



Ecole d'ingénieurs et d'architectes de Fribourg
Hochschule für Technik und Architektur Freiburg



École supérieure d'ingénieurs de Beyrouth

Thèse de Bachelor :

ESIB@Pad

**Software Design
Description**

Auteur	Elias Medawar elias.medawar@edu.hefr.ch	
Responsables Internes	Omar Abou Khaled omar.aboukhaled@hefr.ch	Elena Mugellini elena.mugellini@hefr.ch
Responsable externe	Dany Mezher dany.mezher@fi.usj.edu.lb	
Experts	Marc Wuergler marc.wuergler@gmail.ch	Roland Marro marror@fr.ch

Version 1

10 août 2011

Table des matières

1	Introduction	2
1.1	But du document	2
1.2	Aperçu du document	2
2	Architecture du système	2
2.1	Architecture choisie	2
2.2	Discussion des alternatives d'architectures	3
2.2.1	Alternative 1 : Sans base de données	3
2.2.2	Alternative 2 : Objet pour la communication en C++	3
2.3	Composants du système	4
2.3.1	Description des composants	4
3	Conception et Implémentation des composants	5
3.1	Système de cache	5
3.2	Base de donnée	5
3.3	Base technique du développement iOS	5
3.4	Navigation	6
3.5	Settings	9
3.6	Map	11
3.7	News	15

Évolution de ce document

Rev	Date	Auteur	Remarque
1	13.06.2011	Medawar	Création de la premières version du STD.

1 Introduction

1.1 But du document

Ce document décrit les variantes d'architecture étudié pour le projet ESIP@PAD, l'architecture finale choisie ainsi que le détail du design final du projet. A l'aide de ce document il est possible de comprendre le fonctionnement technique de l'ensemble du projet.

1.2 Aperçu du document

:TODO: Describe this

2 Architecture du système

2.1 Architecture choisie

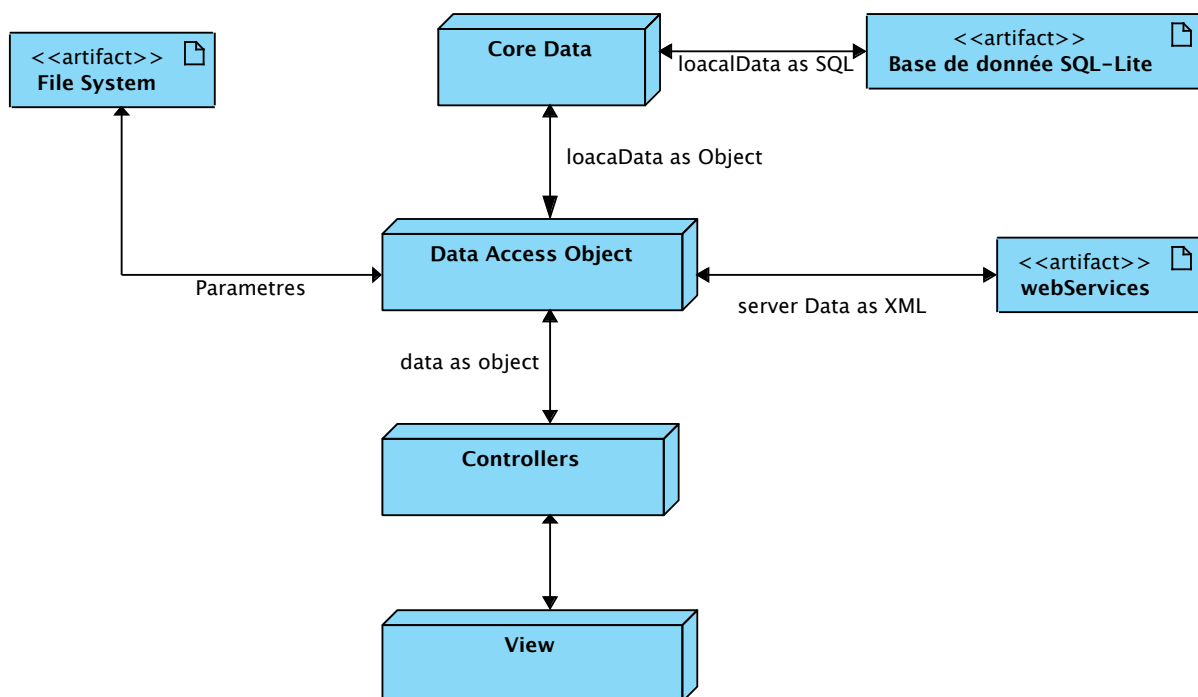


FIGURE 1: Vue global de l'architecture du système

Ce diagramme(Figure 1) nous donne un aperçu des différentes couche qui forment l'architecture du système.

View : Cette couche du système sert a représenter graphiquement l'information. Elle est étroitement lié à la couche Controllers.

Controllers : Cette couche du système s'occupe de charger les données dans la vue. Et

de réagir correctement au événement reçu depuis l'utilisateur.

Data Access Object : Cette couche fait référence au pattern de DAO¹ qui nous permet d'accéder aux données de notre application sans se soucier d'où elles proviennent, d'où elles seront stockées ni comment. Elle contient aussi la logique métier de l'application. A l'aide de cette couche, il nous est facile de changer le support de stockage des données car toute la logique et l'accès au données y est centralisé.

Core Data est le frameworks de persistance de l'iOS qui permet d'accéder d'un manière simplifié au données stockées dans la base de données SQL-Lite ou sous format XML. Ce frameworks nous permet de décrire la base de données et ses relations et de générer des objets Objective-c qui correspondes aux entités de la base de données. Les principes du fonctionnement sont les mêmes que ceux du fameux framework JPA² de java.

2.2 Discussion des alternatives d'architectures

Beaucoup d'alternatives d'architectures se sont offertes à nous en début du projet et voici les principales.

2.2.1 Alternative 1 : Sans base de données

Cette alternative supprime la couche core data et la remplace par un stockage des données xml reçu des web-services sur le support de données de l'appareil. Cette alternative requière moins de temps de développement mais en contrepartie, il faut à chaque utilisation des données parser les fichiers xml. L'iOS n'est pas encore optimisé pour le traitement des fichiers xml et permet uniquement de parser un fichier mais sans faire des requêtes du type XPath sur les fichiers xml. Des frameworks ont été développés par divers entreprises pour permettre le requêtage de fichiers XML, mais leurs performances restent tout de même moins bonne que celle d'une base de données.

2.2.2 Alternative 2 : Objet pour la communication en C++

Cette alternative propose de décrire les objets pour la communication(Core data \Leftrightarrow DAO \Leftrightarrow Controllers) en C++. Avec cette alternative, on augmente la portabilité de notre application et offre la possibilité de réutilisé ces même objet sur d'autre plateforme(tel que Android). Après une bref recherche il s'est avéré que Core data n'est pas capable d'utiliser les objet C++.. Comme nous utilisons une base de données SQL-Lite et que Core data génère automatiquement les objets correspondant au contenu de la base de données nous avons mis de côtés cette alternative. Par contre la base de données étant décrite en SQL-Lite peut être exporter vers d'autre appareil et de cette base de données il est facile de générer les objets de communication à l'aide des outils propre à chaque plateforme.

