



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

Table des matières

1	Introduction	2
1.1	But du chapitre	2
1.2	Aperçu du chapitre	2
2	Architecture du système	2
2.1	Architecture choisie	2
2.2	Discussion des alternatives d'architectures	3
2.3	Composants du système	4
3	Conception et Implémentation des composants	6
3.1	Compatibilité graphique iPhone,iPad	6
3.2	Système de cache	7
3.3	Accès aux données des services web	8
3.4	Base de données	12
3.5	Navigation	14
3.6	Settings	18
3.7	Map	20
3.8	News	24
3.9	Directory	28
3.10	Calendrier	33
3.11	ExamResult	36

1 Introduction

1.1 But du chapitre

Ce chapitre décrit les variantes d'architectures étudiées 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 chapitre

La section 2 contient l'architecture de notre système ainsi que les différentes alternatives possible.

La section 3 contient la description détaillée de l'implémentation de chaque composant du système ainsi que le détail concernant les algorithmes et techniques utilisés pour réaliser les parties majeures du système.

2 Architecture du système

2.1 Architecture choisie

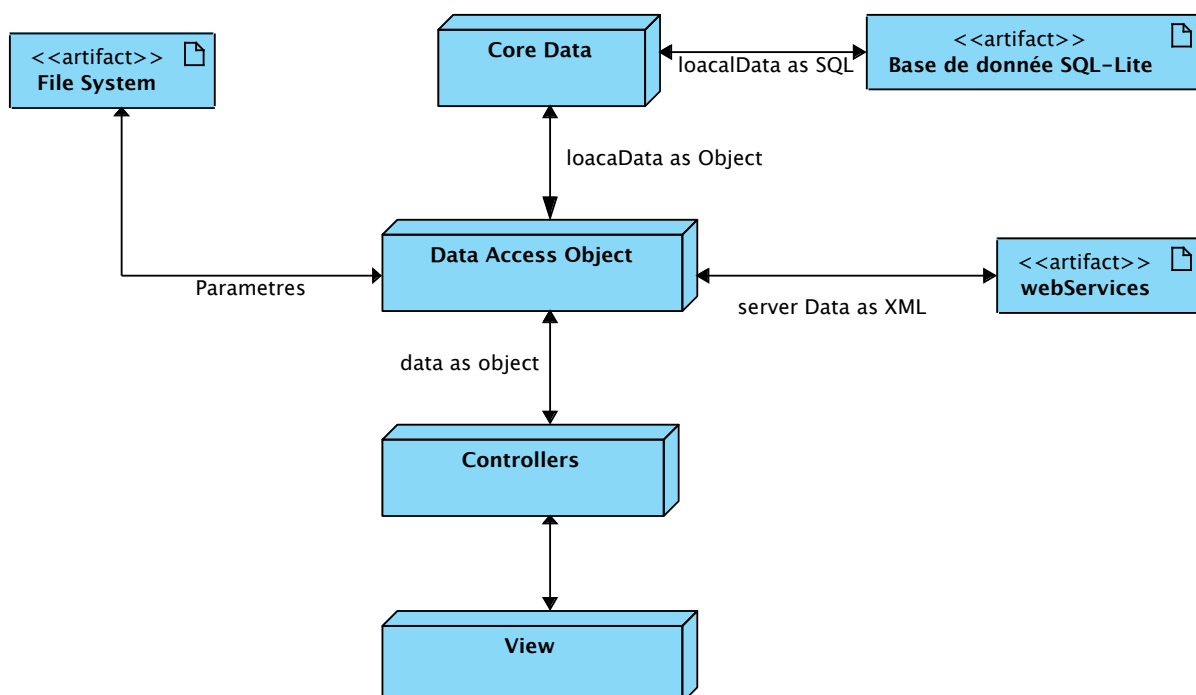


FIGURE 1: Vue global de l'architecture du système

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

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

Controllers : Cette couche du système s'occupe de charger les données dans la vue. Et de réagir correctement aux événements reçus de l'utilisateur.

Data Access Object : Cette couche fait référence au pattern de DAO [?] qui nous permet d'accéder aux données de notre application sans se soucier d'où elles proviennent, 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 aux données y sont centralisées.

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

2.2 Discussion des alternatives d'architectures

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

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 parser les fichiers xml à chaque utilisation des données. L'iOS n'est pas encore optimisée 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 diverses entreprises pour permettre le requêtage de fichiers XML, mais leurs performances restent tout de même moins bonne que celles d'une base de données.

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 on offre la possibilité de réutiliser ces mêmes objets sur d'autres plateformes (telle que Android). Après une brève 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 correspondants au contenu de la base de données nous avons mis de côté cette alternative. Par contre la base de données étant décrite en SQL-Lite, elle peut être exportée vers d'autres appareils et à partir de cette base de données il est facile de générer les objets de communication à l'aide des outils propres à chaque plateforme.

2.3 Composants du système

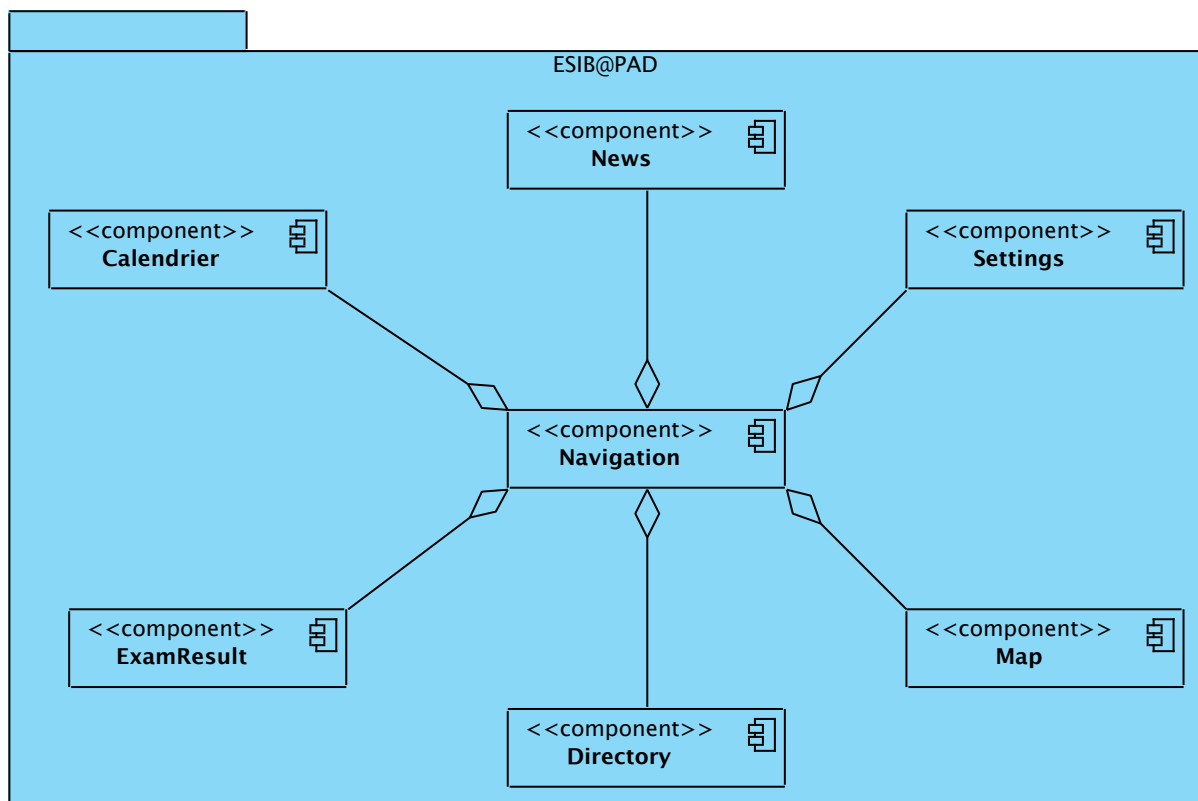


FIGURE 2: Diagrammes de composant du système

L'application est découpée en composants pour ainsi permettre de bien séparer les tâches que l'on est en train d'effectuer, faciliter la réutilisation de parties de l'application et rendre les tests plus efficaces vu que l'on se concentre sur une partie et non pas un tout.

