

UnivMap

L3 informatique

Damien Laoussing Dylan Cherrier

Département d'informatique

2 mai 2021

Plan

1 Introduction

2 Présentation d'UnivMap

- Comment cela fonctionne ?

3 Architecture du code

- Firebase
- Mapbox
- Paramètres : apparence
- Paramètres : langues

4 Quelques difficultés

5 UnivMap 2.0 : À Suivre

6 Conclusion

Introduction



Introduction : pourquoi cette application ?

- Contribuer au développement de l'Université
- Aider les nouveaux étudiants Réunionnais ou étrangers
 - Généralement perdu
 - Par conséquent : retard en cours voir pire en examen !
 - Donc qualité d'apprentissage diminué
- Très peu d'application voir pas du tout

Plan

1 Introduction

2 Présentation d'UnivMap

- Comment cela fonctionne ?

3 Architecture du code

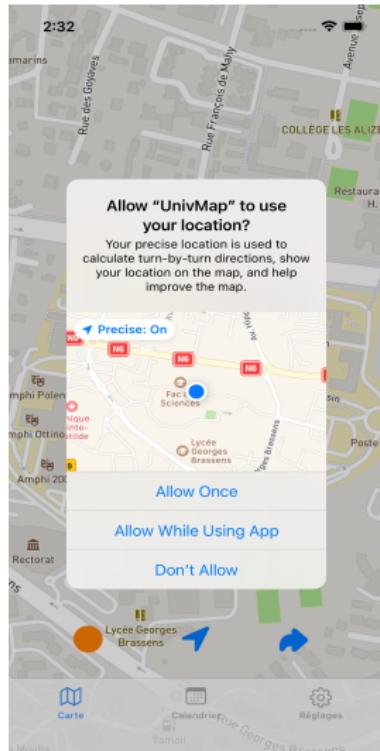
- Firebase
- Mapbox
- Paramètres : apparence
- Paramètres : langues

4 Quelques difficultés

5 UnivMap 2.0 : À Suivre

6 Conclusion

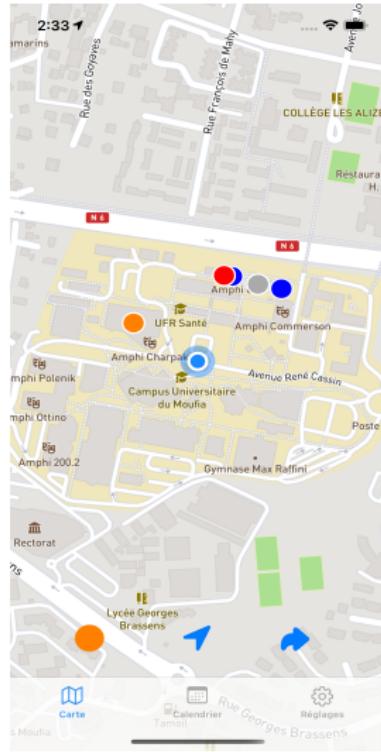
Comment fonctionne : la carte



UnivMap requiert une connexion internet afin de localiser la position de l'utilisateur

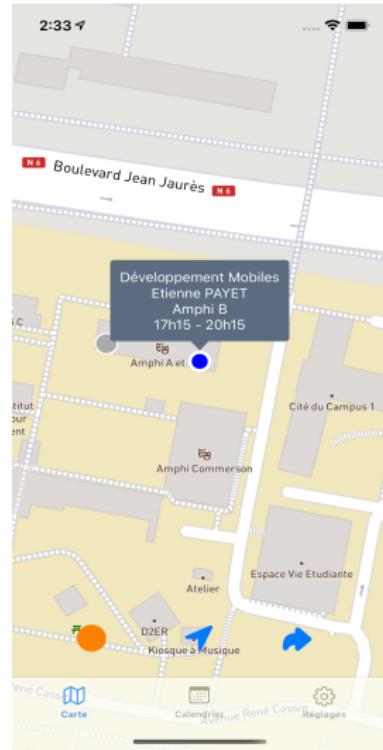
Comment fonctionne : la carte

- Marqueur **BLEU** : cours qui n'ont pas encore débuté
- Marqueur **ORANGE** : cours qui débuterons dans moins de 2 heures
- Marqueur **ROUGE** : cours qui ont déjà commencé
- Marqueur **GRIS** : cours terminé



