Conception Logicielle Smartphone Wear Os
Rapport projet

Monument Tracking
Année: 2022/2023

UNIVERSITÉ CÔTE D'AZUR

POLYTECH NICE SOPHIA

Membres du groupe:

Noé BERNIGAUD

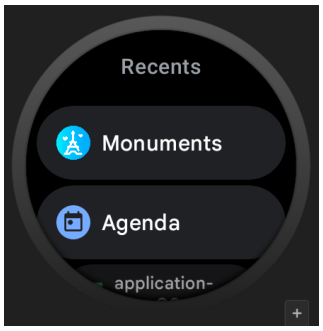
Nadine Carel AZEBAZE ZEWO

Antoine VENTURELLI

Lien github: https://github.com/AntoineVen/GPS-Tracking

Sommaire

Présentation Générale

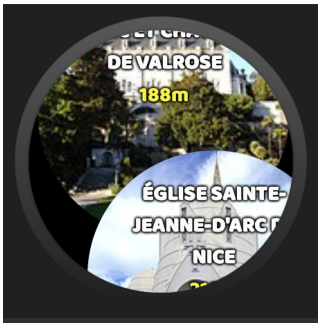


Notre projet est une application pour smartwatch permettant à l'utilisateur de localiser et afficher les monuments autour de lui grâce à sa position GPS et à la base de données Wiki Loves Monuments. C'est une application destinée aussi bien aux touristes en visite qu'aux citadins souhaitant en découvrir plus sur leur propre ville. Le choix d'une application smartwatch permet d'être guidé sans avoir à se soucier de sortir son téléphone, et permet une découverte plus naturelle.

Lors de l'ouverture de l'application, celle-ci vérifie que les autorisations de géolocalisation sont bien attribuées, puis récupère la localisation de l'utilisateur. Cette localisation est ensuite utilisée comme paramètre pour récupérer la liste des monuments environnants, grâce à la base de données fournie par l'API Wiki Loves Monuments.

Une fois la liste des monuments et de leurs caractéristiques récupérés, ceux-ci sont affichés sous forme de liste.
Le nom du monument, sa distance de l'utilisateur, ainsi que son image sont les éléments affichés à l'utilisateur. Un symbole de coche est aussi affiché si l'utilisateur a déjà par le passé visité ce monument, information stockée dans une base de données en ligne. L'utilisateur peut cliquer sur un monument afin de directement lancer la navigation vers celui-ci depuis sa position avec Google Map si cette application est installée sur l'appareil.

L'utilisateur peut défiler dans la liste des monuments environnants simplement en montant ou descendant grâce à l'écran tactile. La liste est triée par distances croissantes, permettant à l'utilisateur de voir au plus vite les monuments les plus proches.



Conditions

Fiche Technique

Comme notre application est déployée sur Smartwatch, nous dressons ici la fiche technique de la Samsung Galaxy Watch 4 afin de mettre en perspective les besoins de notre application avec les capacités du matériel.

Smartwatch Samsung Galaxy 4 :

Système: Wear OS 3.0, **Systèmes compatibles**: Android 6.0, **Fréquence processeur**: 1.18 GHz, **Mémoire vive (RAM)**: 1.5 Go, **Capacité de stockage disponible**: 7600 Mo, **Capacité de la batterie:** 361 mAh, **Taille (diagonale):** 1.4 ", **Technologie de l'écran:** Super AMOLED, **Fonction appareil photo:**Non, **Fonction enregistrement vidéo:**Non, **Webcam:**Non, **Capteurs:** Accelerometer, Barometer, Gyroscope, Geomagnetic sensor, Ambient light sensor, **Samsung BioActive sensor**: optical heart rate (PPG), electrocardiogram (ECG), bioelectrical impedance analysis sensor (BIA) Connectivity LTE (available in select models), **Bluetooth:** 5.0, Wi-Fi 802.11 a/b/g/n 2.4+5GHz, NFC, GPS/GLONASS/Beidou, Galileo.