1. <http://java.sun.com/blueprints/corej2eepatterns/Patterns/DataAccessObject.html>

2. http://en.wikipedia.org/wiki/Java_Persistence_API

2.3 Composants du système

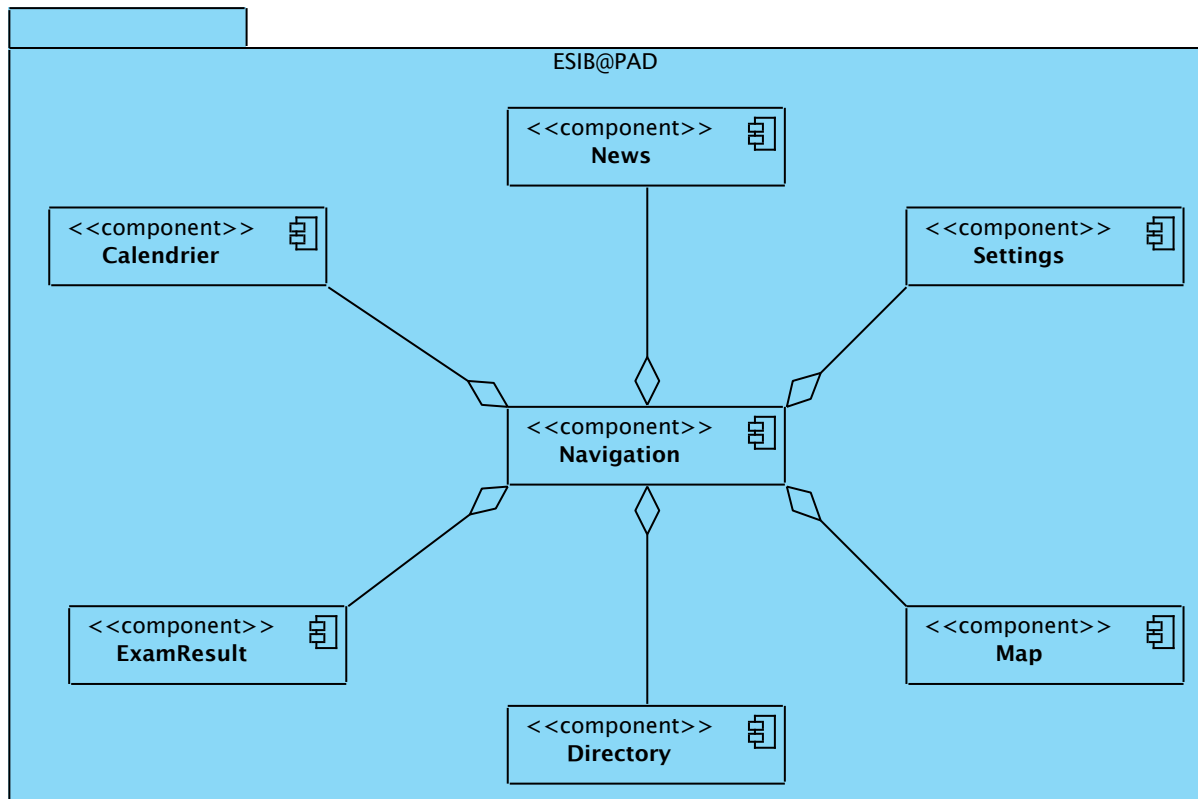


FIGURE 2: Diagrammes de composant du système

L'application est découpée en composant pour ainsi permettre de bien séparer les tâches que l'on désire offrir, faciliter la réutilisation de partie de l'application et rendre les tests plus efficace vu que l'on se concentre sur une partie et non pas un tout.

Les composants n'ont pas été développés complètement dans le cadre de l'art vu qu'on n'a pas pour chaque composant un fichier binaire qui permet sa réutilisation. Cette entrave à la règle est due au manque de connaissance dans la création de composant sur la plateforme iOS. Mais cependant le code est organisé et conçu de manière à faciliter sa réutilisation dans un autre cadre.

2.3.1 Description des composants

Voici une brève description des composants, le détail concernant leur implémentation se trouve dans le chapitre suivant.

Navigation :

Ce composant se charge de présenter un menu avec des boutons pour accéder aux composants de l'application. Lors d'un clic sur un de ses boutons, il est présenté à l'endroit approprié. Chaque composant appelé fera appel au composant "Navigation" pour être déchargé de la vue principale.

Map :

Ce composant affiche une carte avec des indications sur les divers lieux de l'université. Il permet aussi de chercher l'emplacement d'une personnes ou d'un bureaux.

Settings :

Ce composant permet de configurer les différents paramètres de l'application.

News :

Ce composant permet d'afficher les news de l'université. Le détail des news est aussi affichable sous forme de page web intégré dans l'application et contenant le détail tel qu'il se trouve sur le site internet de l'USJ.

Calendrier :

Ce composant permet d'afficher pour chaque membre de l'université son emploi du temps.

ExamResult :

Ce composant permet aux étudiants d'afficher les résultats des examens

Directory

Ce composant permet d'afficher l'annuaire de l'université. Il offre la possibilité d'envoyer des emails ou de lancer des appels à partir de l'application.

3 Conception et Implémentation des composants

3.1 Système de cache

:TODO: Describe the cache phiylsofy

3.2 Base de donnée

:TODO: Core data layer

3.3 Base technique du développement iOS

:TODO: Describe delegate, etc.