Comment fonctionne : la carte

En sélectionnant un marqueur, un pop-up apparaît affichant les informations du cours



Architecture du code : le calendrier



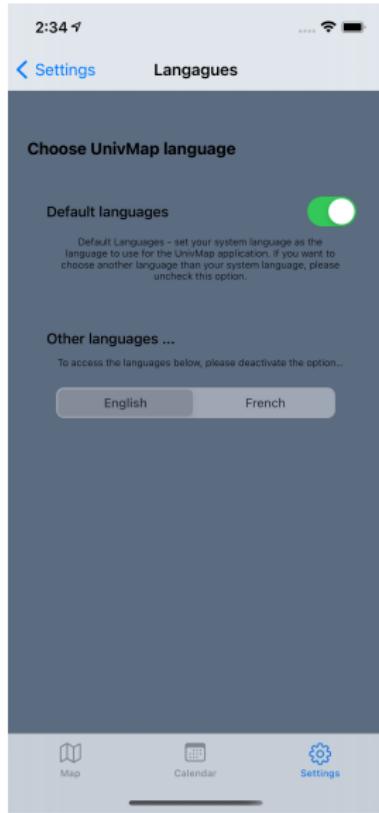
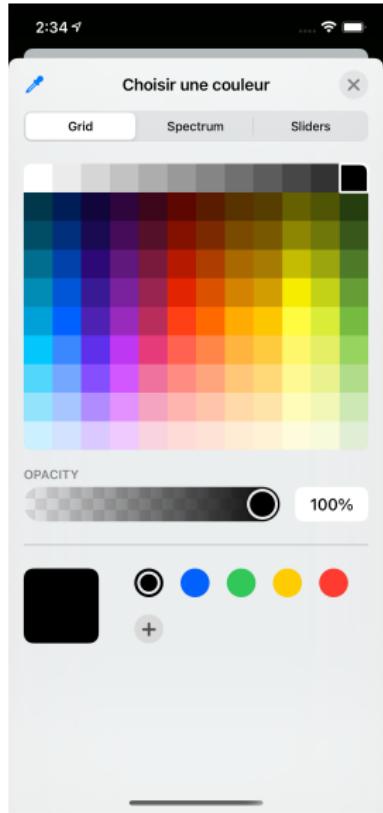
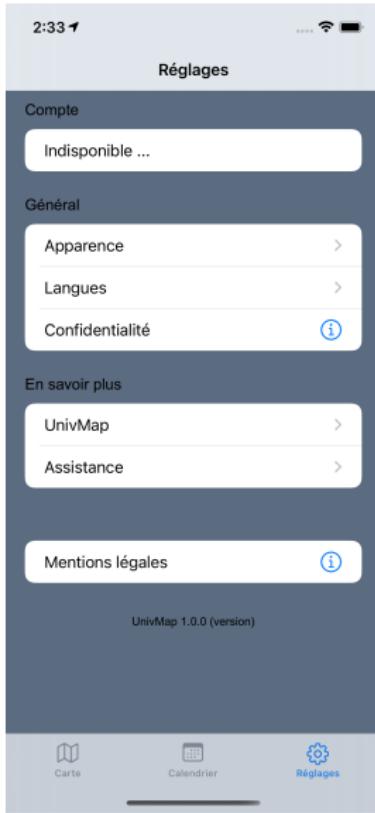
Ici tous les cours sont affichés, trier par ordre croissant selon l'heure

Architecture du code : le calendrier



En cliquant sur un cours, l'utilisateur est redirigé vers une autre vue affichant la position du cours.

Introduction



Plan

1 Introduction

2 Présentation d'UnivMap

- Comment cela fonctionne ?

3 Architecture du code

- Firebase
- Mapbox
- Paramètres : apparence
- Paramètres : langues

4 Quelques difficultés

5 UnivMap 2.0 : À Suivre

6 Conclusion

Firebase : c'est quoi ?