Besoins spécifiques de l'application

Afin d'assurer le bon fonctionnement de l'application, le contexte de déploiement aura besoin de fournir:

- Une **connection GPS** mise à disposition par la montre ou le téléphone associé. Les besoins de stabilité et de précision de la localisation GPS sont faibles dans notre cas. En effet l'application n'est pas soumise à des contraintes d'urgence temporelles importantes, et étant donné l'échelle de distance avec les bâtiments (0 à 4000 m) la localisation GPS reste pertinente même dans le cas d'une imprécision d'une centaine de mètres.
- Une **connexion Internet** mise à disposition par la montre ou le téléphone associé. Cette connexion ne nécessite pas une grande stabilité ni puissance pour le fonctionnement minimum de l'application, mais une bonne passante relativement conséquente sera consommée occasionnellement lors de l'import des images des monuments. Tout besoin de connexion internet est uniquement dédié à la récupération d'informations stockées ailleurs, de façon asynchrone, et peut être délayé sans problème de fonctionnement.
- Dans le cas où la montre connectée utilise la connexion GPS ou internet d'un téléphone, une **connexion Bluetooth** sera nécessaire au pairage de la montre avec celui-ci. Cependant, l'application peut aussi être utilisée en mode standalone si la montre possède une connectique GPS et internet, et dans ce cas ne nécessitera pas de connexion Bluetooth.

Contextes d'utilisation

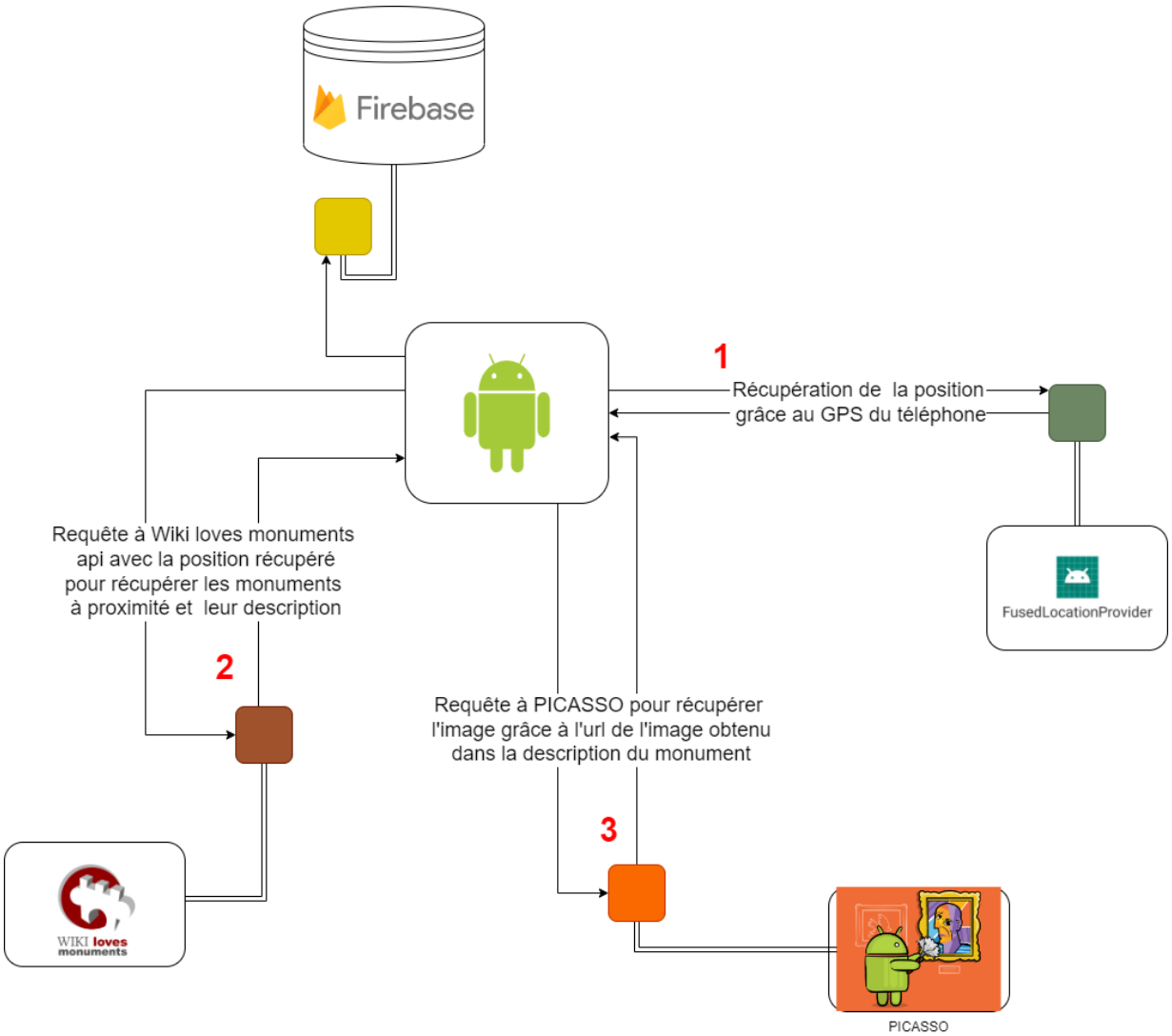
- **Extérieur en ville**: Cas usuel d'utilisation, cas typique de la visite d'une ville. La connexion internet est habituellement excellente, cependant le GPS peut

