

UnivMap

Damien LAOUSSING, Dylan CHERRIER, L3 informatique

30 avril 2021

Résumé

UnivMap est une application mobile iOS et Android destinée au nouveau étudiant 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, visant les nouveaux étudiants de l'Université de la Réunion.

Beaucoup d'étudiants découvrant leurs nouveau campus, sont souvent perdu pour retrouver leurs chemin. Par conséquent, ils arrivent souvent en retard dans leur cours et perdent ainsi leur qualité d'apprentissage. Aujourd'hui, afin de répondre aux besoin de ces nouveaux étudiants, nous allons vous présenter l'application UnivMap.

Dans un premier temps, nous allons voir une description générale de l'application, puis nous verrons 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

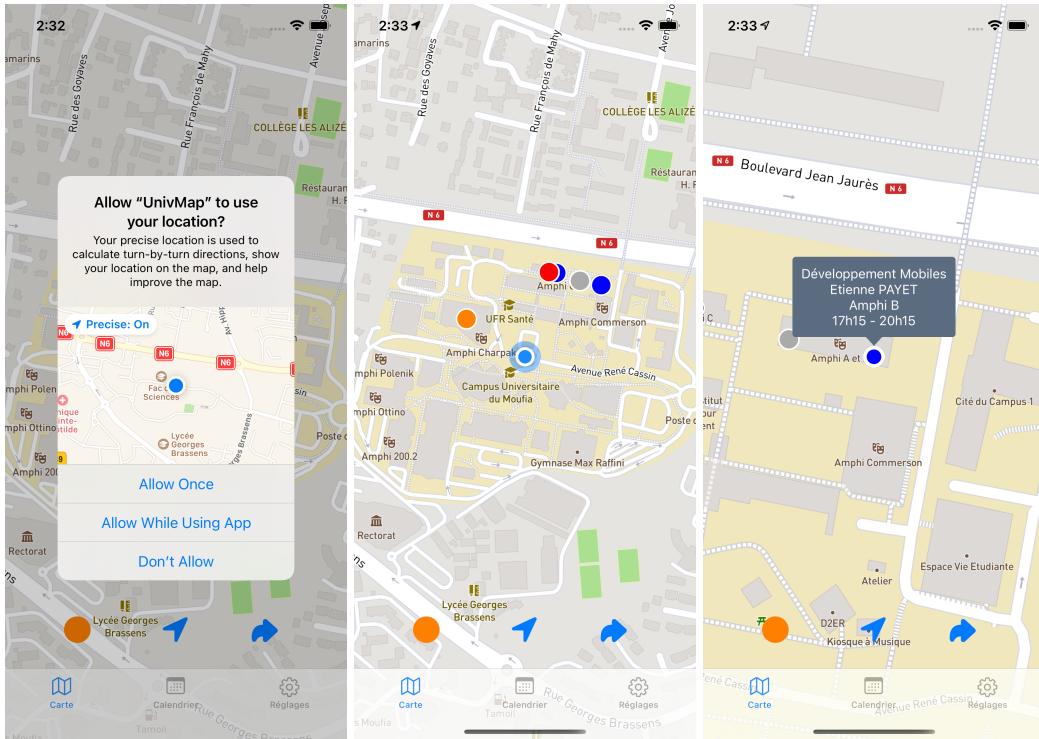
L'application possède trois onglets : la carte de l'Université, le calendrier des cours ainsi que des options. La carte intègre des fonctionnalités permettant l'affichage des enseignements sous forme de cercle coloré. De plus, des boutons sont à la disposition de l'utilisateur afin de parcourir automatiquement ces cercles. Le calendrier permet d'afficher sous forme de liste le planning. Les options incluent la possibilité de changer la couleur du fond de l'application et de changer la langue.