3.4 Navigation

Diagramme de séquence

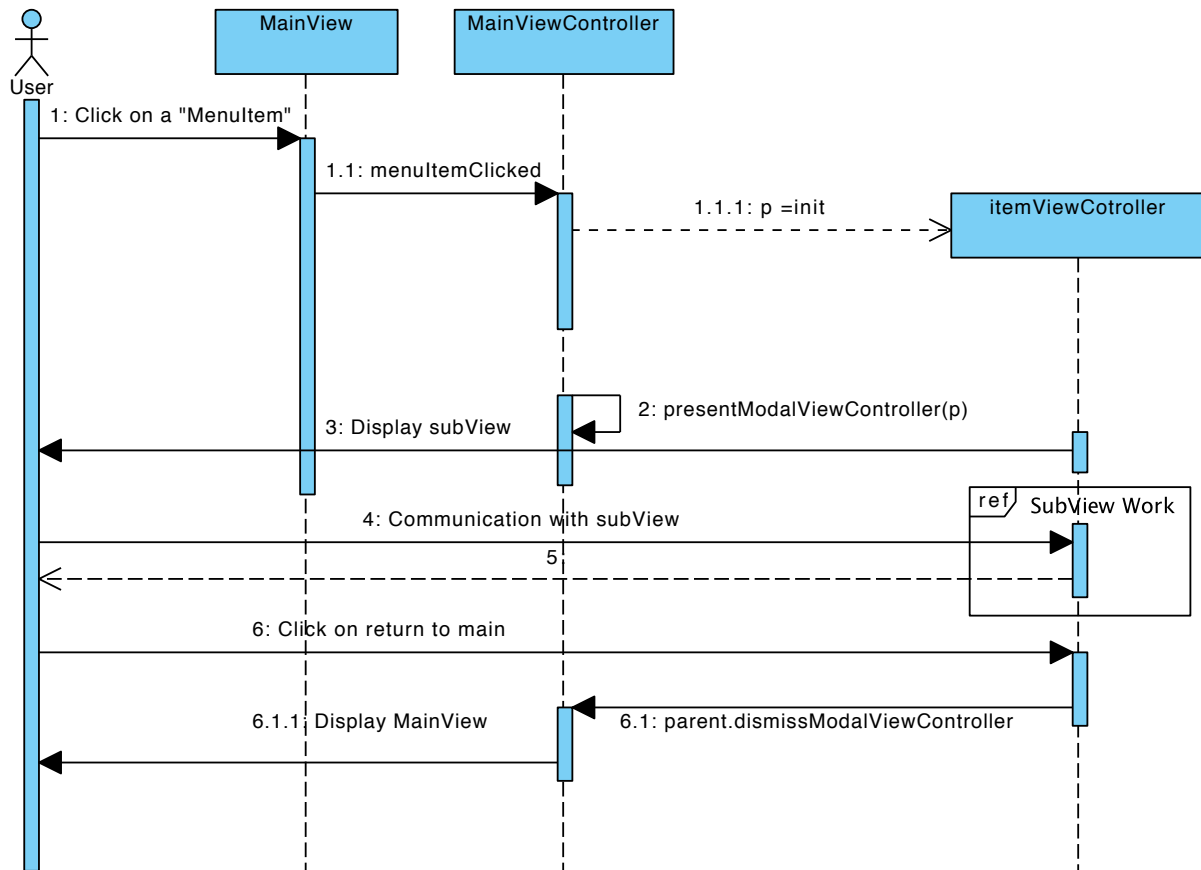


FIGURE 3: Diagramme de séquence du principe de la navigation

Le diagramme de séquence est valable pour les deux appareils la seule différence est que sur l'IPad la vue chargée ne cachera pas l'écran entier mais seulement une partie de l'écran.

Diagramme de classe

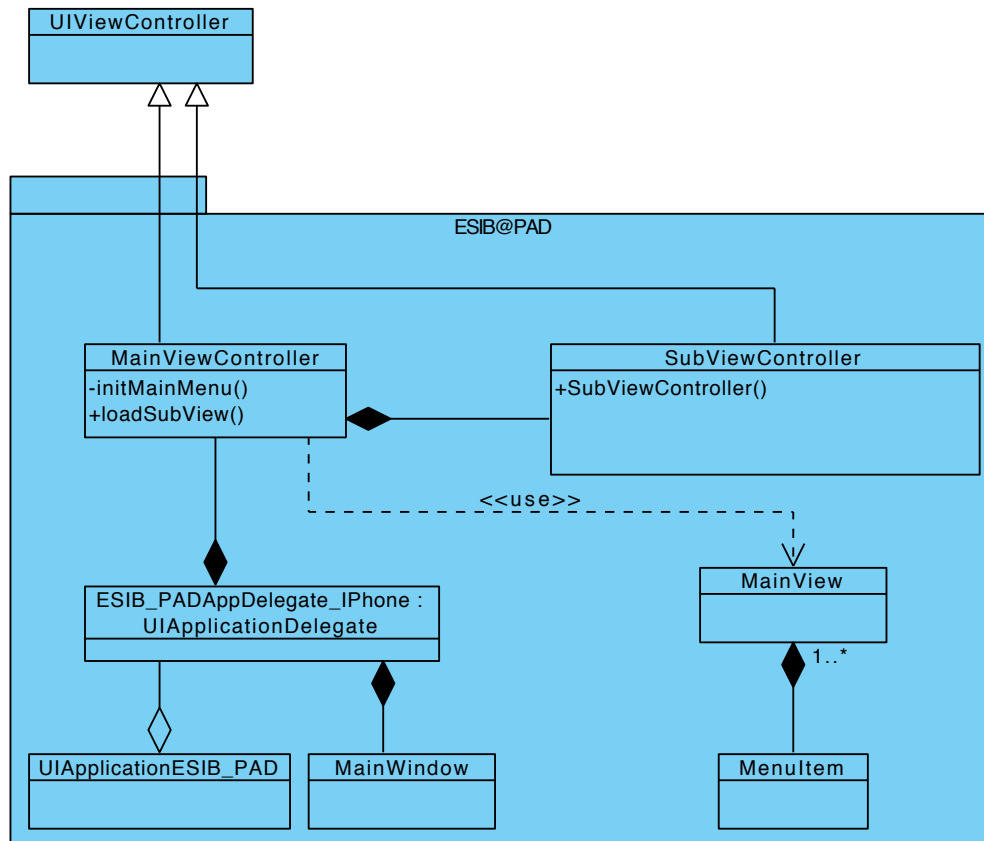


FIGURE 4: Diagramme de classe du composant MainView

Discussion

Pour faciliter l'ajout ou suppression d'éléments dans le menu, ce dernier est créé automatiquement à partir d'un fichier plist (Fichier xml de configuration pour l'iOS).

HeightElement	Number	120
HeightElementIPad	Number	70
▼ ItemLogo	Diction...	(7 items)
About	String	Information.png
Calendar	String	plan.png
Directory	String	Directory.png
Examination	String	exam.png
Map	String	Maps.png
News	String	news.png
Settings	String	SystemPreferences.png
▼ ListItem	Array	(7 items)
Item 0	String	News
Item 1	String	Map
Item 2	String	Directory
Item 3	String	Calendar
Item 4	String	Examination
Item 5	String	Settings
Item 6	String	About
WidthElement	Number	100

FIGURE 5: Contenu du fichier plist de configuration du menu pour la navigation

L'accès en lecture et écriture aux données des fichier plist se fait très facilement comme ceci :

```
// Writing value in Plist file
-(void)setValueForKey:(NSString *) theKey value:(NSString *) value {
    NSArray *paths = NSSearchPathForDirectoriesInDomains( NSDocumentDirectory,
                                                         NSUserDomainMask, YES);
    NSString *path =[[paths objectAtIndex: 0] stringByAppendingPathComponent: @"PlistFile.plist"];
    NSMutableDictionary * plistDict = [[NSMutableDictionary alloc] initWithContentsOfFile:path];
    if (! plistDict){
        plistDict = [[NSMutableDictionary alloc] init];
    }
    [plistDict setValue:value forKey:theKey];
    [plistDict writeToFile:path atomically: YES];
    [plistDict release];
}

// Reading value form Plist file
-(NSString *)getValueForKey:(NSString *) theKey {
    NSString *docsDir = [NSSearchPathForDirectoriesInDomains(NSDocumentDirectory,
                                                             NSUserDomainMask, YES) objectAtIndex:0];
    NSString *path = [docsDir stringByAppendingPathComponent: @"PlistFile.plist"];
    NSMutableDictionary* plistDict = [[NSMutableDictionary alloc] initWithContentsOfFile:path];
    NSString * d = [plistDict valueForKey:theKey];
    [plistDict autorelease];
    return d;
}
```

Listing 1: Code d'écriture et de lecture dans un fichier plist.