montrer quelques imprécisions à cause des intéractions des infrastructure avec le signal. Cependant, celui-ci reste suffisamment précis pour une utilisation excellente de l'application.

- **Intérieur:** Connection internet pouvant être ralentie, sauf si l'utilisateur peut se connecter au réseau local. Notre méthode de localisation GPS pouvant croiser des informations liées au réseau environnant, la position reste généralement précise. Cependant si la localisation doit se reposer uniquement sur les informations GPS, la précision sera grandement détériorée.
- **Zones isolées:** Dans le cadre d'une utilisation dans une zone dépourvue de connexion internet, comme une région reculée (ex: montagne) ou un bâtiment avec brouilleur d'ondes, l'application ne pourra pas fonctionner normalement et ces cas entreront dans la gestion d'erreurs.

Architecture et Implémentation Logique

Architecture du projet



Description des services utilisés

1 - Wiki Loves Monuments API

Wiki Loves Monuments est à l'origine un projet incitant chacun, par le biais d'un concours, à prendre des photographies de monuments et les placer sous licence libre pour permettre au plus grand nombre d'y accéder via Internet. Les données récoltées à travers le monde par ce portail sont mises à disposition grâce à une API entièrement gratuite et associant chaque bâtiment avec de nombreuses informations.
Exemple de requête:
https://heritage.toolforge.org/api/api.php?action=search&radius=3000&props=country|id|name|municipality|lat|lon|image&limit=100000&coord=43.714283333333334%2C7.264885
Cette requête renvoie un objet XML contenant la liste des monuments dans un rayon de 4km autour des coordonnées passées en paramètre. Le nombre de monuments dans la réponse est limité à 100000. Pour chaque monument, son pays, numéro d'identifiant, nom, ville, coordonnées, et nom d'image sont récupérés.

2 - Picasso

Picasso est une bibliothèque d'images pour Android. Il est créé et maintenu par Square, et prend en charge le chargement et le traitement des images. Il simplifie le processus d'affichage des images à partir d'emplacements externes. La bibliothèque gère chaque étape du processus, de la requête HTTP initiale à la mise en cache de l'image.

