de test, binaires) qui peuvent être stockés et partagés entre jobs.

3.4 Solutions techniques

La méthode *GetNearByAttractions* a été refactorisée pour retourner une liste enrichie des attractions les plus proches sous forme d'objets NearAttraction, contenant des informations détaillées telles que la distance, les coordonnées de l'utilisateur et de l'attraction, ainsi que les points de récompense associés. Les attractions sont d'abord triées par distance via un dictionnaire, puis transformées de manière asynchrone grâce à une méthode de mapping.

Pour répondre aux besoins de performance, l'approche asynchrone avec Task.WhenAll a été adoptée, notamment dans la classe RewardCentral ainsi que pour exécuter les tests de performance plus rapidement. Plusieurs méthodes ont été rendues asynchrones avec async/await, notamment celles appelant CalculateReward et TrackUserLocation. De plus, une pipeline d'intégration continue utilisant GitHub Actions garantit la qualité des modifications avant le déploiement.

Terme	Description
async	Mot-clé utilisé pour définir une méthode asynchrone. Une méthode marquée avec `async` peut contenir des opérations asynchrones (comme `await`) et permet d'éviter le blocage du thread principal.
await	Mot-clé utilisé dans une méthode `async` pour indiquer qu'une tâche doit être attendue avant de continuer l'exécution. Il permet de libérer le thread principal pendant l'exécution de l'opération asynchrone.
Task	Représente une opération asynchrone. `Task` est un type qui contient des informations sur l'état de l'opération asynchrone (en cours, terminée, etc.). Lorsqu'une méthode renvoie `Task`, elle s'exécute de manière non bloquante.
Task.Run()	Utilisé pour exécuter une tâche de manière asynchrone dans un thread séparé. Par exemple, une tâche longue ou un calcul intensif peut être exécuté sans bloquer le thread principal de l'application.
Task.WhenAll()	Méthode qui permet d'attendre que toutes les tâches spécifiées dans un tableau ou une collection soient terminées avant de continuer l'exécution. Elle permet de lancer plusieurs tâches en parallèle et d'attendre leur achèvement simultanément.
ConcurentBag	Une collection thread-safe fournie par .NET pour stocker des objets sans garantir un ordre particulier. Elle est particulièrement utile dans les scénarios multi-thread où plusieurs threads ajoutent ou lisent des éléments simultanément.

3.5 Autres solutions non retenues

Une alternative envisagée consistait à utiliser la classe Parallel avec Parallel.ForEach pour paralléliser les calculs dans la méthode CalculateRewards. Cependant, cette approche n'a pas été retenue, car elle s'est avérée moins performante que l'approche asynchrone avec Task.WhenAll lors des tests de performance.

3.6 Annexes

YML Intégration Continue :

```yaml
name: .NET Core Desktop Simple Workflow

on:
  push:
    branches: [ "master" ]
  pull_request:
    branches: [ "master" ]

jobs:

  build:

    strategy:
      matrix:
        configuration: [Debug, Release]

    runs-on: windows-latest

    env:
      Solution_Name: TourGuide.sln
      Test_Project_Path: TourGuideTest/TourGuideTest.csproj

    steps:
    - name: Checkout
      uses: actions/checkout@v4

    - name: Install .NET Core
      uses: actions/setup-dotnet@v4
      with:
        dotnet-version: 8.0.x

    - name: Build solution
      run: dotnet build ${{ env.Solution_Name }} --configuration ${{ matrix.configuration }}

    - name: Execute unit tests
      run: dotnet test ${{ env.Test_Project_Path }} --configuration ${{ matrix.configuration }}

    - name: Publish application
      run: dotnet publish ${{ env.Solution_Name }} --configuration ${{ matrix.configuration }} --output ./artifacts

    - name: Upload build artifacts
      uses: actions/upload-artifact@v3
      with:
        name: Application Package
        path: ./artifacts
```