Suite à des problèmes d'affichage rencontré lors de la rotation des appareils, il a été décidé de centraliser la gestion de rotation des appareils dans cette partie de l'application. Pour se faire on va s'enregistrer pour recevoir les notifications de rotation et après chaque rotation, on va après chaque rotation redessiner l'interface en fonction de l'orientation. Une fois l'orientation de l'appareil détecté on va forcer le système à redessiner la vue comme on le désire.

```

//Registring for notification
[[NSNotificationCenter defaultCenter] addObserver:self selector:@selector(didRotate:) name:@"
    UIDeviceOrientationDidChangeNotification" object:nil];

- (void) didRotate:(NSNotification *) notification
{
    UIInterfaceOrientation currentOrientation = [[UIDevice currentDevice] orientation];

    // Important: Somme times, the current device orientation is Unknown and then the only othe
    // way to nows the orientation is the variable self.interfaceOrientation
    if (currentOrientation == UIDeviceOrientationUnknown ||
        currentOrientation == UIDeviceOrientationFaceUp ||
        currentOrientation == UIDeviceOrientationFaceDown){
        currentOrientation = self.interfaceOrientation;
    }
    if( UIDeviceOrientationIsLandscape(currentOrientation)){
        // Devise is in landscape redraw view for this orientation
    }else{
        // Devise is in portrait redraw view for this orientation
    }
}

```

Listing 2: Code d'enregistrement pour la notification de rotation des appareils.

3.5 Settings

Diagramme de séquence

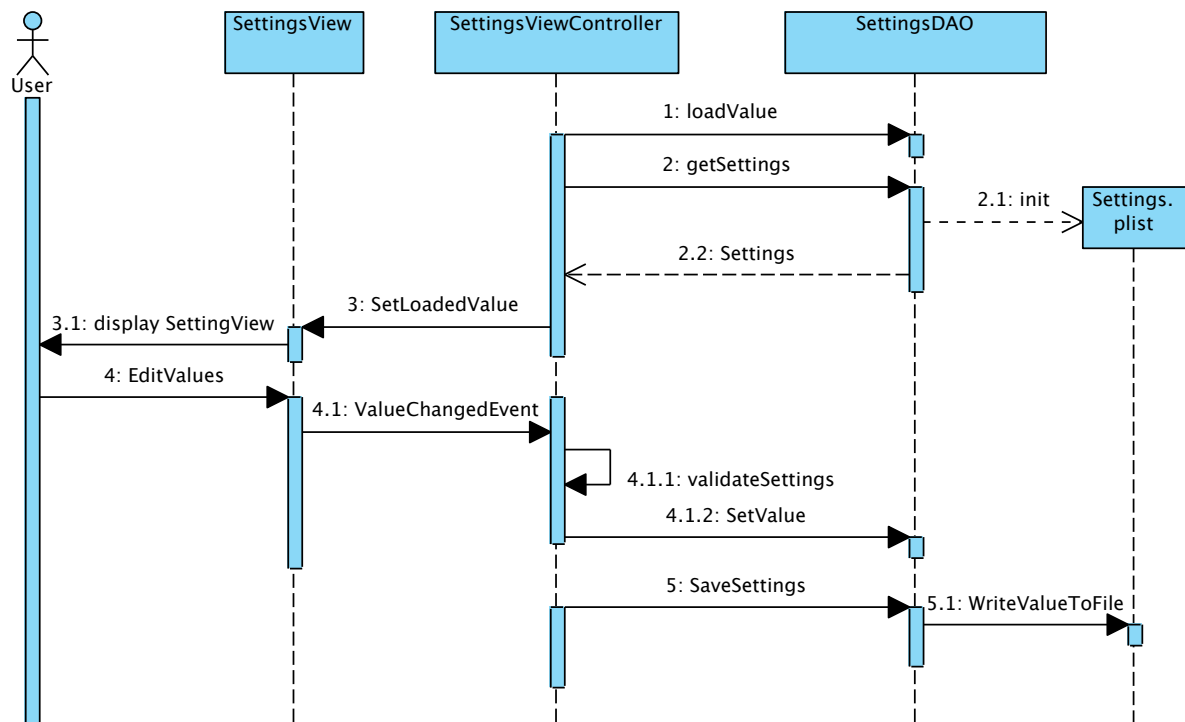
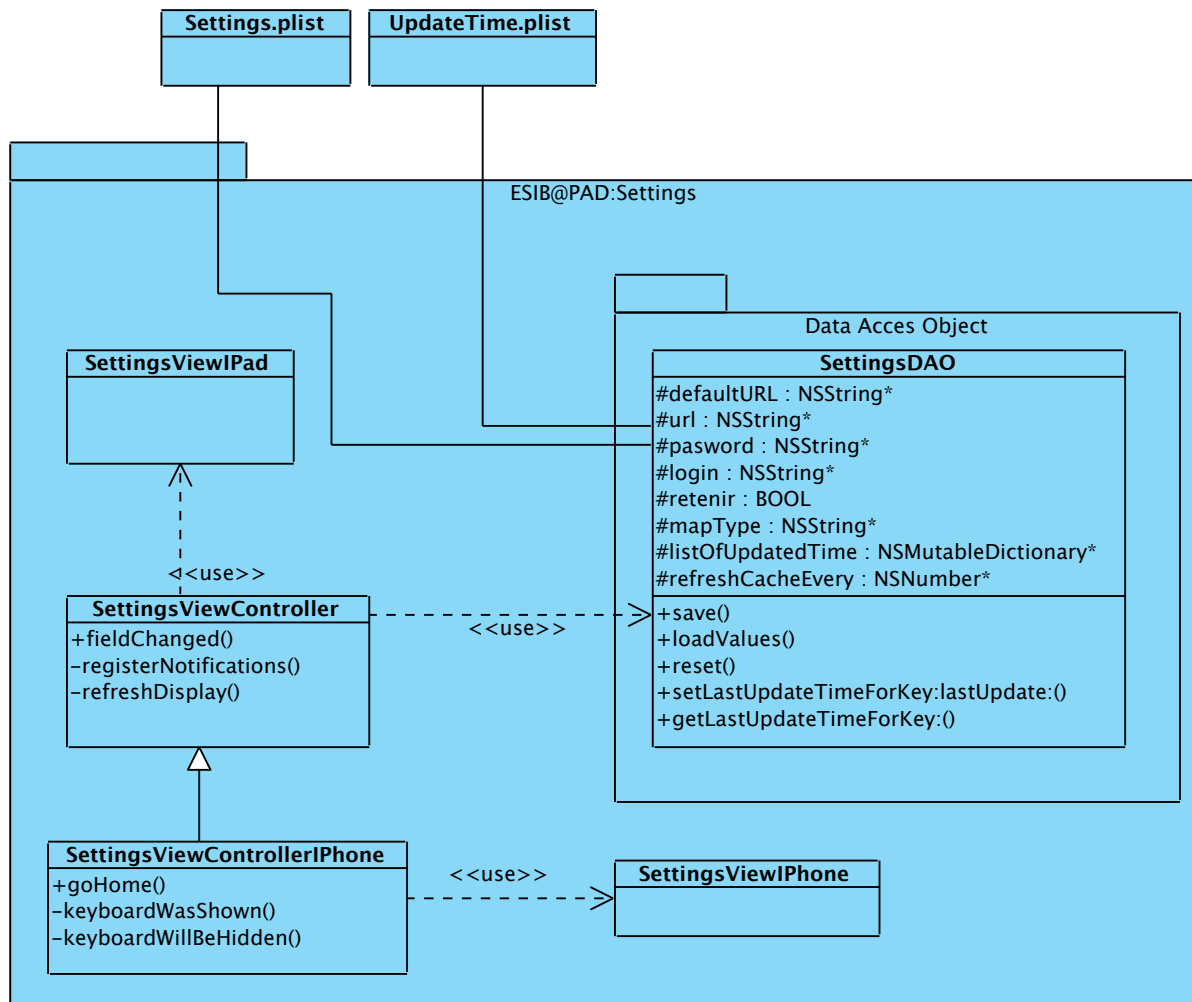