Les composants n'ont pas été développés complètement dans les règles de l'art vu qu'on n'a pas pour chaque composant un fichier compiler qui permet sa réutilisation ainsi qu'une interface pour y accéder (système de black-box). Cette entrave à la règle est due au manque d'intérêt pour l'obtention de composants si sophistiqués vu qu'il ne seront pas réutilisés en dehors de notre application et pour des raisons d'économie de temps. Mais cependant

le code est organisé et conçu de manière à faciliter sa réutilisation dans un autre cadre et à pouvoir être transformé en composant complet(black-box) si besoin.

Description des composants

Voici une brève description des composants, le détail concernant leurs implémentations se trouve dans la section suivante.

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 des ces 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'une salle.

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 intégrée 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 l'emploi du temps de chaque membre de l'université.

ExamResult :

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

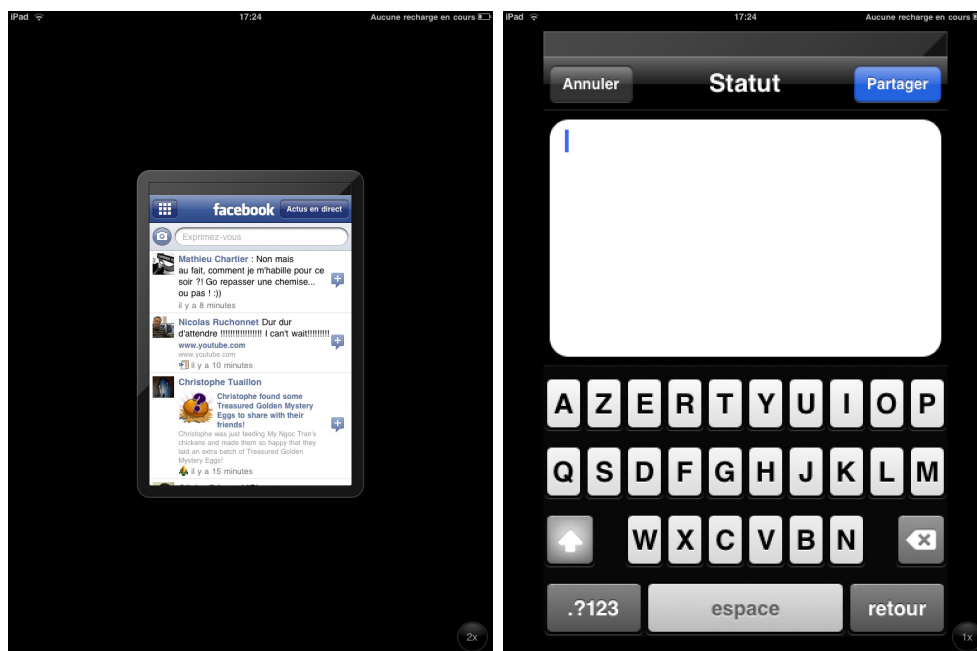
Directory

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

3 Conception et Implémentation des composants

3.1 Compatibilité graphique iPhone,iPad

Dès le lancement de l'iPad, Apple y a intégré un simulateur d'iPhone qui permet à toute application iPhone de s'exécuter sur l'iPad.



(a) Aperçu de l'application FaceBook version iPhone exécutée sur iPad (zoom 1x) (b) Aperçu de l'édition d'un texte d'une application iPhone sur l'iPad (zoom 2x)

Cette façon de faire peut être considérée comme une compatibilité, mais elle est minime vu qu'on étire simplement l'application iPhone pour la rendre plus grande mais on n'exploite nullement les capacités de l'écran de l'iPad.

A la création d'un projet à l'aide de XCode, ce dernier nous demande si l'on veut créer une application iPhone, iPad ou universel. En choisissant universel, on s'attend à pouvoir faire une application pour un appareil et qu'elle soit compatible avec les deux. Mais l'illusion de cette compatibilité apparaît assez vite. En effet pour la partie graphique il y a deux dossiers, un premier nommé iPhone et l'autre iPad. Et là on comprend que la partie graphique doit être faite en grande partie à double. Ceci ne signifie pas que tout doit

être fait à double, la plupart des composants ont leurs équivalents sur les deux appareils. Les deux composants uniquement disponibles sur iPad sont le composant Split views et Popovers. Dans une publication concernant[?] ce sujet Apple dit :

Conditional Coding

In order to achieve your design goals for a Universal application, you will need to use conditional coding to determine the availability of features when your app is running. Conditional coding allows you to make sure you're loading the right resources, using functionality that's supported by the device and properly leveraging hardware that's available.

Et en effet tout au long du développement et pour les éléments que l'on désire utiliser sur les deux appareils sans duplication du code, il a fallu tester pour savoir quel appareil exécute l'application et redimensionner les vues pour qu'elles soient à la bonne taille.

Cette incompatibilité peut s'expliquer par la différence de taille des écrans, iPad : 241.2 mm x 185.7 mm et iPhone : 115.2 mm x 58.6 mm. Et de plus l'information ne doit pas être organisée de la même façon sur les 2 appareils ; sur iPad on pourra facilement présenter un plus grand nombre d'informations sur un seul écran. Tandis que sur iPhone on doit essayer de minimiser l'information à afficher pour garantir qu'elle reste lisible.

3.2 Système de cache

Un des points importants pour notre application est la mise en cache des données. Cette importance est due à la vitesse de la connexion internet au Liban et de manière générale sur tous les appareils mobiles. Ce manque de capacité des appareils mobiles augmente le temps nécessaire pour accéder aux données. De plus, même de nos jours, il n'est pas possible d'accéder à Internet de partout.

Pour le stockage en cache, un mécanisme de sauvegarde des dates d'exécution des requêtes a été mis en place. Pour chaque requête que l'on fait au serveur, 3 variables sont envoyées : le nom et password de l'utilisateur ainsi que le contenu que l'on désire obtenir. Dès l'exécution d'une requête, la date de son exécution est enregistrée, à la prochaine exécution si le temps écoulé entre la dernière exécution et aujourd'hui est supérieur à X (actuellement défini à 30 jours¹) on exécute à nouveau la requête, sinon on récupère les valeurs du cache.

1. Valeurs modifiable dans le fichier Settings.plist du code source

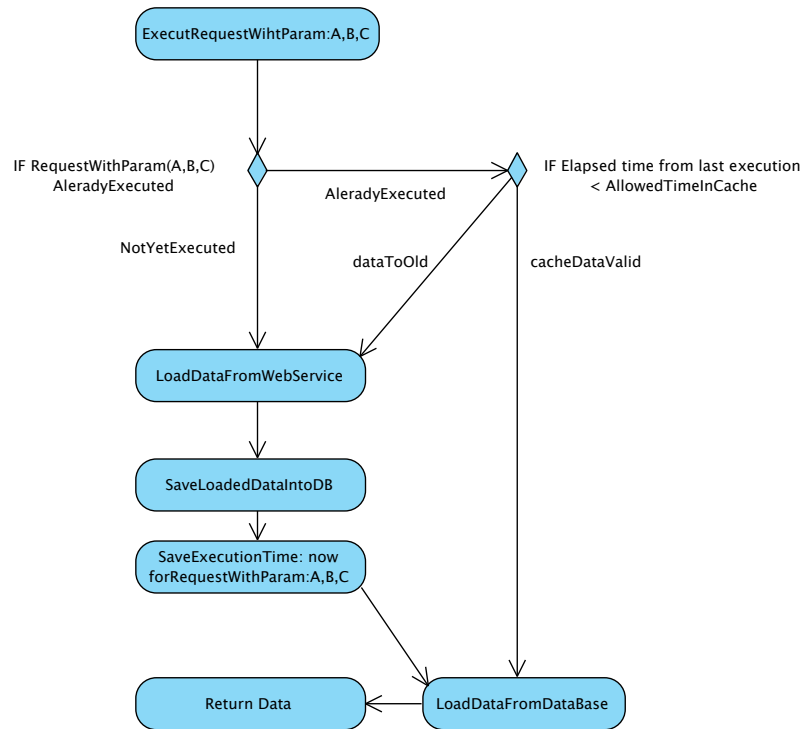


FIGURE 3: Illustration du fonctionnement du système de cache.

