EasyTravel

MADE BY SOFIANE AZARKAN

Rapport de projet final

PROJET MOBILE - EASYTRAVEL

Sofiane AZARKAN | Développement Mobile | 25/05/2022

Table des matières

Introduction 2
Sources et références 2
Description des technologies utilisées 3
1. openweathermap api 3
1.1. Requête pour récupérer la météo actuelle 3
1.2. Requête pour récupérer les informations sur les prévisions météo 4
2. Librairie Retrofit 5
3. ANDROID STUDIO 5
4. Sharedpreferences 5
5. Alarmmanager 6
6. Cardview 6
Fonctionnalités 6
1.activite meteo 7
1.3. Qu'est-ce que les fragments et pourquoi les utiliser ? 7
1.4. Fragment 1 : Météo actuelle 7
1.5. Fragment 2 : Prévisions météos 8
1.6. Fragment 3 : Les villes populaires 8
2. Activité Find a destination 9
3. Activité calendrier 10
Analyse 10
Limitations et développement futur 11
Conclusion 12

Introduction

Dans le cadre du cours de Développement Mobile, il m'a été demandé de développer une application Android devant répondre à certaines contraintes et présenter un ensemble minimum de fonctionnalités. Voici quelques exemples de ce qui doit être implémenté : l'application doit être composée d'au minimum 3 activités et présenter un système de navigation entre les activités complexe mais logique. Celle-ci doit également supporter au minimum le format téléphone et tablette et réagir adéquatement au changement d'orientation de l'appareil. Elle doit également générer des notifications à l'utilisateur. Sans oublier qu'elle doit d'une manière ou d'une autre accéder et faire usage de données accessibles en ligne (Exemple JSON API).

Au départ, je voulais faire une simple application météo. Mais pour finir, j'ai décidé, avec l'aide de mon chef de projet, de faire une application de permettant à un utilisateur de visualiser la météo partout dans le monde afin de savoir où aller en vacances.

Dans ce rapport, je vais commencer par introduire les différentes technologies utilisées et motiver leur utilisation. Deuxièmement, je vais décrire les fonctionnalités offertes par mon application d'un point de vue fonctionnel mais aussi d'un point de vue technique. Ensuite, je vais expliquer la structure de mon implémentation en utilisant des outils d'analyses. Après cela, je vais expliquer les limites de mon application dans le cas d'un développement futur.

Enfin, je vais conclure en reprenant ce qui a été vu de manière générale dans ce rapport, ce que j'ai réussi à faire ou non durant le projet et les apprentissages que j'en ai tirés.

Sources et références

Avant de commencer à expliquer les fonctionnalités de mon application je me dois de référencer les vidéos YouTube qui m'ont aidé dans l'implémentation de mon application.

Après beaucoup de recherches, j'avais finalement trouvé un tutoriel sur YouTube où le développeur montre comment avoir un bon UX design avec Android Studio tout en utilisant la même API que moi. Je me suis donc inspiré de ses vidéos pour l'affichage de mon application météo et pour une bonne utilisation de l'API.

- ➔ Lien de la vidéo 1 : https://www.youtube.com/watch?v=awYSrhUZQLo
- ➔ Lien de la vidéo 2 : https://www.youtube.com/watch?v=v7fW262s7yI
- ➔ Lien de la vidéo 3 : https://www.youtube.com/watch?v=z7ArkouYLTI

En effet, avec la première version que j'avais développée, j'avais appris à me servir de l'API OpenWeatherMap mais malheureusement l'affichage était médiocre et n'était vraiment pas bien travaillé. Mais avec ces vidéos, j'ai appris beaucoup de choses telles que l'existence des « CardView ». J'ai également appris à utiliser les Fragments,

favoriser l'utilisation de classes Java dans mon projet et séparer les fichiers sous formes de dossier afin de m'y retrouver plus facilement et de bien structurer mon code. C'est également grâce à ces vidéos que j'ai pu implémenter la librairie Retrofit dans mon projet.