FIGURE 6: Diagramme de séquence concernant la lecture et la modification des paramètres



3.6 Map

Diagramme de séquence

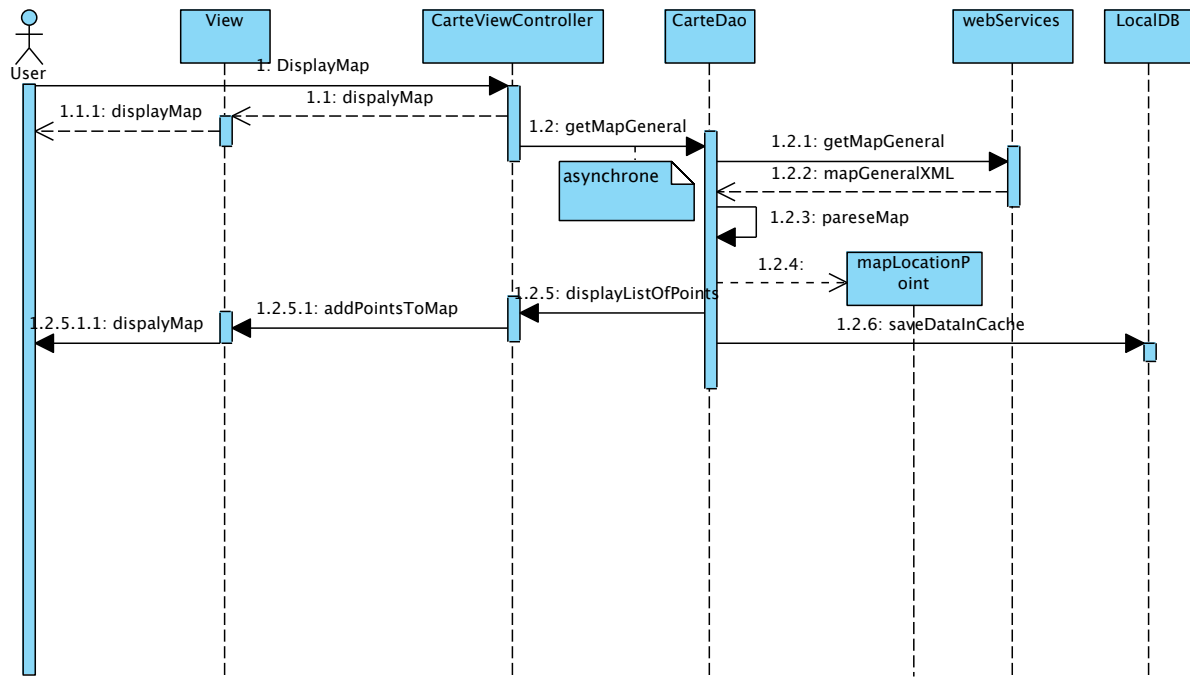


FIGURE 8: Exemple de séquence concernant l'affichage de la carte

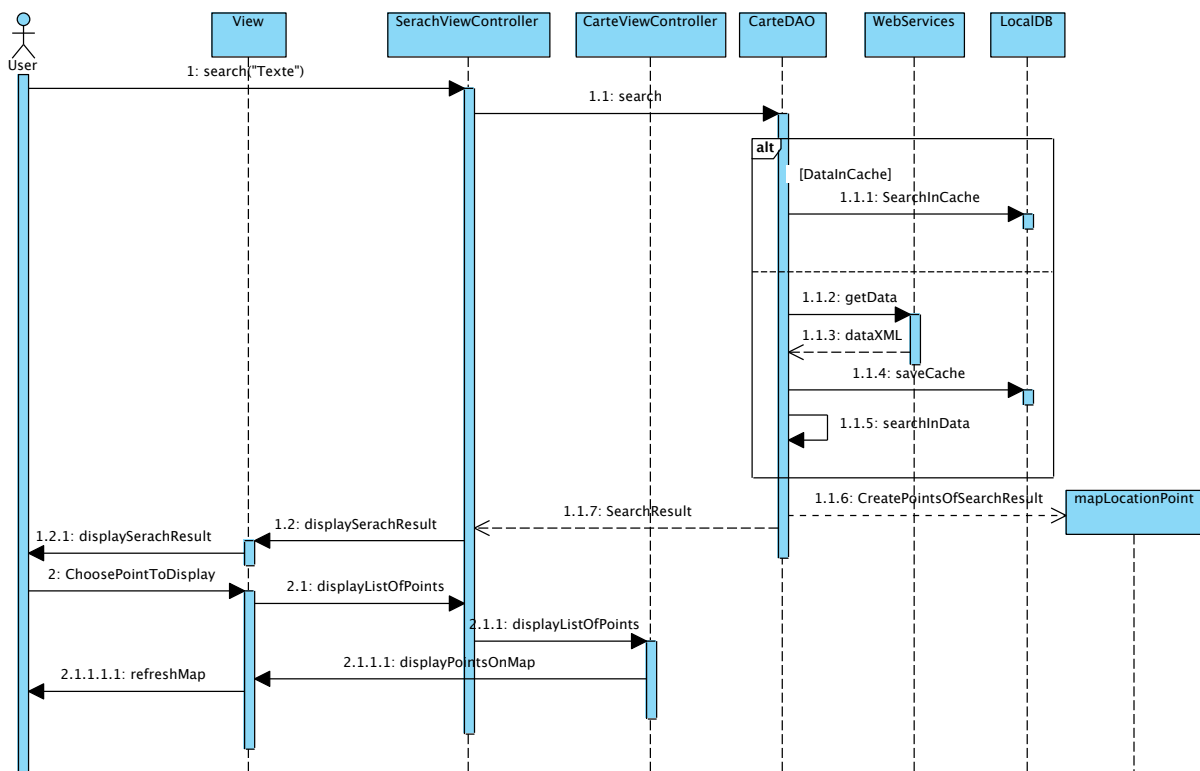


FIGURE 9: Exemple de séquence concernant la recherche d'un élément sur la carte

Le diagramme de séquence est valable pour les deux appareils la seule différence est que sur l'iPad la vue chargée ne cachera pas l'écran entier mais seulement une partie de l'écran.

Diagramme de classe

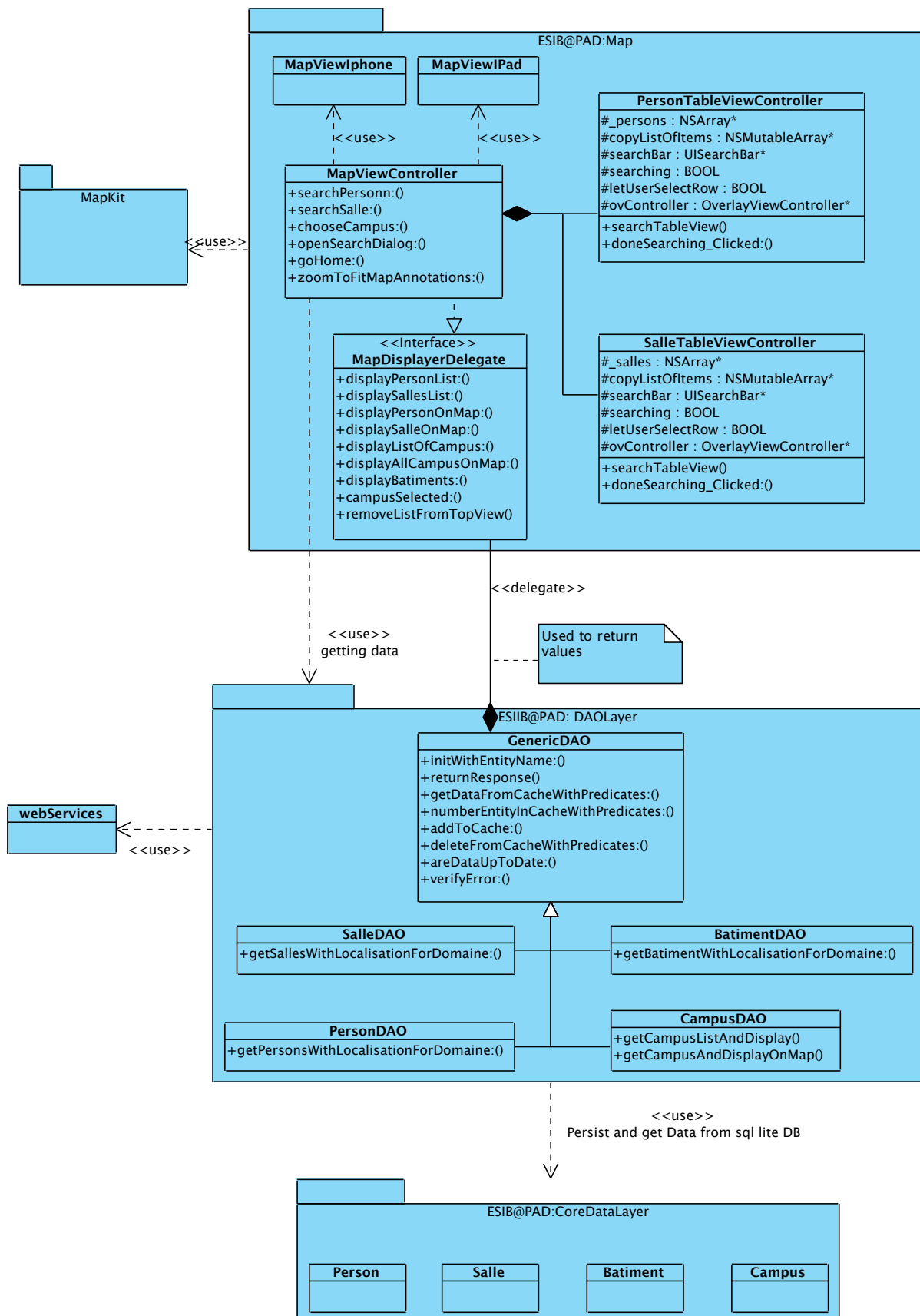


FIGURE 10: Diagramme de classe du composant Map

Discussion

Le framework MapKit qui exploite les images de googleMap est utilisé pour afficher la carte. Son utilisation est simple et l'excellent tutoriel à l'adresse <http://www.raywenderlich.com/2847/introduction-to-mapkit-on-ios-tutorial> a été un bon point de départ. Ce framework nous permet d'ajouter des indicateurs sur la carte pour signaler les emplacements intéressants.

```
MKMapView * map = [[MKMapView alloc] initWithFrame:self.view.frame];

CLLocationCoordinate2D coordinate;
coordinate.latitude = 35.000;
coordinate.longitude = 33.000;
MapLocations *annotation = [[[MapLocations alloc] initWithName:@"Un exemple d'annotation"
description:@"Voici une description" coordinate:coordinate] autorelease];
[map addAnnotation:annotation];
```

Listing 3: Code de création d'un objet MKMapView et l'ajout d'une annotation.

Il est tout à fait pensable si par la suite on obtient des cartes des campus plus détaillé de les intégrer aux cartes existante.

L'appel asynchrone nous permet de télécharger des données comme la liste de personnes ou l'emplacement des bâtiment depuis internet d'une manière transparente. Avec les appels asynchrone on évite que toute l'interface graphique soit gelé.