Pour plus de détails à ce sujet, voir le code source de la méthode `areDataUpToDate` de la classe `GenericDAO` ainsi que `setLastUpdateTimeForKey` et `getLastUpdateTimeForKey` de la classe `SettingsDAO`.

3.3 Accès aux données des services web

Le service informatique de l'université a mis en place un service web qui permet l'accès aux données via le protocole HTTP et il retourne essentiellement le contenu de la base de données sous format XML. Le service web se trouve actuellement à l'adresse : <http://www.usj.edu.lb/web-services/web-service.php>. Le paramétrage des services web se fait à l'aide de la méthode Post du protocole HTTP et notamment en transmettant les paramètres suivants :

1. **usr** le nom de l'utilisateur. `guest` est le nom par défaut et est valable pour les utilisateurs n'ayant pas de compte dans la base de données de l'USJ. Les autres utilisateurs doivent saisir leur id habituel pour les logins dans l'école.
2. **pwd** le mot de passe de l'utilisateur. `guest` est le mot de passe pour les invités
3. **op** est le nom de l'opération que l'on veut exécuter. Voici la liste des opérations possibles :

Nom de l'opération	Description	paramètres
listeServRec	Renvoie la liste des services de l'USJ.	aucun
listeCampus	Renvoie la liste des campus de l'USJ.	aucun
listeInst	Renvoie la liste des institutions de l'USJ.	param0 = Code campus (optionnel)
listeEmpNom	Recherche la liste des employés de l'USJ d'après le prénom et/ou le nom.	param0 = Nom(optionnel) param1= Prénom(optionnel). Au moins un paramètre est obligatoire
listeEmpInst	Renvoie la liste des employés d'une institution de l'USJ.	param0 = Code Institution(obligatoire)
listeEmpCampus	Renvoie la liste des employés d'un campus de l'USJ.	param0 = Code campus(obligatoire)
listeBatiments	Renvoie la liste des bâtiments d'un campus de l'USJ.	param0 = Code campus(obligatoire)
listeActualites	Renvoie les news de l'USJ.	aucun
listeHoraires	Renvoie l'horaire d'une personne selon les données de login.	Fonctionnalité uniquement implémentée sur les webServices en local.
listeNotes	Renvoie les résultats d'examen d'une personne selon les données de login.	Fonctionnalité uniquement implémentée sur les webServices en local.

TABLE 1: Liste des opérations possibles via les services web

Une page internet <http://www.usj.edu.lb/web-services/send.php> permet de saisir les paramètres et de les exécuter pour tester les services web.

Pour plus d'informations, concernant les web services, contacter M.Pascal TUFENKJ ,
Tel : +961 1 421 132 , Email : ptufenkji@usj.edu.lb

Services web locaux

Afin de minimiser la dépendance vis-à-vis des services web de l'USJ et de leur état d'avancement, des services web locaux ont été créés. Leur fonctionnement est très simple. Il s'agit

d'une page PHP hébergée sur la machine du développeur qui renvoie le fichier xml correspondant au nom de l'opération op.

Exemple : on envoie une requête http avec les paramètres en poste suivant : usr=Elias, pwd=1234,op=listeNotes. Le fichier xml listeNotes.xml qui est stocké sur la machine du développeur est retourné à l'utilisateur.

Pour plus d'informations, concernant les web services locaux voir le code source de la page webServices.php .

Téléchargement de fichier XML en Objective-c

Le téléchargement de données de grande taille se fait de manière asynchrone à l'aide de la classe `NSURLConnection`.

```
-(void)loadDataFromInternet{

    // Initialise the request with the server url.
    NSMutableURLRequest *request = [NSMutableURLRequest requestWithURL:[NSURL
    URLWithString:@"http://serveurAdress.com/webServices.php"] cachePolicy:
    NSURLRequestReloadIgnoringCacheData timeoutInterval:120.0];

    [request setHTTPMethod:@"POST"]; // Define the method of the request to Post

    NSString * postParam = [NSString stringWithFormat:@"usr=%@&pwd=%@&op=%@&param0=
    CST", set.login,set.pasword,@"listEmpCampus"];
    // Post parm for getting the list of person of the CST campus

    [request setHTTPBody:[webServicePostHeader dataUsingEncoding:NSUTF8StringEncoding]];
    // The delegate is self and self implement the
    NSURLConnection *connection = [[[NSURLConnection alloc] initWithRequest:request delegate:
    self] autorelease];
    if (connection) {
        receivedData = [[NSMutableData data] retain];
    }
}

// this method is called for recieving every 256 byte of data
- (void)connection:(NSURLConnection *)connection didReceiveData:(NSData *)data {
    [receivedData appendData:data];
}
// this method will be called at the end of the connection (all data recieved)
- (void)connectionDidFinishLoading:(NSURLConnection *)connection{
    // Parse recieved XML file.
}
```

Listing 1: Téléchargement d'un fichier XML depuis internet de manière asynchrone et en transmettant les paramètres de la requête par POST .

Exploiter des XML en Objective-c

L'exploitation de fichiers XML se fait à l'aide du parser de l'iOS qui nous donne l'accès à l'information par événement (SAX).

```
// receivedData is an NSData object with row data of an XML file.
NSXMLParser *parseur=[[NSXMLParser alloc] initWithData:receivedData];
// Self is realising the NSXMLParserDelegate protocol
[parseur setDelegate: self];
// For the example the xml file is :
// <root>
//     <row>
//         <variable1>the value of variable 1 </variable1>
//         <variable2>the value of variable 2 </variable2>
//     </row>
// </root>
if ([parseur parse] == NO){
    //Parsing error, mange it hier.
}
[parseur release];

//This method is called each time that an XML balise is opened
- (void)parser:(NSXMLParser *)parser didStartElement:(NSString *)elementName namespaceURI:(
    NSString *)namespaceURI qualifiedName:(NSString *)qName attributes:(NSDictionary *)
    attributeDict{
    if ([elementName isEqualToString:@"row"]){// Create new object for getting data
        crntObject = [[NSObject alloc] init]
    }
    self.crntElementName =elementName;
}

//This method is called each time that a string (balise content) is found
- (void)parser:(NSXMLParser *)parser foundCharacters:(NSString *)string{

    NSString *newString = [[NSString alloc] initWithFormat:@"%s", crntCharacters,string];
    self.crntCharacters = newString;
    [crntObject setValue:newString forKey:self.crntElementName];
    // We set the string value to the variable with the name of the current element
}

}

//This method is called each time that an XML balise is closed
- (void)parser:(NSXMLParser *)parser didEndElement:(NSString *)elementName namespaceURI:(
    NSString *)namespaceURI qualifiedName:(NSString *)qName{
    if ([elementName isEqualToString:@"row"]){// Close of the root element
        if (![crntObject save]) {
            NSLog(@"Whoops, couldn't save");
        }
    }
}

}
```

Listing 2: Parsing d'un fichier XML en Objective-c .

3.4 Base de données

Pour stocker les données reçues depuis les services web, une base de données SQL-Lite est utilisée. La base de données est complètement gérée par Core Data. La création des entités se fait dans XCode en éditant le fichier ESIB_PAD_SOURCES.xcdatamodeld

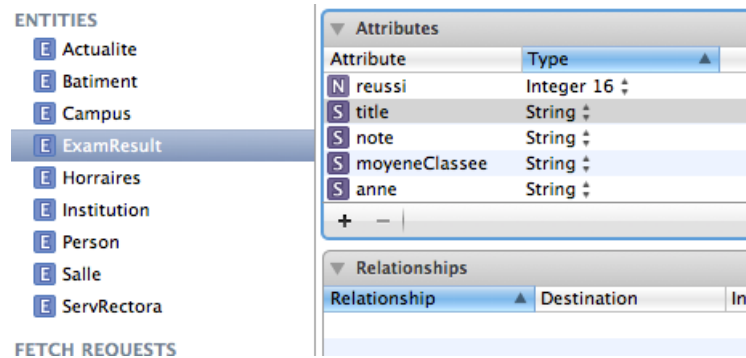


FIGURE 4: Interface dans X-Code permettant l'édition de la base de données

Une fois les entités créées et éditées, il est possible de générer les objets correspondants aux entités à l'aide de la manipulation suivante : Sélectionner les entités ⇒ Dans le menu : Editor ⇒ choisir Create NSMangedObject Subclass...

