

# UnivMap

Damien LAOUSSING, Dylan CHERRIER, L3 informatique

1<sup>er</sup> mai 2021

## Résumé

UnivMap est une application mobile iOS et Android destinée aux nouveaux étudiants afin de les aider à se repérer au sein de l'Université de la Réunion.

## 1 Introduction

Dans le cadre du projet de l'unité d'enseignement "Développement pour Mobiles", nous avons décidé de développer une application iOS et Android. Cette application a pour but d'aider nos utilisateurs à se repérer à l'Université de la Réunion.

Le changement peut-être difficile pour certains étudiants réunionnais ou étranger. Beaucoup de ces étudiants, découvrant leurs nouveau campus, sont souvent perdu pour trouver leur cours. Par conséquent, ils arrivent généralement en retard et perdent ainsi leurs qualités d'apprentissages. Aujourd'hui, afin de répondre au besoin de ces étudiants, nous allons vous présenter l'application UnivMap.

Dans un premier temps, nous verrons une description générale de l'application, puis nous analyserons l'architecture du code. Ensuite, nous aborderons les difficultés rencontrés durant le développement du projet pour enfin terminer par une conclusion.

## 2 Description générale de l'application



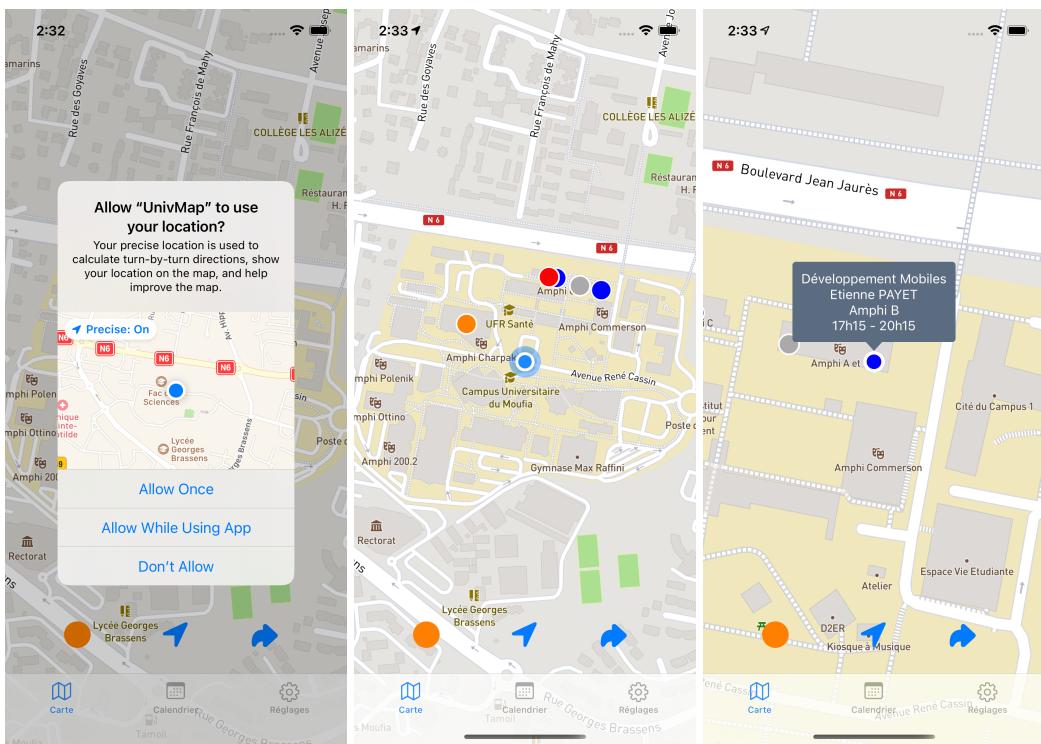
L'application possède trois onglets : la carte de l'université, le calendrier des cours ainsi que des options. La carte intègre des fonctionnalités permettant l'affichage des enseignements sous forme de cercle coloré. De plus, des boutons sont à la disposition de l'utilisateur afin de les parcourir. Le calendrier permet d'afficher sous forme d'une liste le planning de l'étudiant. Les options inclus la possibilité de changer la langue et la couleur de l'arrière plan.