- Services d'hébergement pour tout type d'application
- Racheté par Google en 2014
- Firebase : NoSQL
- Utilisé par plus de 110 000 développeurs :
 - Shazam (iOS et Android)
 - Le Figaro (iOS et Android)
 - ...

Firebase : c'est quoi ?

The screenshot shows the Firebase Cloud Firestore interface for a project named "UnivMap". The left sidebar includes links for Vue d'ensemble du projet, Authentication, Firestore Database, Realtime Database, Storage, Hosting, Functions, Machine Learning, Publier et surveiller (Analytics, Performance, Test Lab), and Extensions. The main area displays the "Données" tab for the "planning" collection. A banner at the top says "Prototypez et testez vos applications de bout en bout avec la suite d'émulateurs locaux, désormais compatible avec Firebase Authentication". Below the banner, the "planning" collection has a document with ID "UC59DC9pZEgg...". This document contains fields such as "enseignant" (value: "Etienne PAYET"), "filiere" (value: "L3 Informatique"), "hDebut" (value: "17"), "hFin" (value: "20"), "latitude" (value: "-20.90098"), "longitude" (value: "55.48532"), "mDebut" (value: "15"), "mFin" (value: "15"), "nom" (value: "Développement Mobiles"), and "salle" (value: "Amphi B").

Architecture du code : Firebase

```
341     // Fonction pour récupérer les données sur Firecore en Callback
342     public void getAllData(ListPlanningCallback listPlanningCallback) {
343         db.collection( collectionPath: "planning")
344             .get()
345             .addOnCompleteListener(new OnCompleteListener<QuerySnapshot>() {
346                 @Override
347                 public void onComplete(@.NonNull Task<QuerySnapshot> task) {
348                     if (task.isSuccessful()) {
349                         List<Planning> listLocal = new ArrayList<>();
350
351                         for (QueryDocumentSnapshot document : task.getResult()) {
352                             //Log.d(TAG, document.getId() + " => " + document.getData());
353
354                             String nom = document.getString( field: "nom");
355                             String filiere = document.getString( field: "filiere");
356                             String enseignant = document.getString( field: "enseignant");
357                             String hDebut = document.getString( field: "hDebut");
358                             String hFin = document.getString( field: "hFin");
359                             String mDebut = document.getString( field: "mDebut");
360                             String mFin = document.getString( field: "mFin");
361                             String salle = document.getString( field: "salle");
362                             String latitude = document.getString( field: "latitude");
363                             String longitude = document.getString( field: "longitude");
364
365                             Planning planning = new Planning(nom, filiere, enseignant, hDebut, hFin, mDebut, mFin, salle, latitude, longitude);
366                             listLocal.add(planning);
367
368                         }
369                         listPlanningCallback.onCallback(listLocal);
370                     } else {
371                         Log.d(TAG, msg: "Error getting documents: ", task.getException());
372                     }
373                 }
374             });
375         }
376     }
377
378     public interface ListPlanningCallback {
379         void onCallback(List<Planning> listPlanning);
380     }
}
```

Mapbox : c'est quoi ?

- Entreprise américaine spécialisée dans la cartographie en ligne
- Mapbox reposent principalement sur le logiciel libre et sur les données d'OpenStreetMap
- Fournit ses services notamment à :
 - Pinterest
 - Le Monde
 - Snapchat
 - ...