Une fois les objets correspondants aux entités générés, il est possible de les manipuler directement depuis le code. Dans notre application, la manipulation des données se fait dans la couche DAO du système.

```
//Getting the managedObjectContext
// It's important at the creation of the project to choose application with support of Core Data to
// get access to the managedObjectContext
NSManagedObjectContext * managedObjectContext = [(ESIB\_PAD\_SOURCESAppDelegate *)[[
    UIApplication sharedApplication] delegate] managedObjectContext];

// Example: creating a new Campus in the DB.
NSManagedObject newObject = [NSManagedObject alloc];
newObject = [NSEntityDescription insertNewObjectForEntityForName:@"Campus"
    inManagedObjectContext:managedObjectContext];
[newObject setValue:35.000 forKey:@"latitude"];
[newObject setValue:38.000 forKey:@"longitude"];
[newObject setValue:@"CST" forKey:@"code"];
[newObject setValue:@"Rue de la paix 12" forKey:@"adresse"];
// ...
// ...
[managedObjectContext save:&error]; // Persisting the new data.]

//Example: Reading list of person from campus CST
// The sql equivalent: SELECT * FROM Person where campus ='CST'
```

```
NSFetchRequest * crntRequest = [[NSFetchRequest alloc] init];
NSEntityDescription *entity = [NSEntityDescription entityForName:self.entityDescription
    inManagedObjectContext:@'Person'];
[crntRequest setPredicate:[NSPredicate predicateWithFormat:@"campus = CST" ]]; // Filtering
[crntRequest setEntity:entity];
NSError *error;
NSArray *items = [managedObjectContext executeFetchRequest:crntRequest error:&error];

for (Person *p in items) {
    NSLog('User name is: %@', p.name);
}
[crntRequest release];

//Example: Deleting the campus CST from the DB
// The sql equivalent: Delete * from campus where code ='CST';

NSFetchRequest * crntRequest = [[NSFetchRequest alloc] init];
NSEntityDescription *entity = [NSEntityDescription entityForName:self.entityDescription
    inManagedObjectContext:@'Campus'];
[crntRequest setPredicate:[NSPredicate predicateWithFormat:@"code = CST" ]]; // Filtering
[crntRequest setEntity:entity];
NSError *error;
NSArray *items = [self.managedObjectContext executeFetchRequest:crntRequest error:&error];

for (Person *p in items) {
    [managedObjectContext deleteObject:managedObject];
}
[managedObjectContext save:&error]; // Persisting the changement
[crntRequest release];
```

Listing 3: Exemple de création de lecture et de suppression de données dans la base à l'aide de Core Data .

3.5 Navigation

Diagramme de séquence

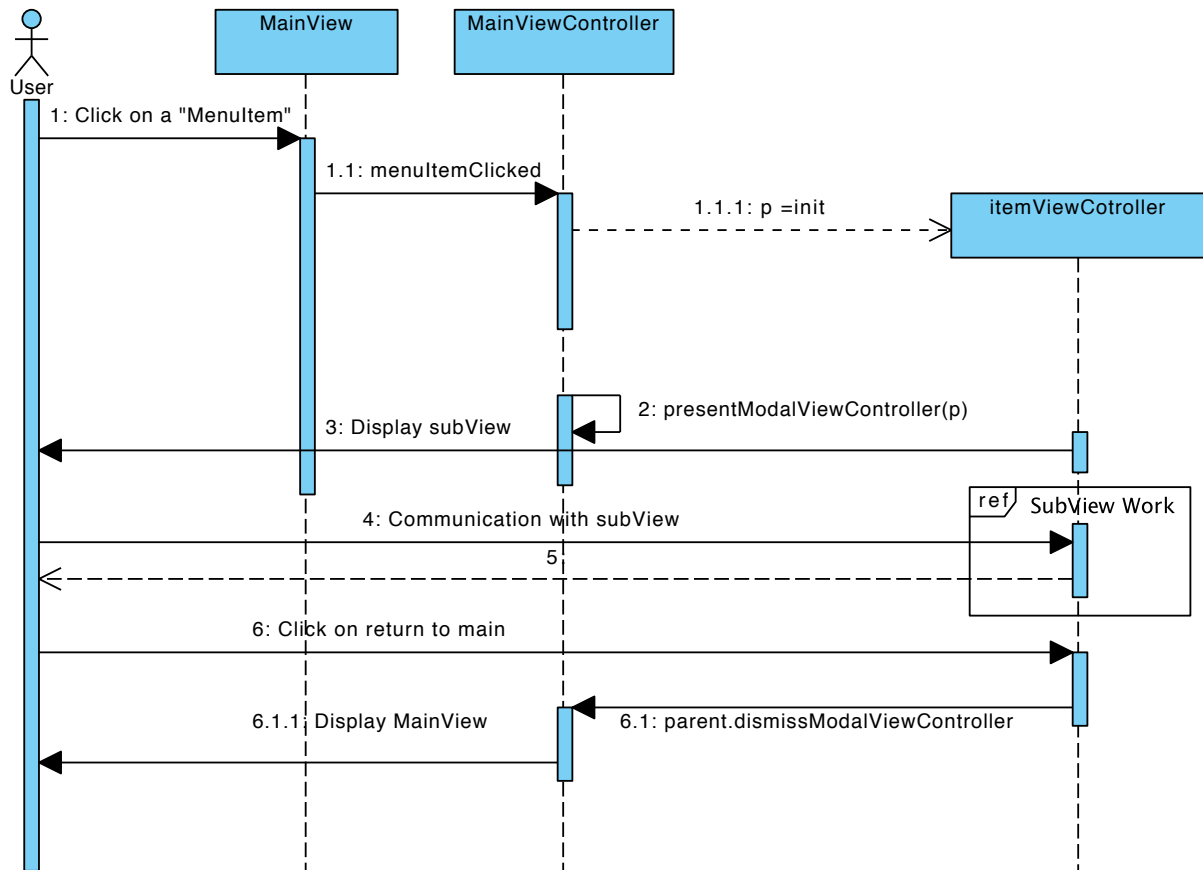


FIGURE 5: Diagramme de séquence du principe de la navigation

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 rien qu'une partie de l'écran.

Diagramme de classe

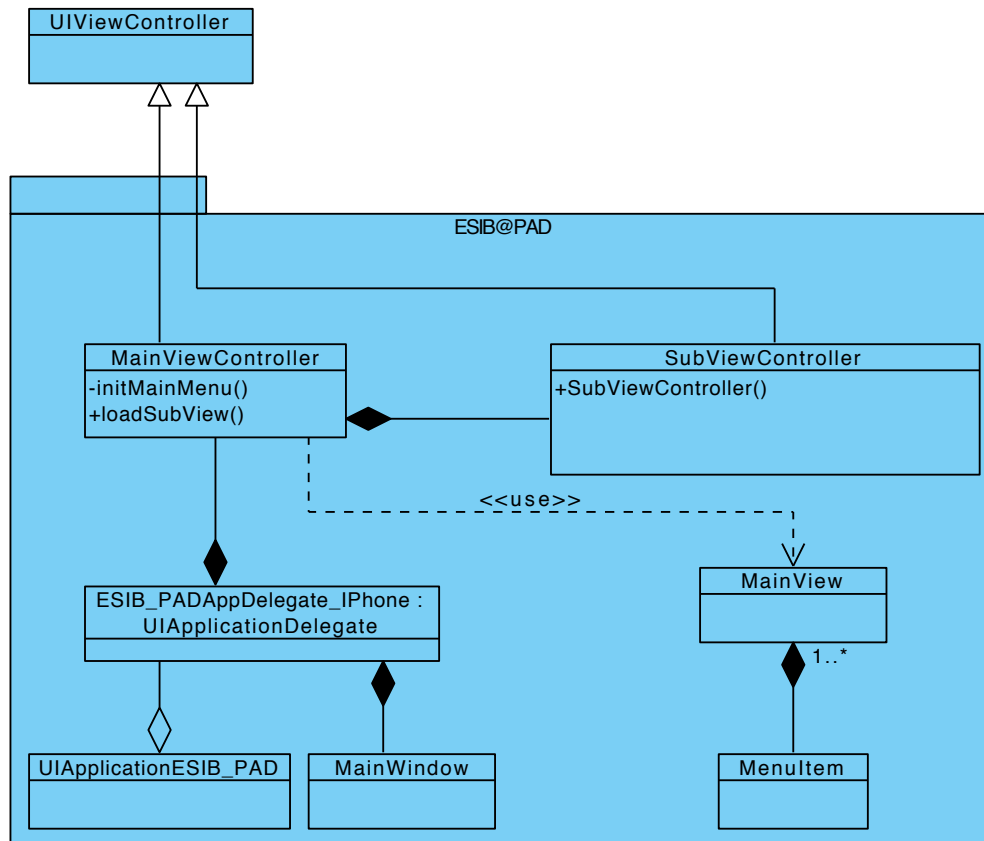


FIGURE 6: Diagramme de classe du composant MainView

Discussion

Pour faciliter l'ajout ou la 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	Dictionary (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 7: Contenu du fichier MenuItemsParam.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 4: Code d'écriture et de lecture dans un fichier plist.