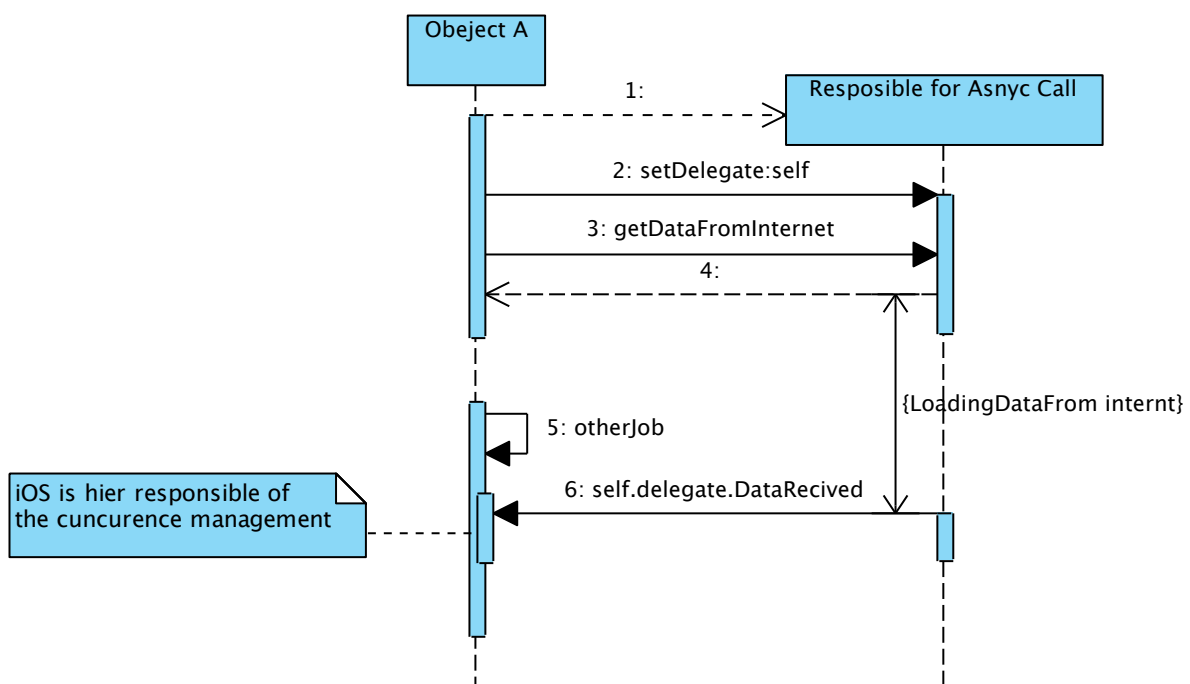


FIGURE 11: Diagramme de séquence illustrant l'appel asynchrone pour télécharger des données depuis internet