Ce qu'il faut retenir, c'est que ces 3 vidéos m'ont fourni une base de développement solide sur laquelle je pouvais me baser pour continuer et développer l'application comme je le souhaitais.

Description des technologies utilisées

1. OPENWEATHERMAP API

En ce qui concerne l'API, j'ai décidé de choisir OpenWeatherMap (https://openweathermap.org/api). J'ai effectué mes recherches et j'ai vu qu'il y a pas mal d'informations et de gens qui utilisent ce « Weather API ». De plus, d'après mes recherches, c'est l'un des meilleurs « Weather API » qui n'est pas payant ou en tout cas qui possède une version gratuite assez complète. Sur leur site, il est clairement indiqué qu'il est possible d'avoir accès aux données météorologiques pour n'importe quel endroit sur Terre (dont plus de 200 000 villes) et en plus de cela, les données sont disponibles sous formes de fichier JSON, XML ou encore en format HTML.

Les requêtes ont été effectuées grâce à la libraire Retrofit. (voir point 2).

1.1. Requête pour récupérer la météo actuelle

Voici l'url permettant de récupérer les informations météorologiques actuelles dans une certaine ville en format JSON :
« https://api.openweathermap.org/data/2.5/weather?q=Brussels&appid=3b760f0288a791f58499b2b689e19d44 ».

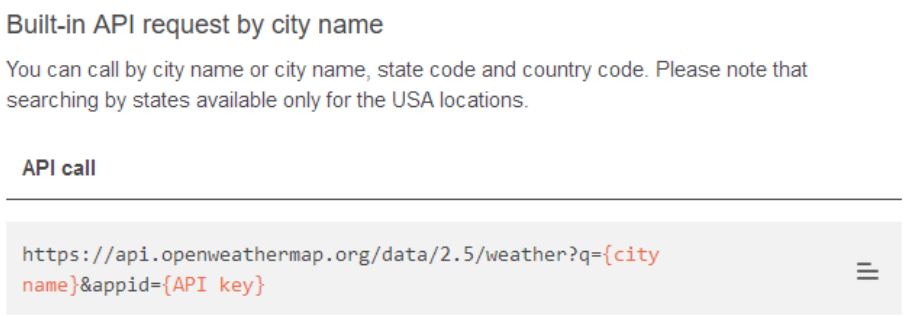
En jaune, on peut voir qu'il y a écrit « weather ». Comme OpenWeatherMap possède plusieurs fonctionnalités, ce paramètre change en fonction de ce que l'on cherche. Par exemple, si l'on cherche les prévisions météorologiques, alors le « weather » va devenir « forecast ». Mais ici, on cherche la météo actuelle, et il est indiqué dans la documentation de OpenWeatherMap que pour obtenir la météo actuelle il faut écrire « weather ».

En bleu, on peut voir que le paramètre « q » correspond au nom de la ville dont on souhaite avoir les informations actuelles. Dans ce cas-ci, c'est Bruxelles.

En vert, le paramètre « appid » correspond à la clé API que je possède. Pour en avoir une, il faut obligatoirement créer un compte sur OpenWeatherMap.

La documentation pour savoir comment faire une requête API pour obtenir la météo actuelle se trouve sur le site à cette adresse : https://openweathermap.org/current.

Voici la documentation de OpenWeatherMap qui nous montre comment rechercher les informations météorologiques actuelles en connaissant le nom de la ville :