Architecture du code : Mapbox

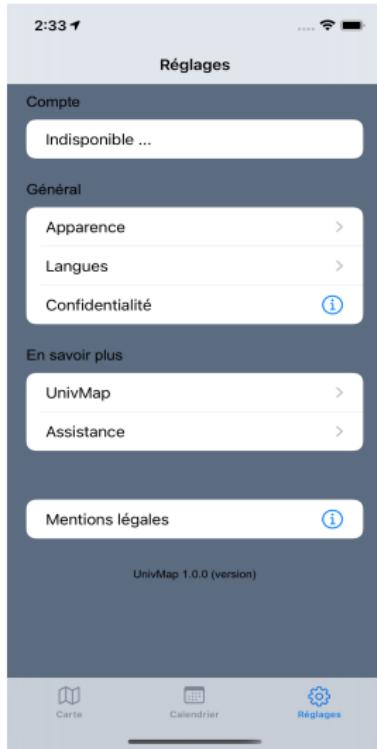
```
707     // Fonction pour créer les points coloré en cercle
708     public Circle CreateCircle(double latitude, double longitude, int color ){
709
710         circle = circleManager.create(new CircleOptions()
711             .withLatLng(new LatLng(latitude, longitude))
712             .withCircleColor(ColorUtils.colorToRgbaString(color))
713             .withCircleRadius(10f)
714             .withCircleStrokeColor(ColorUtils.colorToRgbaString(Color.WHITE))
715             .withCircleStrokeWidth(3f)
716             .withDraggable(false)
717         );
718         return circle;
719     }
720 }
```

Architecture du code : Mapbox

```
730 // Fonction pour ajouter les annotations des points avec les infos(nom, salle...)
731 public MarkerView annotations(String nom, String enseignant, String salle, String horaire, double planningLatitude, double planningLongitude){
732
733     // Supprime l'annotation à quand on clique sur un point différent
734     markerViewManager.removeMarker(markerView);
735
736     // Use an XML layout to create a View object
737     View customView = LayoutInflater.from(MainActivity.this).inflate(R.layout.custom_marker_view, root: null);
738     customView.setLayoutParams(new FrameLayout.LayoutParams(WRAP_CONTENT, WRAP_CONTENT));
739
740     // Set the View's TextViews with content
741     TextView planningNom = customView.findViewById(R.id.planning_nom);
742     planningNom.setText(nom);
743     planningNom.setTextColor(ColorCustomPopUp.adaptativeColor(r,g,b));
744
745     TextView planningEnseignant = customView.findViewById(R.id.planning_enseignant);
746     planningEnseignant.setText(enseignant);
747     planningEnseignant.setTextColor(ColorCustomPopUp.adaptativeColor(r,g,b));
748
749     TextView planningSalle = customView.findViewById(R.id.planning_salle);
750     planningSalle.setText(salle);
751     planningSalle.setTextColor(ColorCustomPopUp.adaptativeColor(r,g,b));
752
753     TextView planningHoraire = customView.findViewById(R.id.planning_horaire);
754     planningHoraire.setText(horaire);
755     planningHoraire.setTextColor(ColorCustomPopUp.adaptativeColor(r,g,b));
756
757     CardView card = customView.findViewById(R.id.cardMarker);
758     card.setCardBackgroundColor(Color.argb(alpha: 255,r,g,b));
759
760     // Use the View to create a MarkerView which will eventually be given to
761     // the plugin's MarkerViewManager class
762     markerView = new MarkerView(new LatLng(planningLatitude, planningLongitude), customView);
763
764     customView.setOnClickListener(new View.OnClickListener() {
765         @Override
766         public void onClick(View v) { markerViewManager.removeMarker(markerView); }
767     });
768     return markerView;
769 }
770 }
```

Architecture du code : les paramètres

- Une liste d'options en UITableViewController(iOS) et listView(Android)
- Sélection d'une option redirige vers une nouvelle View/Activity



Architecture du code : apparence

- Utilisation du colorPickerViewController
- Changement de couleur de chaque Views grâce à la persistance de donnée (UserDefaults(iOS)/ SharedPreferences(Android))