L'application utilise principalement la carte de l'API *Mapbox* [3] développé par l'entreprise du même nom sur les bases du logiciel libre et sur les données d'*OpenStreetMap* [4]. La persistance de donnée de notre application est gérée par le système interne, mais aussi par la plateforme de Google, *Firebase* [2].

## 2.1 Présentation de la carte et annotations des points

Tout d'abord, UnivMap requiert une connexion internet afin de géolocaliser son utilisateur mais aussi pour récupérer des données indispensables à son fonctionnement (carte, liste des cours, ect...). Au lancement, l'utilisateur se retrouve sur l'onglet de la carte avec toutes les positions des cours. En cliquant sur l'un d'eux, un pop-up apparaît affichant les informations du cours (nom du cours, nom de l'enseignant, salle et horaire).

- Point bleu : les marqueurs de couleur bleu représente la position des cours qui n'ont pas encore commencé.
- Point orange : le marqueur de couleur orange représente la position du cours qui commencera dans moins de deux heures.
- Point rouge : le marqueur de couleur rouge représente la position du cours qui a déjà débuté.
- Point gris : le marqueur de couleur gris représente la position des cours terminé.

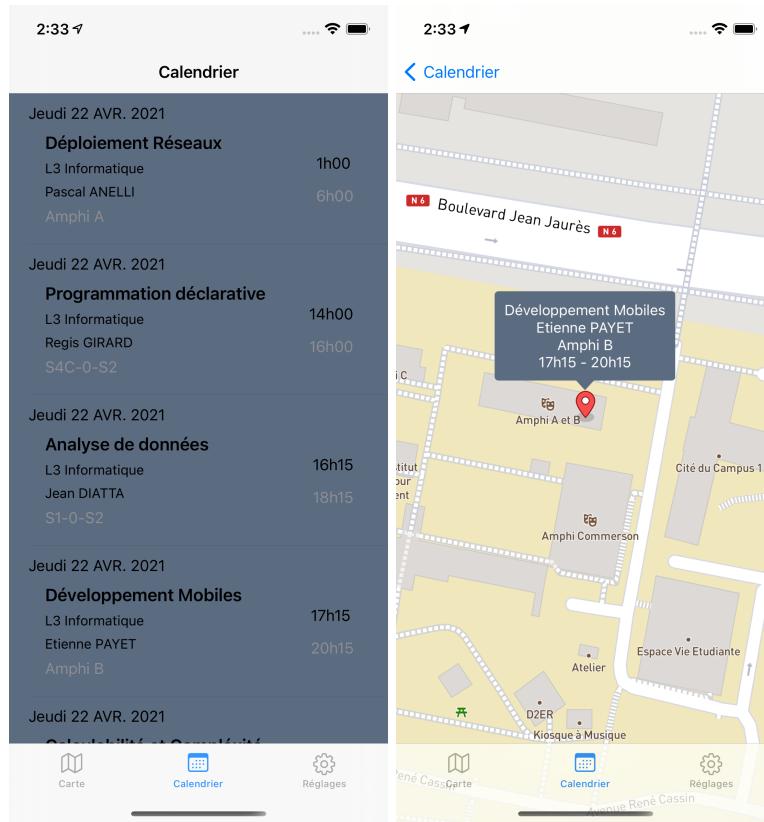


Les trois boutons du bas permettent de changer la vue de la caméra :

- Bouton de gauche : parcourir les cours qui débuterons prochainement.
- Bouton du milieu : centrer sur la position de l'utilisateur.
- Bouton de droite : parcourir parmis chaque cours.