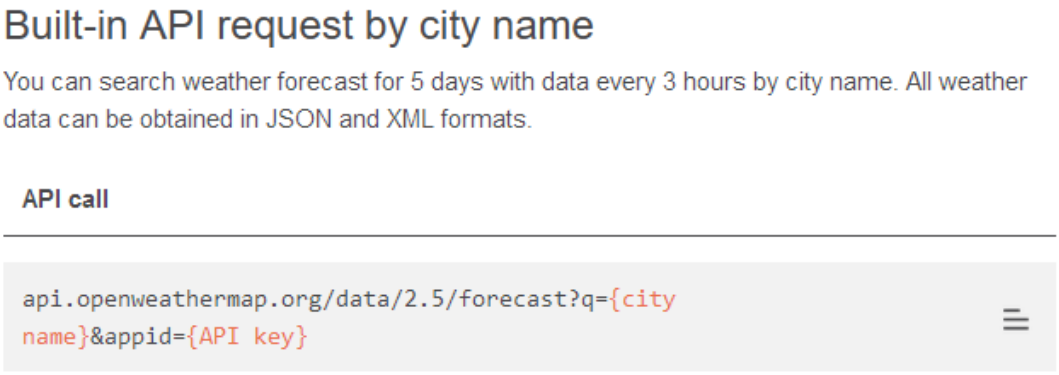
1.2. Requête pour récupérer les informations sur les prévisions météorologiques

Voici l'url permettant de récupérer les prévisions météorologiques dans une certaine ville en format JSON :

« http://api.openweathermap.org/data/2.5/forecast?q=Brussels&appid=3b760f0288a791f58499b2b689e19d44 ».

La documentation permettant de faire une requête API pour les prévisions météorologiques se trouve à cette adresse : https://openweathermap.org/forecast5.

Voici un exemple de OpenWeatherMap pour les prévisions météorologiques :



On peut voir que ce n'est pas très différent de la requête permettant d'obtenir les informations météorologiques actuelles d'une certaine ville.

2. LIBRAIRIE RETROFIT

Il faut savoir que mon application EasyTravel consomme des API REST JSON pour récupérer les données nécessaires à son fonctionnement. C'est grâce à Retrofit qui permet d'implémenter des appels API, que mon application peut récupérer les données dont elle a besoin.

Retrofit est un client REST développé par Jake Wharton ainsi que la compagnie Square. Cette librairie est basée elle-même sur le client REST OKHttp et permet d'implémenter plus facilement et rapidement des requêtes réseau sur Android (Java ou Kotlin).

Retrofit m'évite ainsi d'installer manuellement toutes les parties nécessaires à l'exécution d'une requête (exemple : la gestion des réponses JSON, ...). Cela fait qu'il y a un gain de temps considérable et un code plus clair pour des performances équivalentes.

Source des informations sur la librairie Retrofit :
https://www.axopen.com/blog/2021/01/retrofit-projet-android-kotlin/

3. ANDROID STUDIO

Le développement de mon application a été entièrement réalisé sur l'environnement de développement spécialisé : **Android Studio**. C'est un environnement de développement très puissant et complet mais il est lourd à l'exécution. Il est possible de créer et d'y développer des projets en Java ou Kotlin mais mon application a été codée en Java de manière intégrale.

Un des avantages d'Android Studio est qu'il est possible de tester son application directement grâce à un émulateur qui émule un appareil Android au choix.

L'application a été testé sur un périphérique virtuel : le Pixel 3 API 27 et avec le système d'exploitation Oreo 8.1.

4. SHAREDPREFERENCES

Avec Android Studio, il est possible de stocker des valeurs et leurs clés afin de les réutiliser même si l'on redémarre l'application. Pour se faire, il faut utiliser « SharedPreferences ».

Dans le cas de cette application, j'utilise SharedPreferences afin d'enregistrer le premier jour de vacances de l'utilisateur. Ainsi, lorsqu'il relancera l'application, la date qu'il avait entrée sera bien gardée et il pourra la changer quand il le veut.

Note : L'activité permettant de planifier son jour de vacances sera expliquée en détail plus bas dans le rapport.

5. ALARMMANAGER

Pour planifier une tâche, j'utilise AlarmManager. Cette classe donne accès aux services d'alarme du système (du périphérique android). Cela permet donc de planifier l'exécution de l'application à un moment donné dans le futur.