Suite à des problèmes d’affichage rencontrés 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 ce faire, on va s’enregistrer pour recevoir les notifications de rotation et après chaque rotation, on va redessiner l’interface en fonction de l’orientation. Une fois l’orientation de l’appareil détectée, 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 5: Code d’enregistrement pour la notification de rotation des appareils.

3.6 Settings

Diagramme de séquence

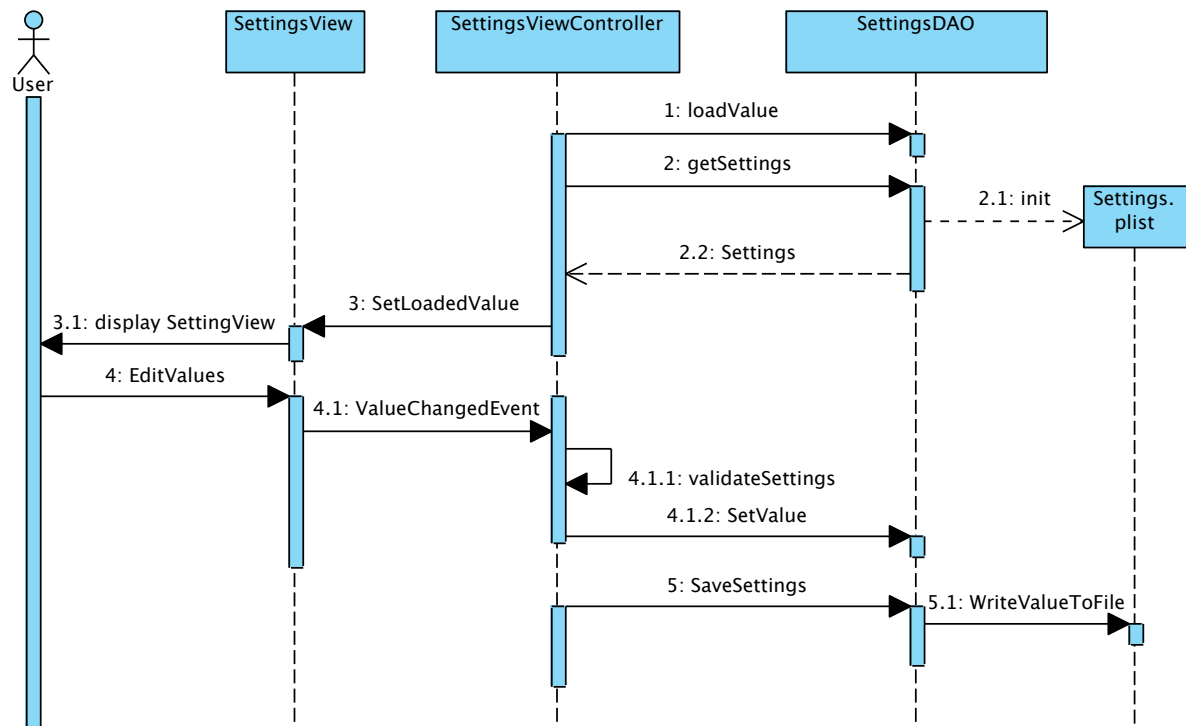


FIGURE 8: Diagramme de séquence concernant la lecture et la modification des paramètres

Diagramme de classe

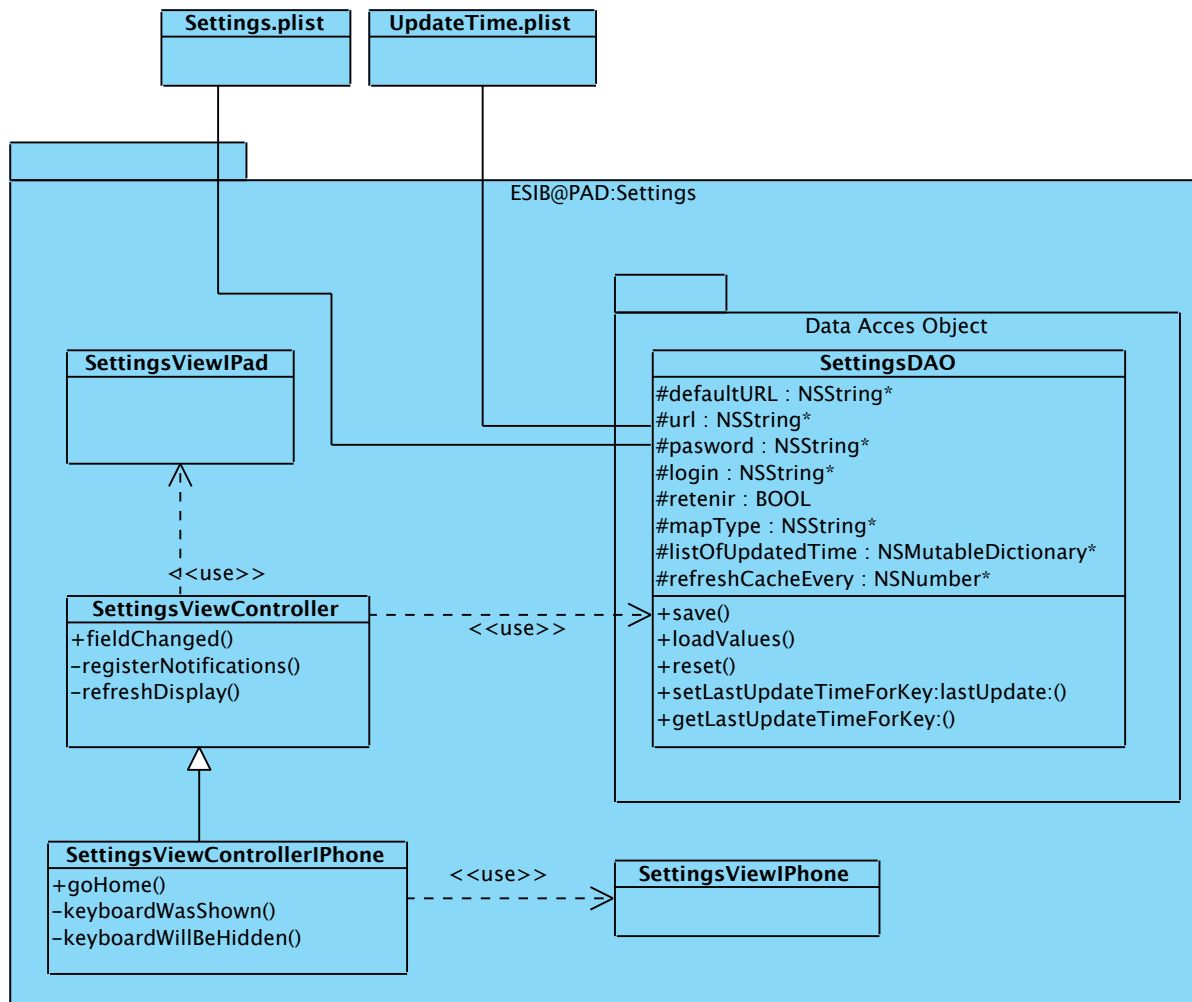


FIGURE 9: Diagramme de classe du composant Settings

Discussion

Appel propose un système relativement simple pour modifier les paramètres d'une application. Ce système est capable, à partir d'un fichier XML, de créer l'interface graphique pour modifier son contenu. Le système d'appel **n'a pas été utilisé car il oblige l'utilisateur à sortir de l'application** et d'aller dans la fenêtre de paramétrage du système d'exploitation pour modifier les paramètres de l'application.