## 2.2 Calendrier : la liste des cours

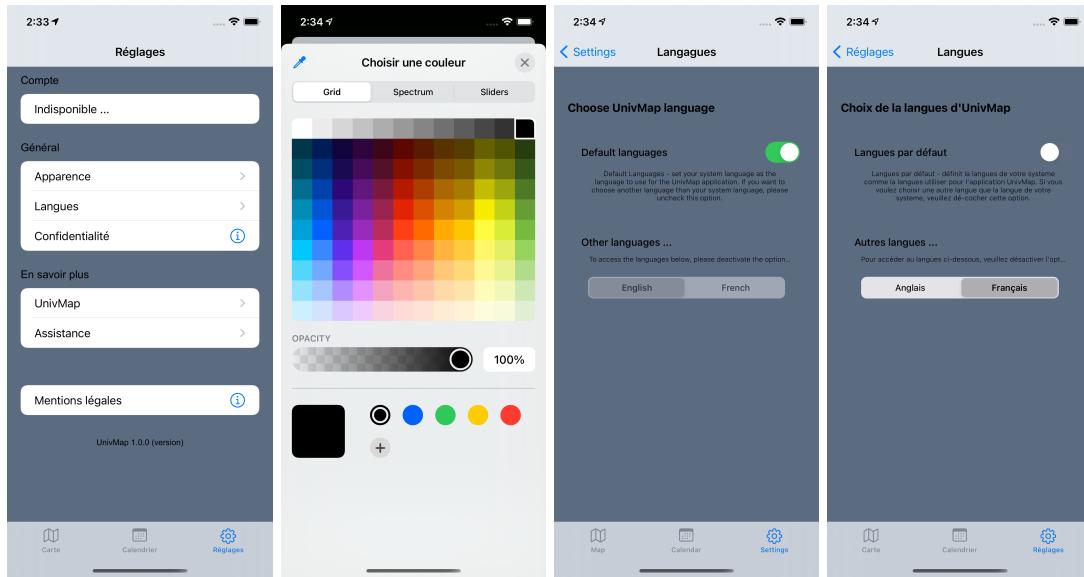
L'onglet calendrier permet l'affichage de tous les cours trier par ordre croissant selon l'heure. Pour récupérer ces données, l'application utilise *Firebase* [2], un service d'hébergement en NoSQL développé par Google. En sélectionnant le cours, l'application nous redirige vers une nouvelle vue, affichant la position exacte de celui-ci.



## 2.3 Présentation des paramétrages

Enfin l'onglet des paramètres de l'application, présenté sous forme de liste (généérer dynamiquement), permet à l'utilisateur de mieux s'approprier l'application et d'en apprendre d'avantage. À travers cet onglet, nous voulions encore une fois mettre en valeur notre volonté d'aider les étudiants, tout en leur apportant le maximum de confort. Pour cela, nous avons choisi de mettre en avant, pour cette première version d'UnivMap, les paramètres suivants :

- Le choix de la langue : la possibilité de choisir la langue de l'application est une option, qui nous semblent toujours indispensables. Cette option augmente le confort certe, mais surtout permet au public de différentes nationalités de l'utiliser. Pour cette option, l'utilisateur a le choix entre utiliser la même langue que son appareil, ou utiliser une des langues listées ci-dessous.
- Le choix du thème de l'application : le choix du thème de l'application permet à l'utilisateur de choisir la couleur de fonds.
- L'assistance : en cas de problème, l'utilisateur est invité à contacter l'assistance par mail. Ainsi en cliquant sur "assistance", puis envoyer, il est redirigé vers une application de mail.
- Mais aussi les mentions légales et la confidentialité : ces deux options sont présent à titre informatif et sont indispensables à tout application. Ces deux parties doivent faire apparaître respectivement la charte de protection des données personnelles, ainsi que les CGU(conditions générales d'utilisation), etc...
- Et enfin, UnivMap : cette option est présente pour permettre à l'utilisateur d'en apprendre d'avantage sur UnivMap. Notamment, le but de l'application ainsi que les développeurs ayant contribué à sa création.