Je m'en sers pour envoyer une notification à l'utilisateur lorsque c'est son premier jour de vacances.

6. CARDVIEW

Tout d'abord, il faut savoir que dans mon application, j'affiche les données météorologiques sous forme de petites cartes contenant toutes les informations que je souhaite afficher, comme par exemple, la température, la description de la météo, l'icône correspondant à la météo, etc.

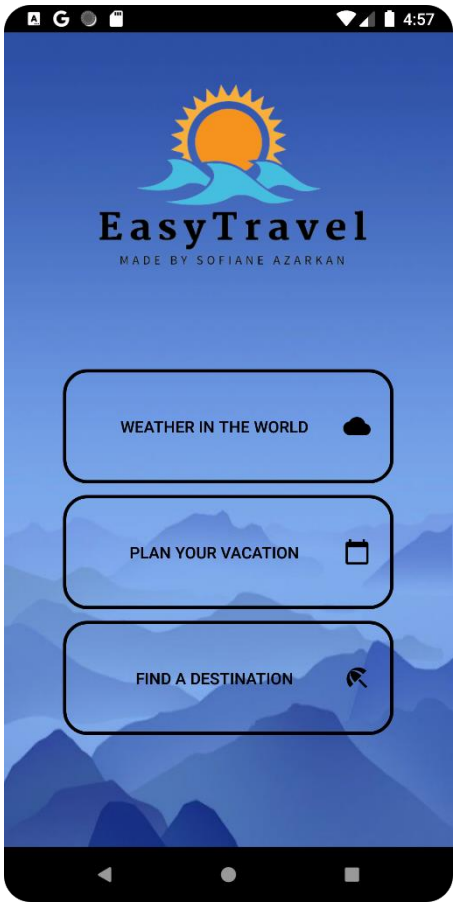
Pour mettre en place ces petites cartes, j'ai utilisé CardView. CardView est un widget dans Android qui peut être utilisé pour afficher n'importe quel type de données. L'utilisation principale de CardView est qu'elle permet de donner une impression et un rendu agréable à la présentation de l'interface utilisateur. CardView peut être utilisé pour créer des éléments dans un RecyclerView, par exemple. Ainsi, on peut créer une seule CardView et la réutiliser autant de fois que l'on veut. Par exemple, on peut afficher une liste de CardView où chaque CardView contiendra des données différentes mais toujours la même structure d'affichage.

Fonctionnalités

Lorsque l'on lance l'application, on arrive premièrement sur l'activité « MainActivity » qui contient la page d'accueil de l'application.

On peut voir qu'il y a 3 différents boutons qui permettent chacun d'accéder à une activité différente.

En cliquant sur le bouton « Weather in the world », on arrive dans l'activité météo. Si on clique sur « Plan your vacation » on arrive sur l'activité « CalendarActivity » et si l'on clique sur le bouton « Find a destination », on se retrouve alors dans l'activité « FindDestinationActivity ».