L'application utilise principalement la carte de l'API *Mapbox* [3] développée par l'entreprise du même nom sur les bases du logiciel libre et sur les données d'*OpenStreetMap* [4]. La persistance de données 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 (map, liste des cours, ect...). Au lancement, l'utilisateur se retrouve sur l'onglet de la carte et voit tous les positions des cours. En cliquant sur l'un d'eux, un pop-up apparait affichant les informations essentielles 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 commençera 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 du cours terminé

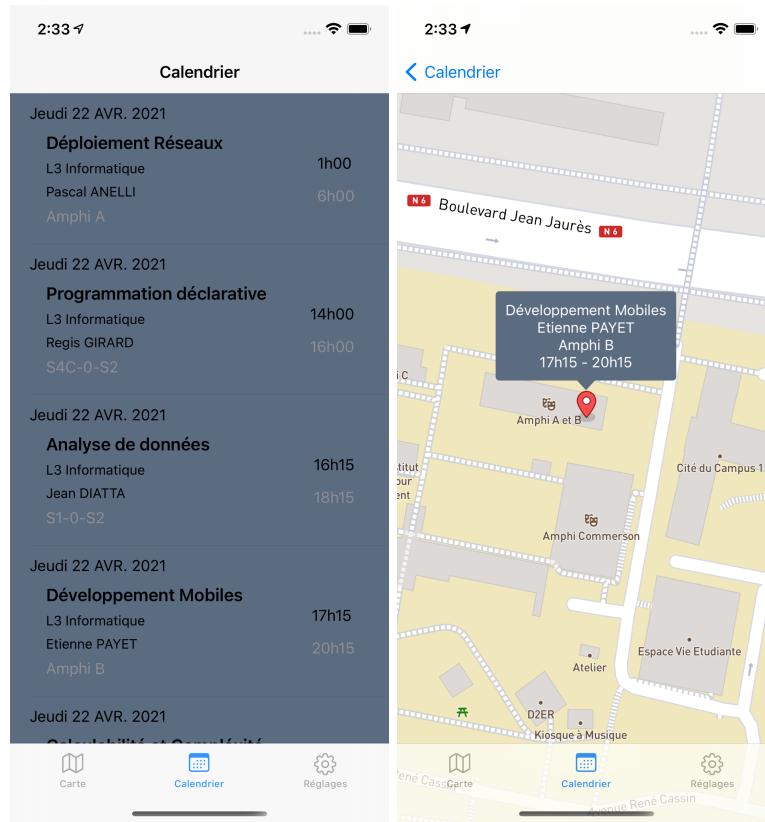


Les trois boutons du bas permet de changer la vue de la caméra.

- Bouton de gauche : parcours les cours qui débuterons prochainement
- Bouton du milieu : centre sur la position de l'utilisateur.
- Bouton de droite : permet de parcourir parmis tous les cours sur la map.

2.2 Calendrier : la liste des cours

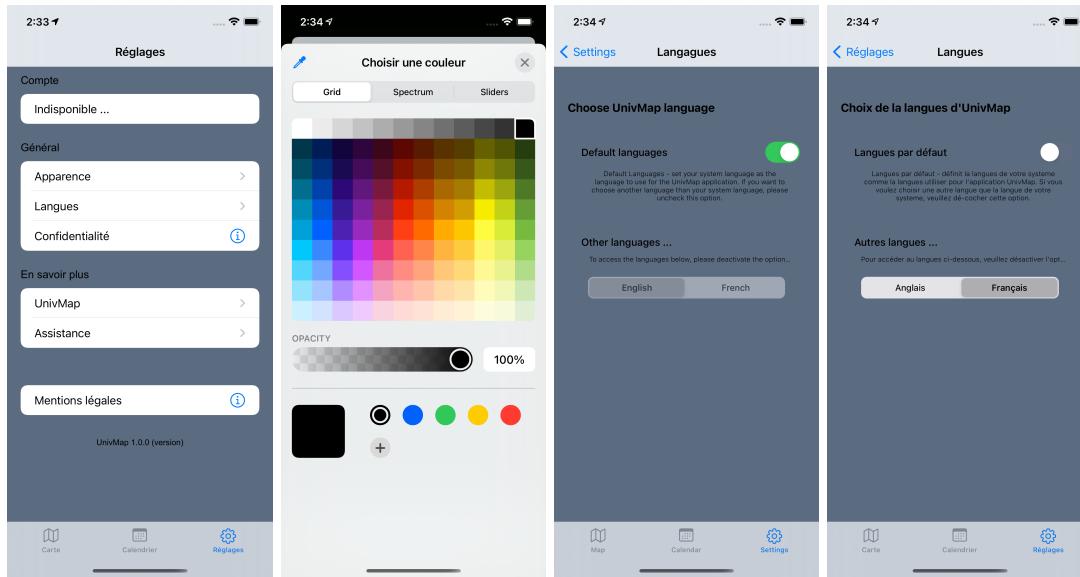
L'onglet calendrier permet l'affichage de tous les cours trier par ordre croissant selon l'heure. Pour récupérer ces données, l'application utilise *Firebase* [2], un service d'hébergement en NoSQL développé par Google. En sélectionnant le cours, l'application nous redirige vers une nouvelle vue, affichant 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 plus sur elle. À travers cette l'onglet, nous voulions encore une fois mettre en valeurs notre volonté d'aider les utilisateurs, tout en leurs apportant le maximum de confort. Pour cela, nous avons choisi de mettre en avant, pour cette première version de UnivMap, les paramètres suivants :