Le détail ainsi que l'utilité concernant les fonctions `setLastUpdateTimeForKey` et `getLastUpdateTimeForKey` est expliqué dans le chapitre 3.2.

3.7 Map

Diagramme de séquence

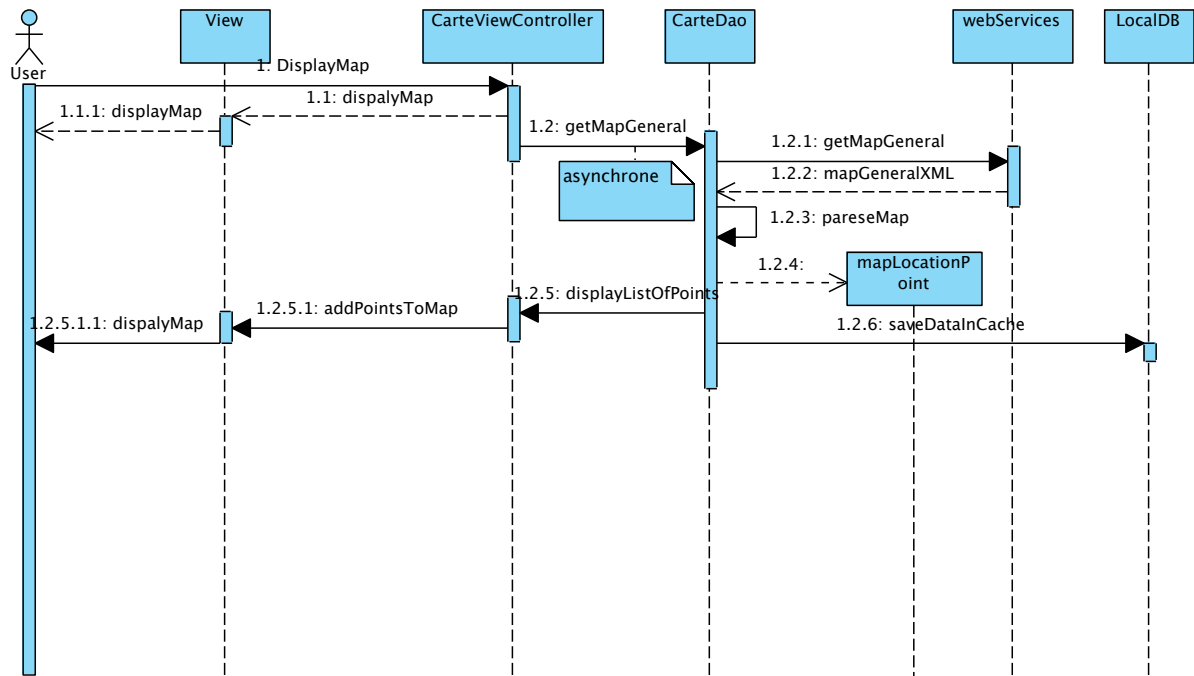


FIGURE 10: Exemple de séquence concernant l’affichage de la carte

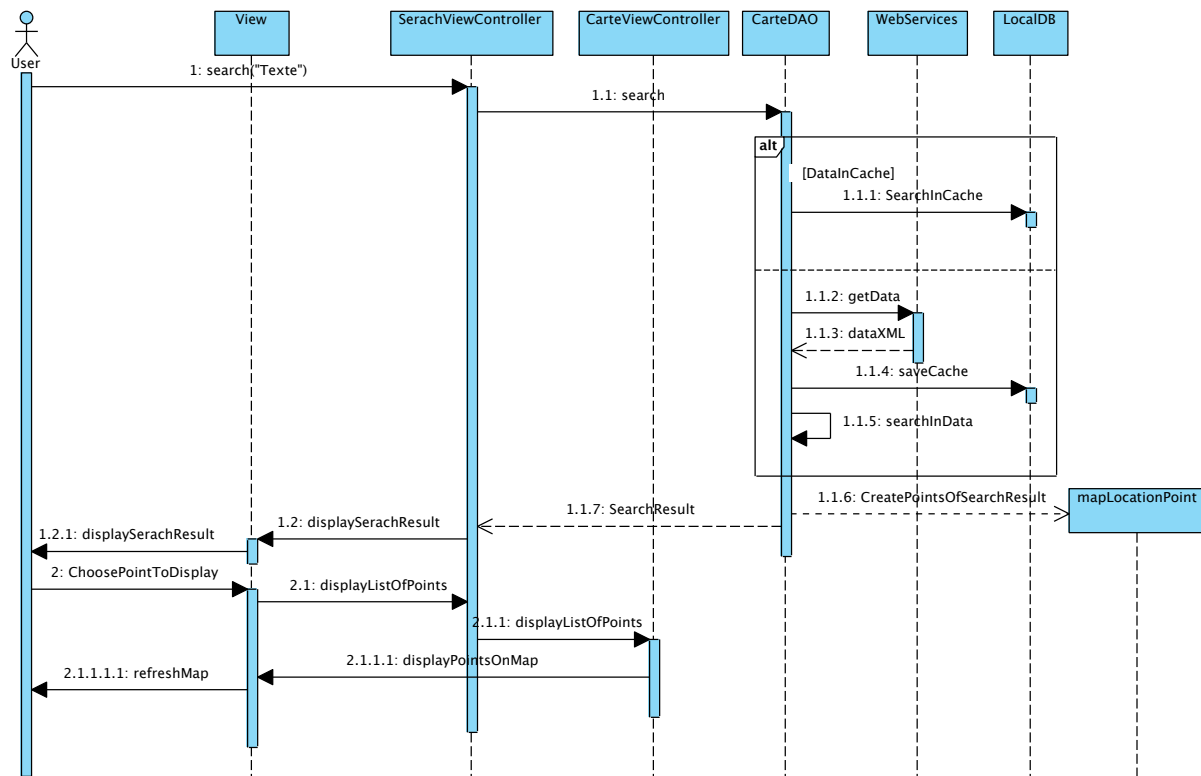


FIGURE 11: Exemple de séquence concernant la recherche d'un élément sur la carte

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 rien qu'une partie de l'écran.