1.ACTIVITE METEO

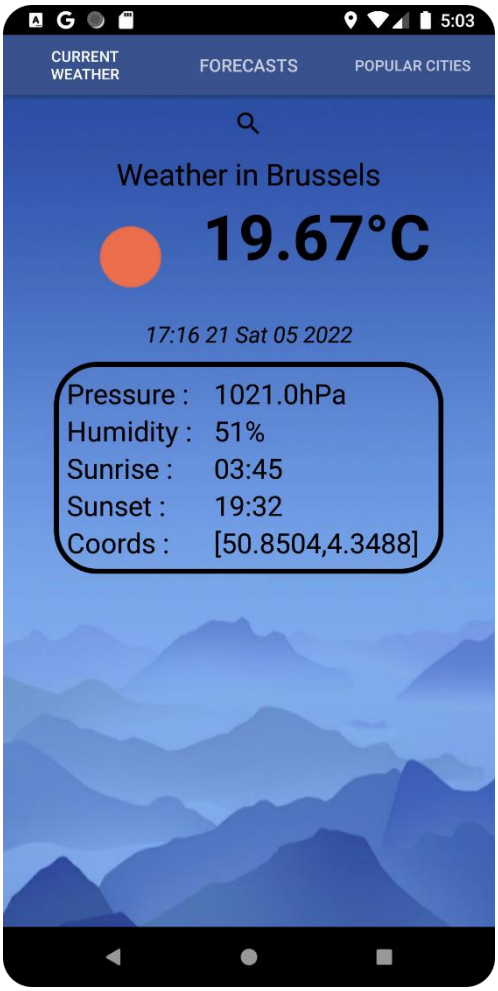
Dans l'application, il y a une activité « Weather in the world ». Cette activité est segmentée en trois fragments. Le premier fragment permet de consulter la météo actuelle dans une certaine ville. Le deuxième permet de consulter les prévisions météos dans une certaine ville et le troisième fragment permet de consulter la météo actuelle dans les villes dites « populaires ».

1.3. Qu'est-ce que les fragments et pourquoi les utiliser ?

« Les fragments permettent de scinder les activités en composants encapsulés et réutilisables qui possèdent leur propre cycle de vie et leur propre interface graphique. » (**source** : *Android Fragments, M. Riggio*). En utilisant des fragments, je pouvais donc diviser une activité en plusieurs modules. Et c'est ce que j'ai fait avec l'activité « WeatherActivity ». Cette activité est divisée en 3 fragments.

1.4. Fragment 1 : Météo actuelle

Dans ce fragment, il est possible de consulter la météo actuelle dans une ville choisie. Lorsque que l'on arrive dedans, on peut directement voir la météo actuelle à Bruxelles. J'ai défini Bruxelles comme étant la ville par défaut. Dans les informations météorologiques affichées dans ce fragment, il est possible de voir la température actuelle, la pression atmosphérique, l'humidité, l'heure du lever du soleil et l'heure du coucher du soleil, et pour finir, les coordonnées géographiques de la ville.



Pour choisir la ville dont on souhaite afficher les informations météorologiques actuelles, il suffit de cliquer sur la petite loupe au-dessus. En cliquant sur cette loupe, une barre de recherche sera ouverte et il sera possible pour l'utilisateur d'introduire la ville qu'il souhaite.

D'un point de vue technique, pour afficher les données de la ville qui nous intéresse, je fais simplement en sorte de récupérer le texte introduit par l'utilisateur lorsqu'il clique sur « Rechercher » et je passe la ville en paramètre dans ma requête API. Ainsi, les informations de la ville concernée seront affichées.

Il faut savoir que OpenWeatherMap fournit plusieurs types d'API et pour ce fragment j'ai utilisé l'API « Current Weather Data » de OpenWeatherMap.

1.5. Fragment 2 : Prévisions météos

Dans ce fragment, il est possible de consulter les prévisions météorologiques de la ville recherchée.

Tout comme pour le fragment précédent, la ville par défaut est Bruxelles et il est également possible de rechercher la ville que l'on souhaite en cliquant sur la petite loupe.

Au niveau de l'affichage, on peut voir que CardView est utilisé pour afficher les prévisions météorologiques. On obtient la température à un jour donné et une heure donnée, une icône et une description qui correspond à l'icône. (Par exemple, si l'icône est un soleil, la description sera « clear sky » pour « ciel dégagé »). Les prévisions météo affichées concernent les 5 prochains jours, toutes les 3 heures.

Pour ce fragment j'ai utilisé l'API « 5 Day / 3 Hour Forecast » de OpenWeatherMap.

1.6. Fragment 3 : Les villes populaires

Ce fragment affiche la météo actuelle dans des villes dites « populaires ». Ainsi, si l'utilisateur ne sait pas vers quelle destination se diriger, il peut se rendre sur ce fragment pour avoir une vue sur les villes les plus connues dans le monde.