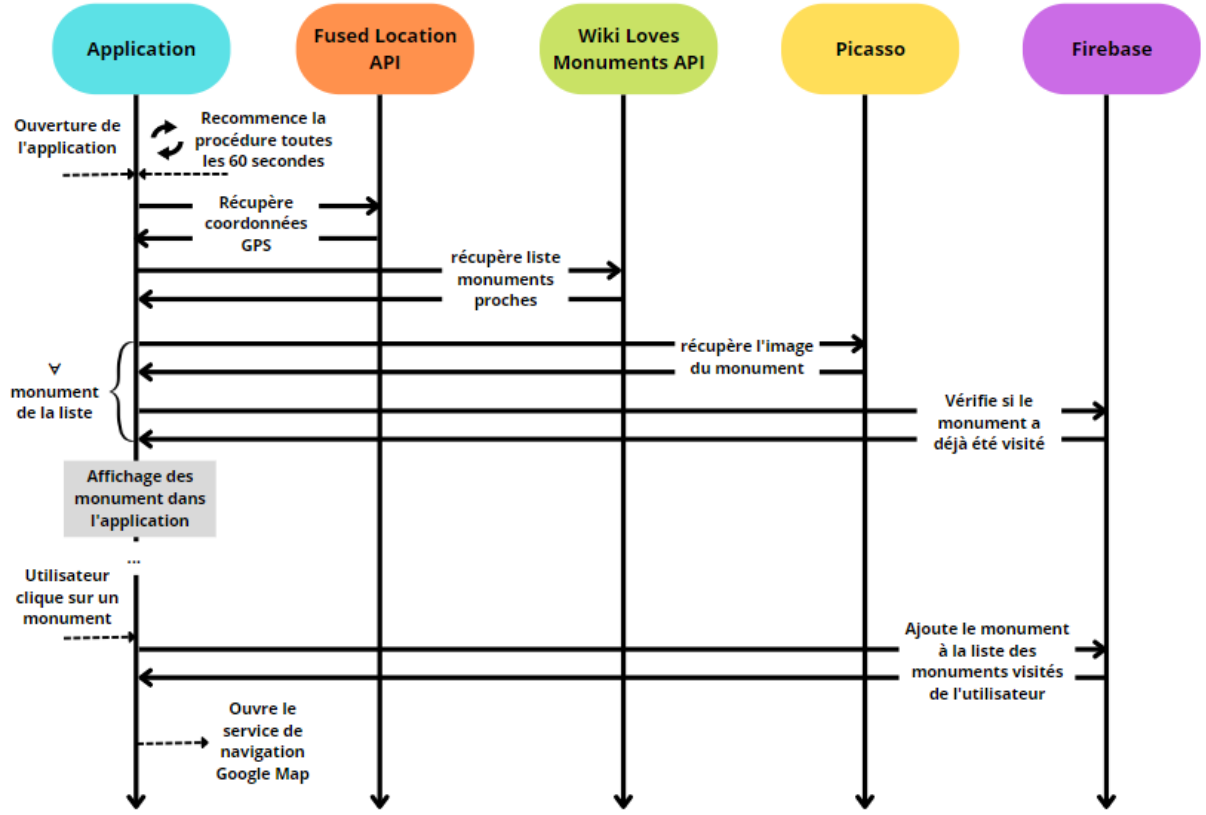
3 - FusedLocationProviderAPI

Développée par Google pour le développement Android, la **Fused Location Provider API** sert à estimer précisément la localisation de l'appareil en utilisant une combinaison de données GPS, Wi-Fi, et capteurs internes. Son atout principal est de gérer la combinaison de ses informations en se basant sur les paramètres passés et enlève au développeur le besoin de gérer lui-même cette combinaison.
Contrairement aux autres services mentionnés dans cette section, ce service de requiert pas de connection quelconque puisqu'il est embarqué dans l'appareil.

4 - Firebase

Firebase est un ensemble de services d'hébergement pour n'importe quel type d'application (Android, iOS, Javascript, Node.js, Java, Unity, PHP, C++ ...). Il propose d'héberger en NoSQL et en temps réel des bases de données, du contenu, de l'authentification sociale (Google, Facebook, Twitter et Github), et des notifications, ou encore des services, tel que par exemple un serveur de communication temps réel.

Diagramme de Séquence



Lors de l'ouverture de l'application, celle-ci commence par vérifier que les autorisations nécessaires afin d'obtenir la localisation de l'utilisateur sont bien attribuées, au cas contraire elle les lui demande. L'application vérifie aussi la disponibilité d'un dispositif GPS.
Gestion d'erreurs: Dans le cas d'absence de dispositif GPS, l'application lance une animation signalant le problème à l'utilisateur puis se ferme.