### 3 Architecture du code

UnivMap est une application iOS et Android développer respectivement en Swift et Java.

#### 3.1 Android

##### 3.1.1 Java : Ouverture à la base de donnée

Cette fonction permet d'ouvrir la connexion à Firebase et de récupérer les données selon la clé. Elle retourne une liste d'objet en Callback.

```
public void getAllData(ListPlanningCallback listPlanningCallback) {
    db.collection("planning")
        .get()
        .addOnCompleteListener(new OnCompleteListener<QuerySnapshot>() {
            @Override
            public void onComplete(@NonNull Task<QuerySnapshot> task) {
                if (task.isSuccessful()) {
                    List<Planning> listLocal = new ArrayList<Planning>();

                    for (QueryDocumentSnapshot document : task.getResult()) {
                        //Log.d(TAG, document.getId() + " => " + document.getData());

                        String nom = document.getString("nom");
                        String filiere = document.getString("filiere");
                        String enseignant = document.getString("enseignant");
                        String hDebut = document.getString("hDebut");
                        String hFin = document.getString("hFin");
                        String mDebut = document.getString("mDebut");
                        String mFin = document.getString("mFin");
                        String salle = document.getString("salle");
                        String latitude = document.getString("latitude");
                        String longitude = document.getString("longitude");

                        Planning planning = new Planning(nom, filiere, enseignant,
                            hDebut, hFin, mDebut, mFin, salle, latitude, longitude);

                        listLocal.add(planning);
                    }
                    listPlanningCallback.onCallback(listLocal);
                } else {
                    Log.d(TAG, "Error getting documents: ", task.getException());
                }
            }
        });
}
```

### 3.1.2 Java : Choix de la langue

La première fonction permet de modifier la langue d'une chaîne de caractère et la seconde permet de récupérer la langue de l'appareil :

```
public static void setLocale(Activity activity, String languageCode) {  
    Locale locale = new Locale(languageCode);  
    Locale.setDefault(locale);  
    Resources resources = activity.getResources();  
    Configuration config = resources.getConfiguration();  
  
    if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.JELLY_BEAN_MR1) {  
        config.setLocale(locale);  
    } else {  
        config.locale = locale;  
    }  
    resources.updateConfiguration(config, resources.getDisplayMetrics());  
}  
  
public static String currentLanguage(){  
    return Resources.getSystem().getConfiguration().locale.getLanguage();  
}
```

## 3.2 iOS

### 3.2.1 Swift : Ouverture à la base de donnée

Contrairement à la section 3.1.1. Le code en Swift permettant de récupérer les données est facilité. Il n'y a aucunement besoin d'une fonction Callback pour ajouter à une liste d'objet.

```
db.collection("planning").getDocuments() { [self] (querySnapshot, err) in
    if let err = err {
        print("Error getting documents: \(err)")
    } else {

        for document in querySnapshot!.documents {
            if let nom = document.data()["nom"] as? String,           //recup data selon clé
                let filiere = document.data()["filiere"] as? String,
                let enseignant = document.data()["enseignant"] as? String,
                let hDebut = document.data()["hDebut"] as? String,
                let hFin = document.data()["hFin"] as? String,
                let mDebut = document.data()["mDebut"] as? String,
                let mFin = document.data()["mFin"] as? String,
                let salle = document.data()["salle"] as? String,
                let latitude = document.data()["latitude"] as? String,
                let longitude = document.data()["longitude"] as? String{

                self.listPlanning.append(Planning(nom: nom, filiere: filiere,
                    enseignant: enseignant, hDebut: hDebut, hFin: hFin, mDebut: mDebut, mFin: mFin,
                    salle: salle, latitude: latitude, longitude: longitude))

            (...)

        }
        (...)

    }
}
```

### 3.2.2 Swift : Choix de la langue

Cette fonction permet la traduction d'une chaîne de caractère de trois manières différentes :