3.7 News

Diagramme de séquence

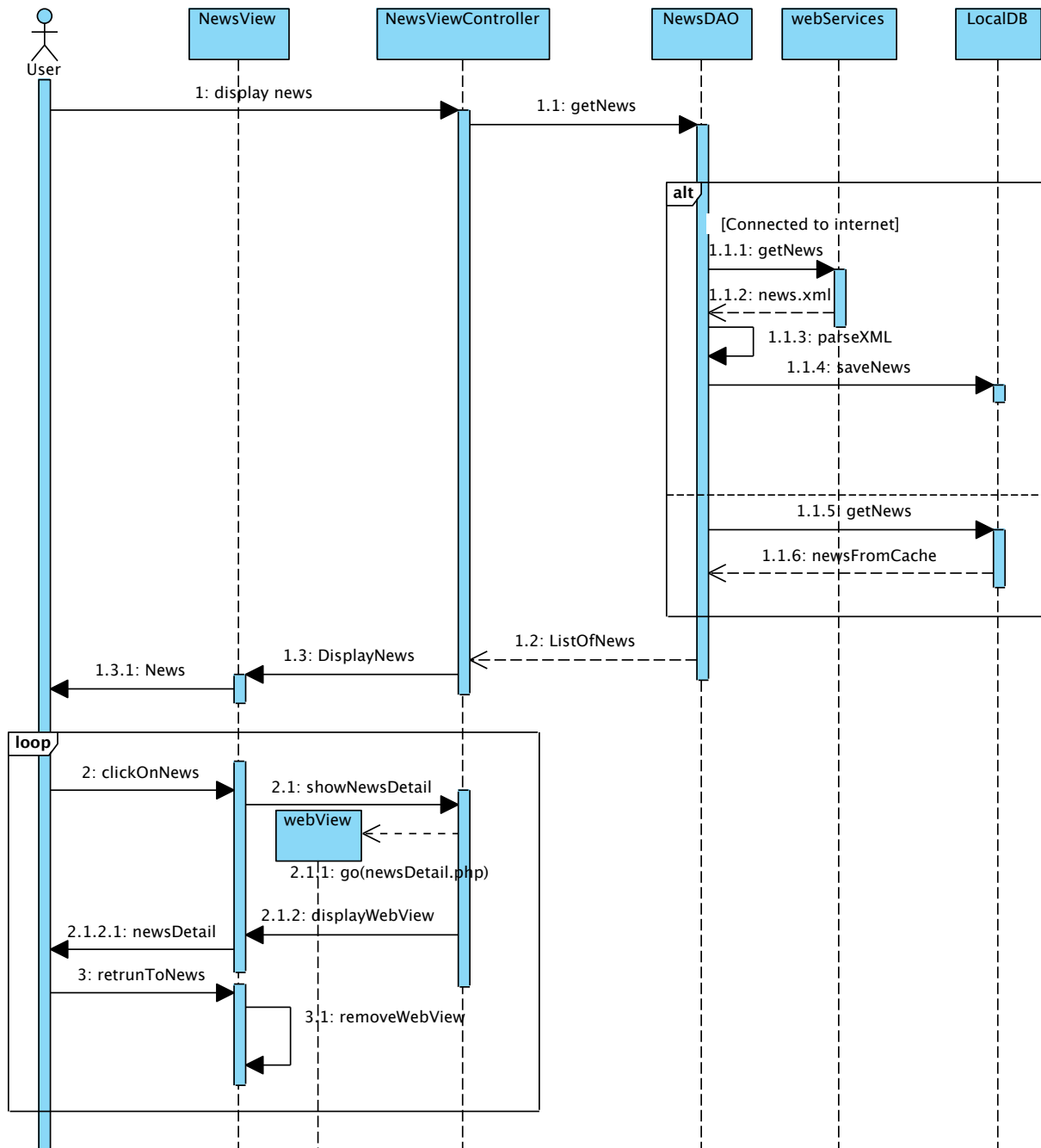


FIGURE 12: Exemple de séquence concernant l'affichage des news

Diagramme de classe

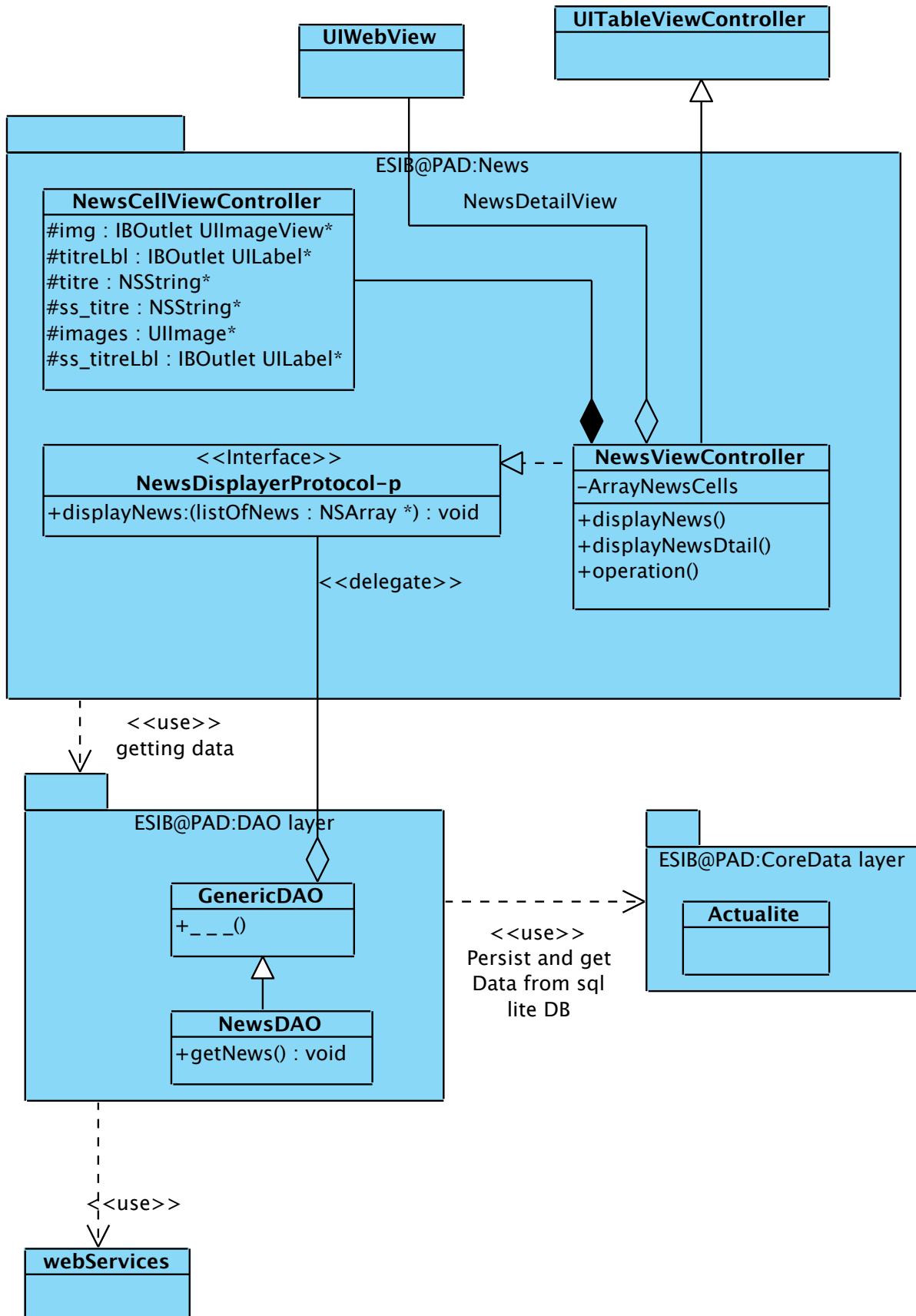


FIGURE 13: Diagramme de classe du composant News

Discussion

Le diagramme de séquence de la figure 12 nous montre que les news sont de tout façon téléchargé depuis internet même si elles sont déjà en cache. Ce choix est dû à la nature des données qui doivent être toujours à jour. Cependant si il n'y a pas de connexion internet, les informations en caches seront tout de même affiché.

Personnalisation des cellules d'un tableau : Affin de rendre le design graphique plus attrayant, les cellules du tableau ont été personnalisés. Il existe 2 principale façon pour modifier l'apparence des cellules :

1. La première consiste à modifier dans le code l'apparence avec des methodes tell que set Background, setColor. Cette méthode a un désavantage qui devoir compiler après chaque modification pour voir le résultat. De plus pour chaque cellule les mêmes opération seront refaites.