D'un point de vue technique, les villes ont simplement été hard-codées dans un tableau de String. Et pour afficher leur température actuelle, je parcours le tableau de villes et je fais une requête API pour chaque ville afin de recevoir la météo actuelle et de l'afficher.

Pour ce fragment, j'ai donc, tout comme pour le Fragment 1, utilisé l'API « Current Weather Data » de OpenWeatherMap puisque c'est la météo actuelle qui m'intéresse dans ce cas-ci.

2. ACTIVITE FIND A DESTINATION

Cette activité a pour but d'aider l'utilisateur dans le cas où il ne sait pas où aller pour ses vacances. Celle-ci ressemble au Fragment « Les villes populaires » expliqué ci-dessus mais ce n'est pas tout à fait la même chose. En effet, ici ce qui est affiché, ce n'est pas la température actuelle dans chaque ville populaire mais plutôt la moyenne des prévisions météorologiques pour les 5 prochains jours de chaque ville populaire.

Pour mettre en place une moyenne ayant du sens et cohérente, je ne pouvais pas me contenter de faire une moyenne des températures sans rien faire d'autre. Je devais également trouver un moyen de faire la moyenne des icônes.

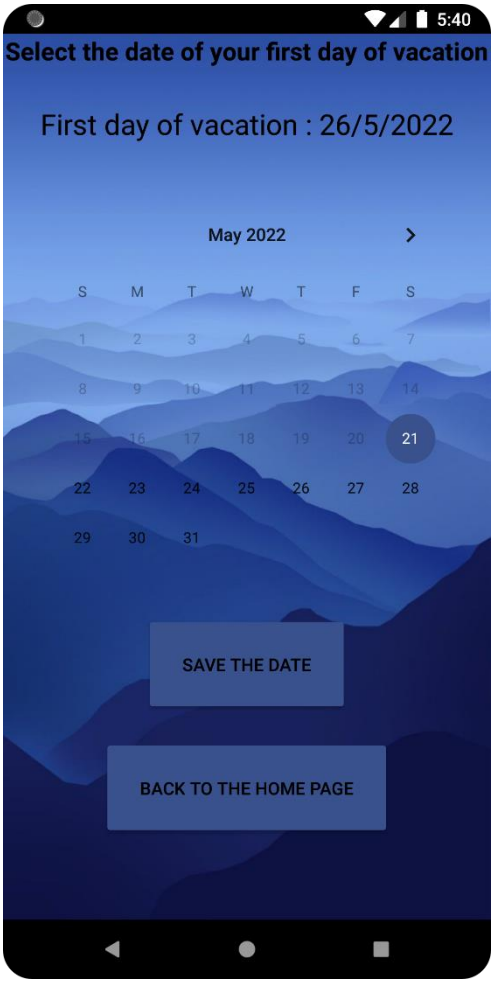
Pour faire la moyenne des icônes, j'ai fait en sorte de stocker toutes les icônes reçus par ville dans une liste. Ensuite, j'associe une valeur à chaque icône qui correspondra à son degré d'importance. Voici la valeur de chaque icône de mon application :

- Soleil : 3
- Soleil + Nuage : 2
- Nuage : 1
- Double Nuage : 1
- Pluie forte : 3
- Pluie : 2
- Orage : 3
- Neige : 4
- Brouillard : 1

Désormais il ne me reste plus qu'à récupérer l'icône la plus présente selon son degré d'importance. Donc même si Nuage apparaît 3 fois, il suffit que Neige apparaisse une fois pour que ce soit l'icône de la neige qui sera affichée.

C'est ainsi que j'ai procédé pour trouver l'icône moyenne. En ce qui concerne la moyenne de la température, il n'y avait pas de difficulté particulière, j'ai simplement stocké toutes les températures dans une liste et j'ai fait une moyenne en divisant la somme des températures de la liste par la taille de la liste.