L'application lance ensuite une requête à la **Fused Location API**. Cette requête à pour effet de mettre à jour la position de l'utilisateur toutes les 60 secondes, et à chaque fois qu'une nouvelle location est déterminée met à jour la liste des monuments. Une vibration de la montre a lieu si de nouveaux monuments sont apparus dans la liste.
Pour mettre à jour la liste des monuments, la première étape est de récupérer la liste des monuments proches et les informations sur ceux-ci grâce à l'**API Wiki Loves Monuments.** Grâce à Retrofit, la réponse XML est directement interprétée en une liste d'objets Java. Cette liste est alors envoyée au Wearable Recycler View, qui gère l'affichage des monuments sur la montre.
Gestion d'erreurs: Dans le cas où toutes les requêtes ont fonctionné mais aucun monuments n'a été trouvé, le message "`No Monument Found`" est affiché à l'utilisateur. Si l'API Wiki Loves Monument renvoie un code d'erreur, le message "`Error requesting database`" est affiché. Si l'on ne reçoit aucune réponse de l'API, c'est le message "`No response from database, please check your internet connection`" qui est affiché. Dans tous les cas, l'application continue son fonctionnement et tente à nouveau d'obtenir la liste des monuments 60 secondes plus tard.

Pour chaque monument ajouté dans le **Wearable Recycler View**, l'adaptateur chargé de la gestion des éléments affichés dans celui-ci va alors obtenir l'image du monument en passant l'url, obtenu grâce aux informations renvoyées par Wiki Loves Monument précédemment, à **Picasso**. L'adaptateur est aussi associé à une instance du **Data Access Object**, objet gérant les accès en lecture et écriture à la Database **Firebase**. Grâce à celui-ci, l'adaptateur vérifie si le monument a déjà été visité par l'utilisateur par le passé, et en fonction du résultat affiché ou non le symbole "✓" sur le monument.
L'adaptateur se charge aussi alors d'attribuer à l'élément un onClickListener, permettant l'interaction de l'utilisateur avec l'élément et ainsi d'ouvrir la navigation Google Map et d'ajouter le monument à la liste des monuments visités par l'utilisateur sur Firebase.
Gestion d'erreurs: Si la récupération de l'image par Picasso échoue, une image noire par défaut est attribuée à la place de l'image du monument. Si l'utilisateur clique sur un monument afin d'ouvrir la navigation Google Map vers celui-ci mais que Google Map n'est pas installé sur l'appareil, le message "`Download Google Map to use navigation`" s'affiche à la place.

Conception et Réalisation

Langages et logiciels utilisés

Nous avons pour le développement de l'application d'utiliser le langage Java car celui-ci est compatible avec le développement Android, met à disposition de nombreuses librairies, et est un langage où nous pouvions témoigner d'une forte expertise technique.
Le logiciel de développement utilisé était Android Studio, qui facilite grandement le développement et le test d'applications en Android natif. En effet, celui-ci propose une organisation de fichiers appropriée ainsi qu'un émulateur très complet permettant de vérifier le fonctionnement correct de l'application dans de nombreuses configurations.

Organisation du Travail

Pour organiser le développement de l'application étape par étape, en suivant les bonnes pratiques, et en gardant une certaine flexibilité nous avons suivi la méthode agile. L'avancée du projet c'est faite selon le programme suivant:

- **Sprint 0** - Définition de l'architecture logicielle et choix des technologies.
- **Sprint 1** - L'application envoie une requête avec des coordonnées fixées et affiche les noms de monuments obtenus.
- **Sprint 2** - La requête vers Wiki Loves Monument utilise les coordonnées de l'utilisateur, se fait à intervalles réguliers, et on obtient un vibration si de nouveaux monuments sont présents.
- **Sprint 3** - Les images des monuments ainsi que leur distance de l'utilisateur sont affichées à l'utilisateur. Les noms sont parés afin d'optimiser l'affichage. Lorsque l'utilisateur clique sur un élément, l'image de celui-ci s'affiche en grand.
- **Sprint 4** - L'affichage de la liste des monuments est redéfinie visuellement afin de permettre un affiche des images en plein écran sans avoir à cliquer.

Lorsqu'un utilisateur clique sur un élément, celui-ci est redirigé vers la navigation Google Map.
- **Sprint 5** - Une base de données Firebase est associée à l'application, permettant de conserver la liste des monuments visités par chaque utilisateur. Les monuments déjà visités sont affichés avec un indicateur visuel.
- **Sprint 6** - Les erreurs possible liée à l'utilisation de l'application sont géré de façon approprié afin d'éviter tout crash et de donner des informations claires à l'utilisateur.