Diagramme de classe

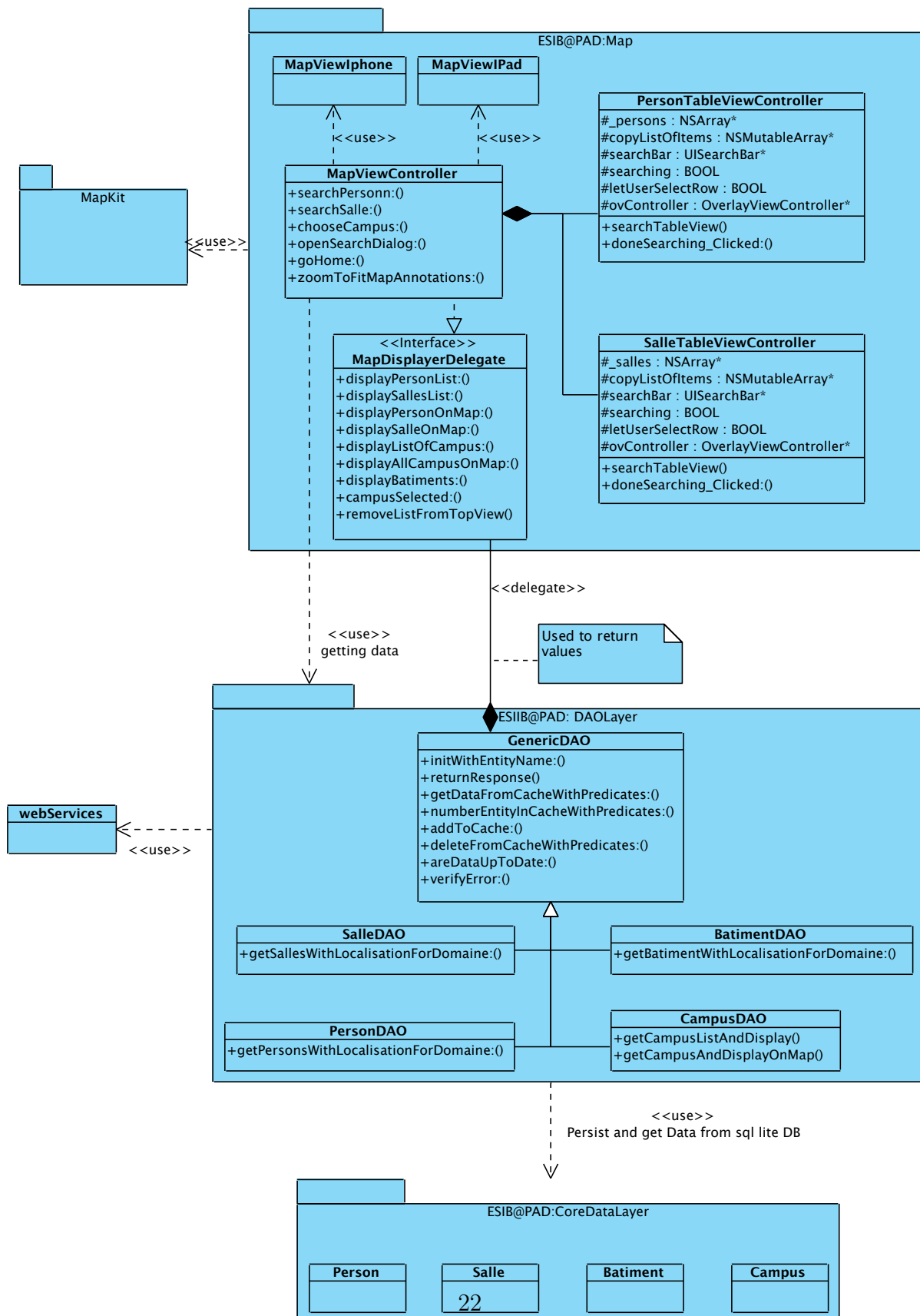


FIGURE 12: Diagramme de classe du composant Map

Discussion

Le framework MapKit qui exploite les images de googleMap est utilisé pour afficher la carte. Son utilisation est simple et possède un excellent tutoriel sur internet [?]. Il est très facile de démarrer avec cette librairie. 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 6: Code de création d'un objet MKMapView et l'ajout d'une annotation.

Il est tout à fait pensable d'intégrer par la suite des cartes des campus plus détaillées aux cartes existantes.

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

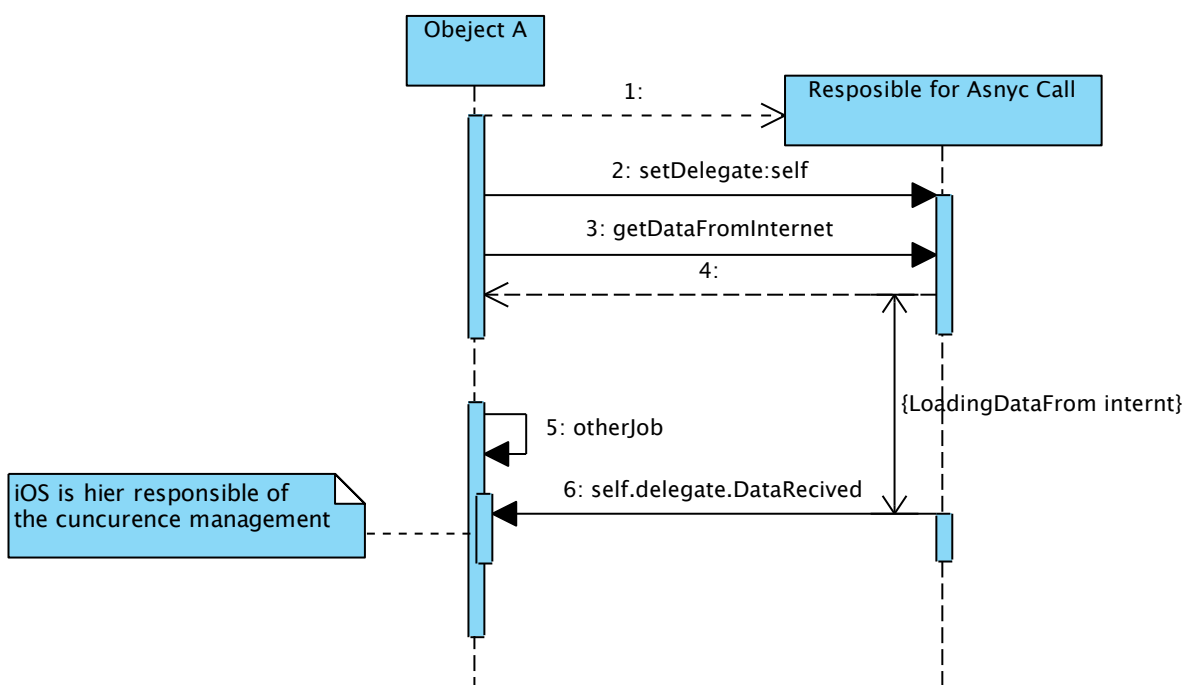


FIGURE 13: Diagramme de séquence illustrant l'appel asynchrone pour télécharger des données depuis internet

3.8 News

Diagramme de séquence

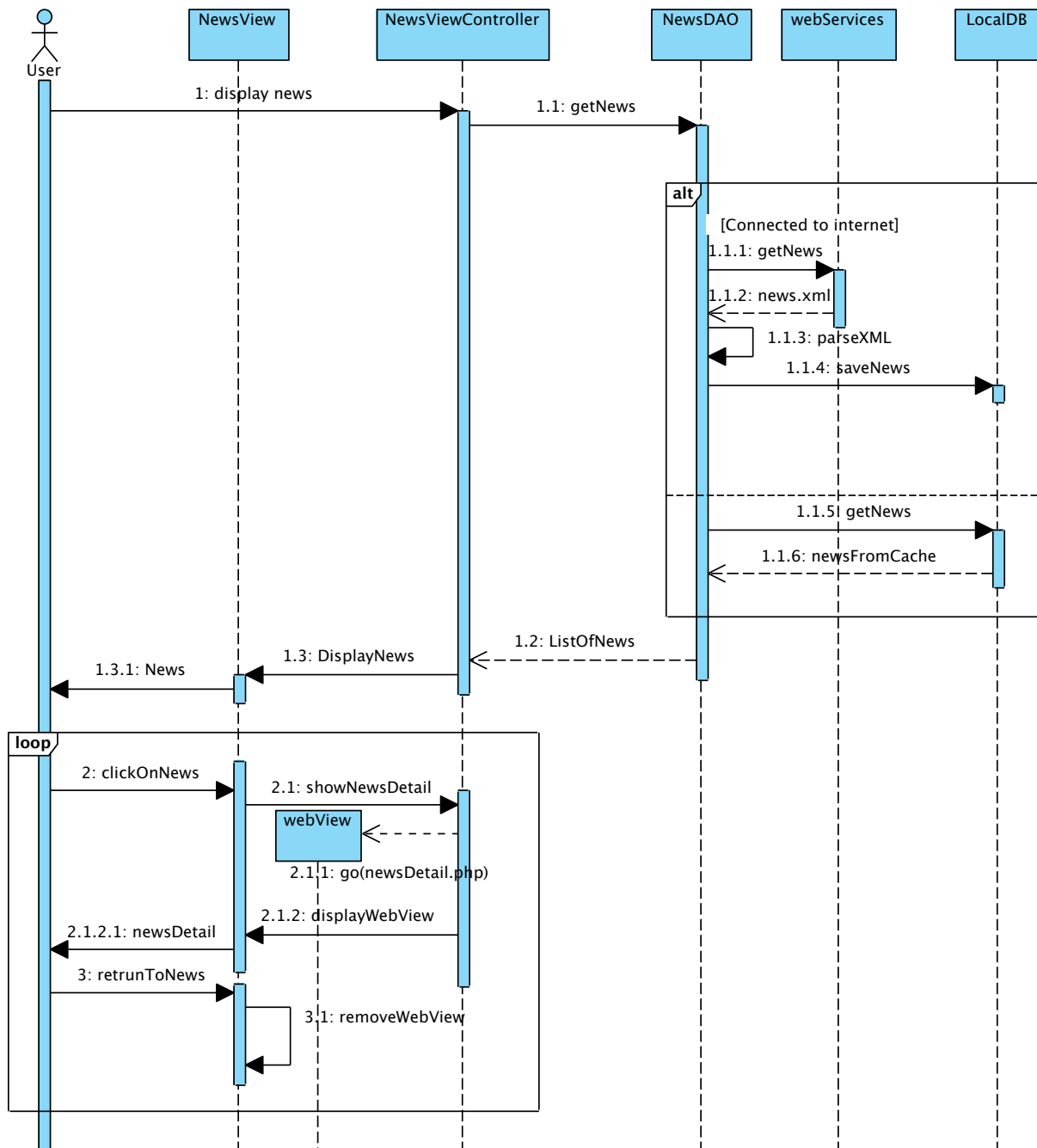


FIGURE 14: Exemple de séquence concernant l'affichage des news

Ce diagramme de séquence nous montre que les news sont de tout façon téléchargées 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 cache seront tout de même affichées.