2. La deuxième solution est de crée un fichier NIB³ à l'aide de l'outil graphique (Interface builder) inclus dans X-Code. Les avantages ici sont que l'on peut visuellement voir le résultat et on a une très grande liberté. De plus les objet sont directement stocké sous format binaire dans l'application et on doit pas chaque cellule perdre des ressources à les redessiner. L'inconvénient est que cette manière de faire nécessite plus de connaissance technique.

La deuxième variante a été utilisé et ainsi il est possible de personnalisé rapidement l'apparence des news.

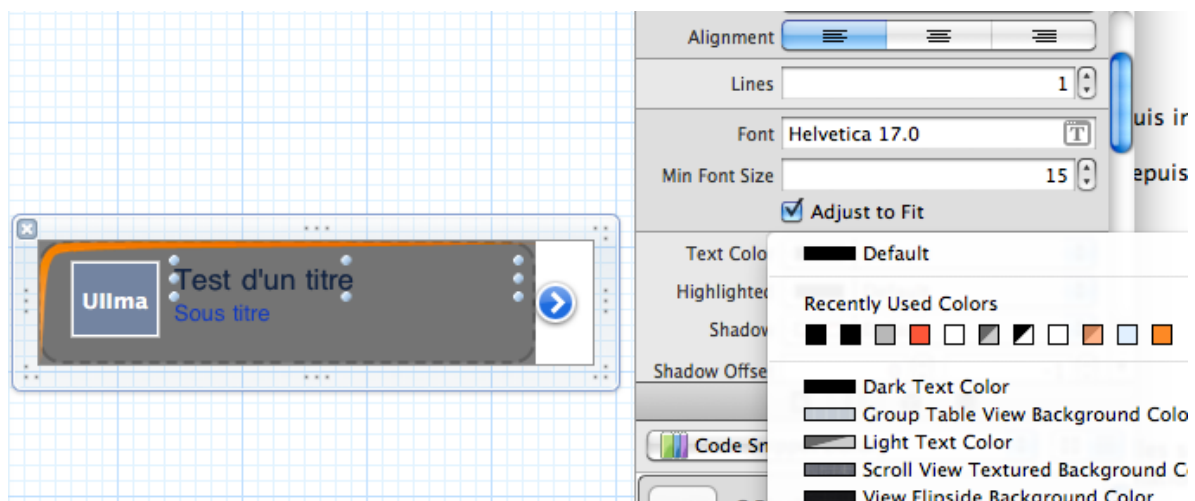


FIGURE 14: Illustration de la modification de l'apparence des cellules et plus précisément la couleur du texte du titre.

Téléchargement d'image à partir d'internt.

3. http://fr.wikipedia.org/wiki/Interface_Builder