Les différentes tâches à effectuer lors d'un sprint étaient chaque semaine attribuées, et nous nous concertons régulièrement afin de communiquer sur l'avancée des différentes fonctionnalités, ajuster notre direction en fonction de nos résultats et avancés, ainsi que partager sur les points bloquants du développement.

Utilisation et Installation

Pour utiliser l'application, il est nécessaire d'avoir une montre connecté possédant un GPS et une connexion internet si l'usage est fait sans appareillage avec un téléphone. Dans le cas contraire, une connexion Bluetooth avec un téléphone possédant la ou les connexions manquantes à la montre est nécessaire.

L'application n'est actuellement pas publiée sur le Play Store compte tenu des restrictions de temps sur le projet. Il est en conséquence nécessaire de "Sideload" le fichier APK depuis un ordinateur vers la montre connectée afin de télécharger l'application.

L'APK de l'application est disponible à la racine du dossier Github de l'application: https://github.com/AntoineVen/GPS-Tracking

Pour déployer une application Android (APK) depuis votre ordinateur vers votre montre connectée Wear OS, suivez ces étapes :
1. Assurez-vous que votre montre connectée est connectée à votre ordinateur via Bluetooth.
2. Sur votre ordinateur, ouvrez l'outil de ligne de commande Android Debug Bridge (ADB). Vous pouvez trouver cet outil dans le dossier Android SDK de votre ordinateur.
3. Dans l'outil ADB, exécutez la commande adb devices pour afficher une liste des périphériques connectés. Vous devriez voir votre montre connectée dans cette liste.
4. Dans l'outil ADB, exécutez la commande adb install suivie du chemin vers le fichier APK que vous souhaitez installer sur votre montre connectée. Par exemple : adb install /chemin/vers/l'application.apk
5. Attendre que le fichier APK soit transféré vers votre montre connectée et installé.
6. Une fois l'installation terminée, vous devriez voir un message indiquant que l'application a été installée avec succès.
7. Sur votre montre connectée, accédez à la liste des applications installées et appuyez sur l'application que vous venez d'installer pour lancer l'application.

Il se peut que vous ayez besoin d'activer le mode développeur sur votre montre connectée pour pouvoir installer des applications à l'aide de l'outil ADB. Consultez la

documentation de votre montre connectée pour obtenir des instructions sur la façon de procéder.

Perspectives et Améliorations

Le résultat de ce projet est une application pleinement fonctionnelle, accomplissant sa mission. Nous avons pu faire le lien entre une application Android pour Smartwatch avec une API ainsi qu'avec une base de données, et utiliser le capteur GPS de la montre. L'application montre un réel intérêt pratique à être déployé sur montre plutôt que sur smartphone, et ne montre aucun défaut majeur d'utilisation.

Cependant, il est bien sûr possible d'encore améliorer certaines fonctionnalités. Nous conclurons donc par présenter quelques perspectives futures d'optimisation.

Tout d'abord, un déploiement sur le **Play Store** serait important afin de rendre l'application disponible facilement au public.

Il serait aussi approprié d'implémenter un **système de cache**, voire d'une fonctionnalité de **pré-téléchargement** de la liste des monuments. Cela permettrait de récupérer les données des monuments en avance, et ainsi de ne pas avoir besoin de faire des requêtes internet une fois sur place, et donc d'économiser la batterie, améliorer la performance, et gérer les situations sans connexion.

Un autre ajout important serait de donner **plus de contrôle à l'utilisateur** en regard de ses données. Actuellement la liste des monuments cliqué par l'utilisateur est stockée et celui-ci ne peut pas supprimer ses données. Un contrôle plus fin serait aussi intéressant pour les fonctionnalités de recherche, permettant à l'utilisateur de choisir par exemple de ne pas afficher les monuments déjà visités, ou de modifier le rayon de recherche.

Finalement, une problématique actuelle de l'application est de parfois noyer l'information avec un grand nombre de monuments, surtout dans le cas des grandes villes. Il pourrait donc être pertinent d'intégrer une **intelligence artificielle** capable de faire des suggestions à l'utilisateur de monuments à visiter en se basant sur les préférences de celui-ci.