Désormais, par exemple, si l'utilisateur veut une destination ensoleillée durant les 5 prochains jours, il n'a plus qu'à consulter cette activité et parcourir les villes populaires jusqu'à trouver ce qui lui correspond.



3. ACTIVITE CALENDRIER

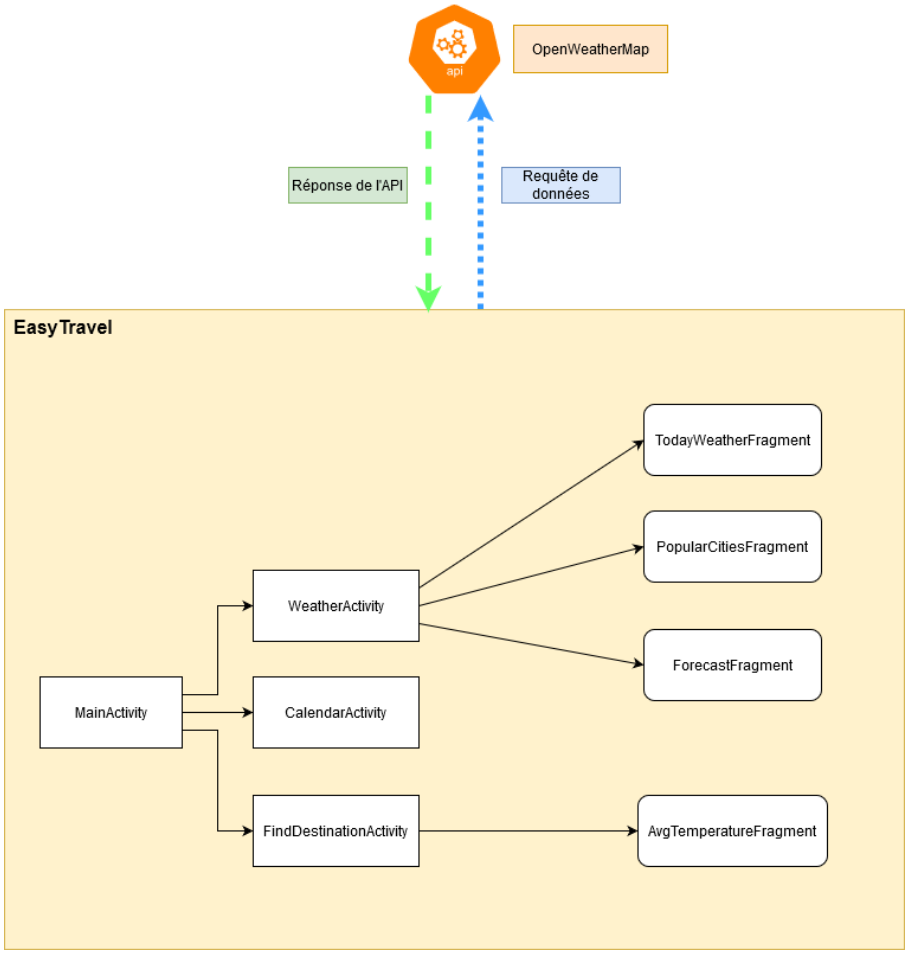
Dans cette activité, l'utilisateur a la possibilité d'enregistrer son premier jour de vacances. Il doit simplement sélectionner la date dans le calendrier, et ensuite, cliquer sur le bouton permettant de sauvegarder cette date.

Lorsque la date sera sauvegardée, une notification sera planifiée à cette date pour rappeler à l'utilisateur que c'est son premier jour de congé et qu'il peut venir chercher sa potentielle destination de vacances.

J'ai également fait en sorte que l'utilisateur ne puisse pas sélectionner une date inférieure à la date actuelle pour que l'application reste cohérente.

Pour le calendrier, j'ai utilisé CalendarView. CalendarView est un widget calendrier qui permet donc d'afficher et de sélectionner des dates.