- Le choix de la langue : La possibilité de choisir la langues de l'application est une option, qui nous semble toujours indispensable. Cette option augmente le confort certe, mais surtout permet aux public de différentes nationalité de l'utiliser. Pour cet option, l'utilisateur a le choix entre utiliser la même langue que son appareil, ou utiliser une des langues lister en-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, .
- Ainsi que 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 rediriger dans une application de mail pour envoyer un mail à l'adresse qui y préfigure.
- Mais aussi le mentions légales et la Confidentialité : Ces deux options sont présent à titre informatif, et sont indispensable à tout application qui ce veut accessible aux public. Dans ces deux partie doivent paraître respectivement la charte de protection des données personnelles, etc... et les CGU(conditions générales d'utilisation), etc...
- Et enfin, UnivMap : Cette option des paramètres est présente pour permet à l'utilisateur d'en apprendre plus sur UnivMap. Notamment, le but d'UnvMap, qui sont ses créateur, et suivre l'évolution de l'application.



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()) {

                        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...);
                        listLocal.add(planning);

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

public interface ListPlanningCallback {
    void onCallback(List<Planning> listPlanning);
}
```

3.1.2 Java : Choix de la langue

La première fonction permet de modifier la langue d'une chaîne de caractère et la deuxième 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

Voici la même code en Swift que dans la section 3.1.1. Ici, il faut juste ajouter à listPlanning (initialiser en variable global).

```

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
                filiere = document.data()["filiere"] as? String,
                enseignant = document.data()["enseignant"] as? String,
                hDebut = document.data()["hDebut"] as? String,
                hFin = document.data()["hFin"] as? String,
                mDebut = document.data()["mDebut"] as? String,
                mFin = document.data()["mFin"] as? String,
                salle = document.data()["salle"] as? String,
                latitude = document.data()["latitude"] as? String,
                longitude = document.data()["longitude"] as? String{

                self.listPlanning.append(Planning(nom: nom, filiere: filiere, ...))

            (...)

        }
    }
}

```

3.2.2 Swift : Choix de la langue

Cette fonction permet de modifier la langue d'une chaîne de caractère de trois manière différente :

- Selon la langue passer 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 choisie 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 nouveau outils de travail tels que Git. De plus, l'implémentation de l'API Mapbox et de *Firebase* [2] à 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 du passé par le gestionnaire de dépendances *CocoaPods* [1]. Malgré tous ces difficultés et après avoir pris en main, l'API Mapbox nous a permis de finir ce projet dans les temps. Mapbox étant très complet, possèdent ses propres classe permettant aux développeur de l'intégrer facilement dans leurs projets. Concernant *Firebase* [2], son utilisation est plus aisé 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.

5 Conclusion

Références

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