- Selon la langue passé en paramètre : affiche toujours un texte dans une langue en particulier !
- Selon la langue par défaut : affiche toujours un texte dans la langue par défaut, c'est-à-dire la langue de l'appareil.
- Selon le choix de l'utilisateur : affiche le texte dans la langue choisi par l'utilisateur.

```
extension String {  
    func localized(str:String?) -> String{  
        if let language = str{  
            let path = Bundle.main.path(forResource: language, ofType: "lproj")  
            let bundle = Bundle(path: path!)  
            return NSLocalizedString(self, tableName: "Localizable", bundle: bundle!,  
                value: self, comment: "nil")  
        }  
        if(UserDefaults.standard.bool(forKey: "switchState")){  
            return NSLocalizedString(self, tableName: "Localizable", bundle: .main,  
                value: self, comment: "nil")  
        }  
        let path = Bundle.main.path(forResource: UserDefaults.standard.string(forKey: "language"),  
            ofType: "lproj")  
        let bundle = Bundle(path: path!)  
        return NSLocalizedString(self, tableName: "Localizable", bundle: bundle!, value: self,  
            comment: "nil")  
    }  
}
```

## 4 Quelques points délicats/intéressants

Durant le développement de notre application, nous avons rencontrés des difficultés concernant la prise en main de nouveaux outils de travail tels que Git. De plus, l'implémentation de l'API Mapbox et de *Firebase* [2] a beaucoup ralenti le développement sur iOS. En effet, *Firebase* [2] étant développé par Google, son importation était une épreuve sur Xcode. Afin d'installer les dépendances requises sur iOS, nous avons utilisés le gestionnaire de dépendances *CocoaPods* [1]. Malgré toutes ces difficultés, l'API Mapbox nous a permis de finir ce projet dans les temps. Mapbox étant très complet, possèdent ses propres classes et fonctions permettant aux développeurs de l'intégrer facilement dans leurs projets. Concernant *Firebase* [2], son utilisation est plus aisée en Swift. En effet, Swift permet l'ajout de valeur dans une liste d'objet déclarer globalement. Contrairement à Java, ou il a fallu passé par une fonction callback (voir section 3.1.1) pour récupérer les données.

La traduction de l'application, en iOS, révèle la puissance des extensions permettant ainsi de rajouter des fonctions, structures, etc... Et cela même à des classes normalement inaccessibles (code source du langage de programmation) comme la classe String. Néanmoins, il était plus simple de gérer la traduction en Java. La gestion de la traduction et de l'apparence de l'application, nous a confronté au cycle de vie (des view/activity) des différentes plateformes. Et l'utilisation des données persistances sous forme de couple (clé, valeur), pour enregistrer des petites données (préférences utilisateurs), est remarquable. Les données persistantes nous ont même permis de récupérer des paramètres initiales et exécuter du code (la première fois qu'on lance l'application).

### 4.1 UnivMap 2.0 à suivre

À la fin du développement de notre application, bien qu'elle soit fonctionnelle, certaines améliorations sont à venir, telles que l'ajout :

- du choix du planning selon la filière ;
- de notification lorsque l'emploi du temps est modifié ;
- d'un onglet accueil avec des redirections vers d'autres plateformes de l'Université et l'affichages d'évènements ;
- de filtre sur la carte.

## 5 Conclusion

Enfin pour conclure le rapport, ce projet nous a beaucoup apporté en tant que développeur mobile que ce soit sur les connaissances basiques en passant par l'implémentation d'API. Le développement d'UnivMap à montré que le travail de groupe, l'entraide et la communication étaient primordial. Malgré quelque appréhension face à l'idée de l'application, notre binôme à su montré son ambition et sa persévérance d'atteindre l'objectif d'aider de nouveaux étudiants.

## Références

- [1] Cocoapods. <https://cocoapods.org/>.
- [2] Firebase. <https://firebase.google.com/>.
- [3] Mapbox. <https://www.mapbox.com/>.
- [4] Openstreetmap. <https://www.openstreetmap.fr/>.