Analyse



Dans le grand carré jaune, on peut voir les différentes activités et les différents fragments par activité de l'application EasyTravel. Lorsque l'utilisateur va lancer l'application, il arrivera dans le MainActivity qui correspond, ici, à la page d'accueil de l'application. Sur cette page d'accueil, l'utilisateur a trois possibilités : accéder à aux activités WeatherActivity, CalendarActivity ou FindDestinationActivity.

S'il décide d'accéder à l'activité WeatherActivity, il pourra alors naviguer entre les différents fragments qui offrent chacun des fonctionnalités distinctes.

S'il décide d'accéder à l'activité CalendarActivity, alors il pourra enregistrer son premier jour de vacances en sélectionnant la date dans le calendrier qui lui sera affiché.

Enfin, s'il décide d'accéder à l'activité FindDestinationActivity, alors il pourra consulter les prévisions météorologiques moyennes pour les 5 prochains jours pour les différentes villes populaires.

Comme on peut le voir sur le schéma, mon application fais des requêtes de données à l'API OpenWeatherMap et celle-ci renvoie une réponse à l'application.

Limitations et développement futur

Si j'avais eu plus de temps, j'aurais voulu faire en sorte que l'utilisateur puisse indiquer ses critères de recherches dans l'activité « Find a destination » afin de lui afficher seulement les villes ayant les prévisions météorologiques qui lui correspondent.

J'aurais également voulu améliorer le design de mon application pour qu'elle soit plus agréable à utiliser. J'aurais voulu ajouter en particulier une barre de navigation en bas permettant de naviguer entre les différentes activités.

Comme je ne possède pas de smartphone Android, je ne pouvais pas utiliser mon application directement sur mon téléphone, j'étais donc contraint à la tester uniquement sur l'émulateur fourni par Android Studio. Mais lors de la dernière séance de développement, j'ai voulu tester l'application sur le téléphone de l'un de mes collègues. En la testant, j'ai pu voir que tout fonctionnait bien mis à part une chose : l'affichage des icônes. En effet, les icônes permettant de décrire la météo ne s'affichent pas pour une raison que j'ignore. Cela veut dire que si je souhaite déployer l'application, il faudra prendre en compte ce problème et tenter de le régler.

Ce que j'aurais également bien voulu implémenter est une carte du monde. Sur cette carte du monde il serait possible, non seulement, d'accéder à la météo de n'importe quelle ville mais également de voir le dernier avis d'un utilisateur qui se serait rendu à cette ville pour des vacances. Cela veut dire que mon application contiendrait un forum sur lequel les utilisateurs de l'application peuvent donner leur avis et leurs conseils pour chaque ville. Cela aurait donc nécessité l'utilisation d'une

base de données afin de mettre en place un système de connexion et d'inscription et pour que chaque utilisateur puisse écrire son propre avis sur la ville qu'il a visité.

J'aurais également voulu faire en sorte que l'on puisse cliquer sur l'une des villes (dans l'activité « Find a destination » par exemple), et d'être redirigé sur un site où l'on peut réserver un vol en destination de la ville choisie.

L'un des défauts de mon application est le fait que l'on ne puisse pas revenir à la page d'accueil sans appuyer sur la flèche en dessous du téléphone (sauf dans l'activité « CalendarActivity »). Ce n'est pas très pratique et c'est un problème que j'aimerais beaucoup régler à l'avenir.

Conclusion

Comme je l'ai expliqué jusqu'ici, j'ai réussi à implémenter toutes les fonctionnalités de base qui ont été demandées par mon chef de projet. J'aurais voulu retravailler le design un peu plus en profondeur mais le temps m'en a empêché.

C'est la première application mobile que j'ai développée jusqu'à maintenant. C'était donc un début pour moi. De plus, j'ai également fait mes premiers pas avec l'utilisation d'une API. Ce projet n'a pu qu'être bénéfique pour moi, et m'a fait acquérir de l'expérience en développement. C'est un projet que je n'hésiterais pas à ajouter dans mon CV.