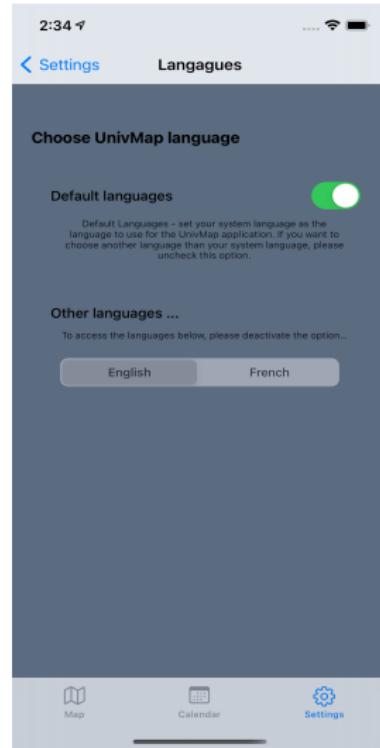


Architecture du code : apparence

```
73     //MARK: Extension UserDefaults
74 extension UserDefaults {
75     func setColor(color: UIColor?, forKey key: String){
76         var colorData: Data?
77         if let color = color{
78             do{
79                 colorData = try NSKeyedArchiver.archivedData(withRootObject: color, requiringSecureCoding: false) as
80                     Data?
81                 set(colorData, forKey: key)
82             }catch let err{
83                 print("error archiving color data", err)
84             }
85         }
86     }
87     func color(key: String) -> UIColor?{
88         var color: UIColor?
89         if let colorData = data(forKey: key){
90             do{
91                 color = try NSKeyedUnarchiver.unarchivedObject(ofClass: UIColor.self, from: colorData)!
92             }catch let err{
93                 print("error unarchiving color data" , err)
94             }
95         }
96         return color
97     }
98 }
```

Architecture du code : langues

- Utilisation du Bundle pour changer la langue
- Bundle : permet d'accéder à une ressource (fichier)



Architecture du code : langues

```
129 extension String {
130     func localized(str:String?) -> String{
131         // display a language choose by dev (constant language for users)
132         if let language = str{
133             let path = Bundle.main.path(forResource: language, ofType: "lproj")
134             let bundle = Bundle(path: path!)
135             return NSLocalizedString(self, tableName: "Localizable", bundle: bundle!, value: self, comment: "nil")
136         }
137         // display a Device language
138         if( UserDefaults.standard.bool(forKey: "switchState")){
139             //print("\(UserDefaults.standard.bool(forKey: "switchState")) vkrhke")
140             return NSLocalizedString(self, tableName: "Localizable", bundle: .main, value: self, comment: "nil")
141         }
142         // display a language choose by users
143         let path = Bundle.main.path(forResource: UserDefaults.standard.string(forKey: "language"), ofType: "lproj")
144         let bundle = Bundle(path: path!)
145         return NSLocalizedString(self, tableName: "Localizable", bundle: bundle!, value: self, comment: "nil")
146     }
147 }
```

Plan

1 Introduction

2 Présentation d'UnivMap

- Comment cela fonctionne ?

3 Architecture du code

- Firebase
- Mapbox
- Paramètres : apparence
- Paramètres : langues

4 Quelques difficultés

5 UnivMap 2.0 : À Suivre

6 Conclusion

Quelques difficultés

- Apprendre l'utilisation des API
- Comprendre les documentations
- Prise en main de Git
- Implémentation de Firebase sur iOS
- Changement de langues sur iOS
- Android : utilisation de fragment pour la navigation (crash de l'application)

Plan

1 Introduction

2 Présentation d'UnivMap

- Comment cela fonctionne ?

3 Architecture du code

- Firebase
- Mapbox
- Paramètres : apparence
- Paramètres : langues

4 Quelques difficultés

5 UnivMap 2.0 : À Suivre

6 Conclusion

UnivMap 2.0 : À Suivre

- Choix du planning selon une filière
- Notification lorsque l'emploi du temps est modifié
- Un onglet accueil afin d'être redirigé vers d'autres plateformes de l'Université
- Ajout de filtre pour la carte pour n'afficher que le contenu voulu
- Un système permettant de rechercher des salles

Plan

1 Introduction

2 Présentation d'UnivMap

- Comment cela fonctionne ?

3 Architecture du code

- Firebase
- Mapbox
- Paramètres : apparence
- Paramètres : langues

4 Quelques difficultés

5 UnivMap 2.0 : À Suivre

6 Conclusion

Conclusion



- Appréhension sur l'idée de l'application
- Travail de groupe : communication, entraide étaient très important
- L'ambition et la persévérance d'atteindre cette objectif