Diagramme de classe

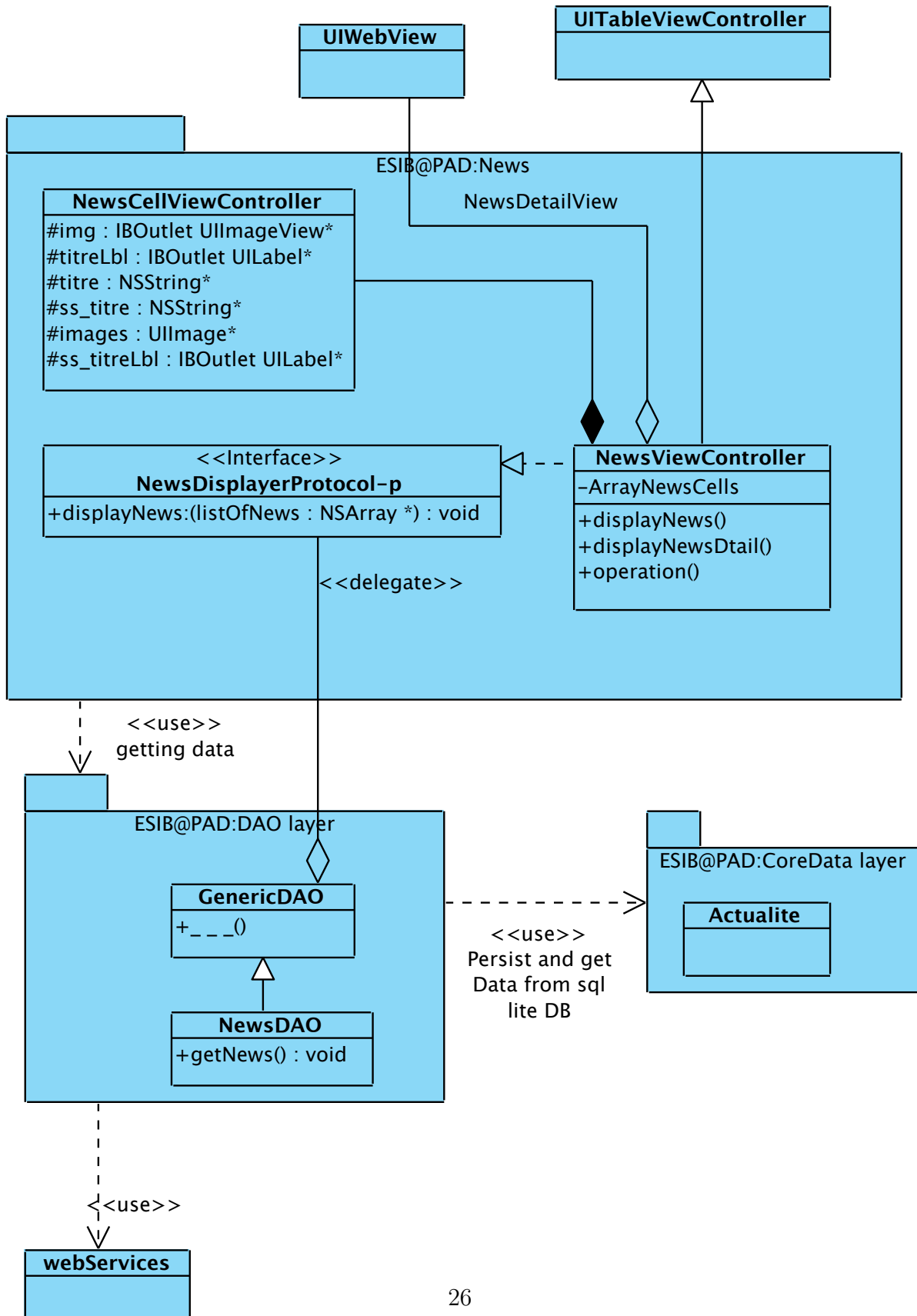


FIGURE 15: Diagramme de classe du composant News

Discussion

Personnalisation des cellules d'un tableau : Afin de rendre le design graphique plus attrayant, les cellules du tableau ont été personnalisées. Il existe 2 principales façons pour modifier l'apparence des cellules :

1. La première consiste à modifier dans le code l'apparence avec des méthodes telles que `setBackground`, `setColor`. Cette méthode a un désavantage qui est de devoir exécuter la modification après chaque modification pour voir le résultat. De plus, pour chaque cellule, les mêmes opérations seront refaites à chaque création.
2. La deuxième solution est de créer un fichier NIB² 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 d'expression. De plus, les objets sont directement stockés sous format binaire dans l'application et on ne doit pas, pour chaque cellule, les dessiner à nouveau. L'inconvénient est que cette manière de faire nécessite plus de connaissances techniques.

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

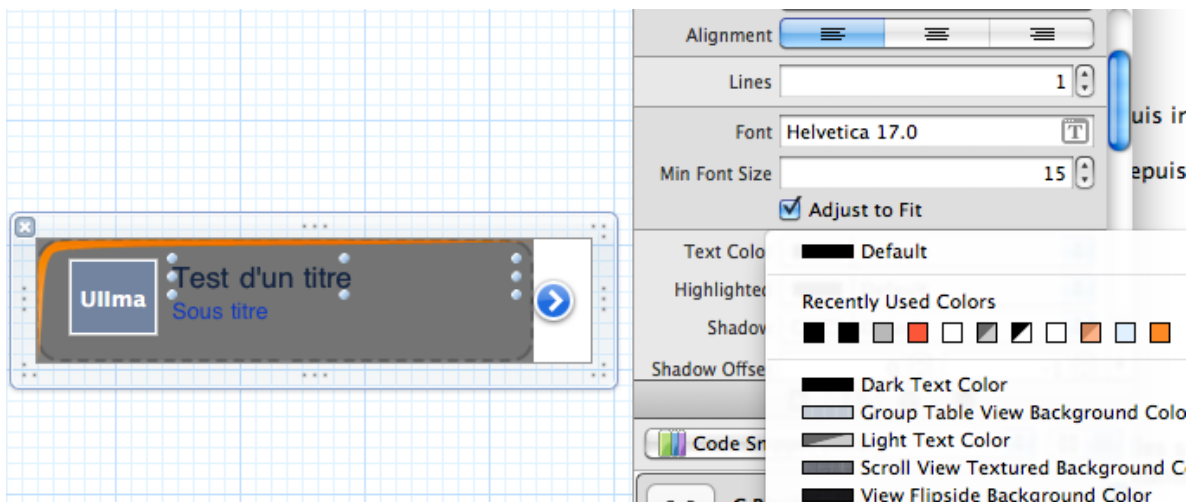


FIGURE 16: Illustration de la modification de l'apparence des cellules et plus précisément de la couleur du texte du titre.

Téléchargement d'image à partir d'Internet l'iOS offre la possibilité de charger des images depuis internet d'une manière simplifiée. Mais le téléchargement est fait d'une manière synchrone. Pour palier à ce problème nous pouvons utiliser la classe `NSURLRequest` pour télécharger les données brutes et les traiter en tant qu'image une fois toutes les données reçues. Pour plus de détails, voir le code source de la classe `AsyncImageView` dans le dossier `Utility`.

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

3.9 Directory

Diagramme de séquence

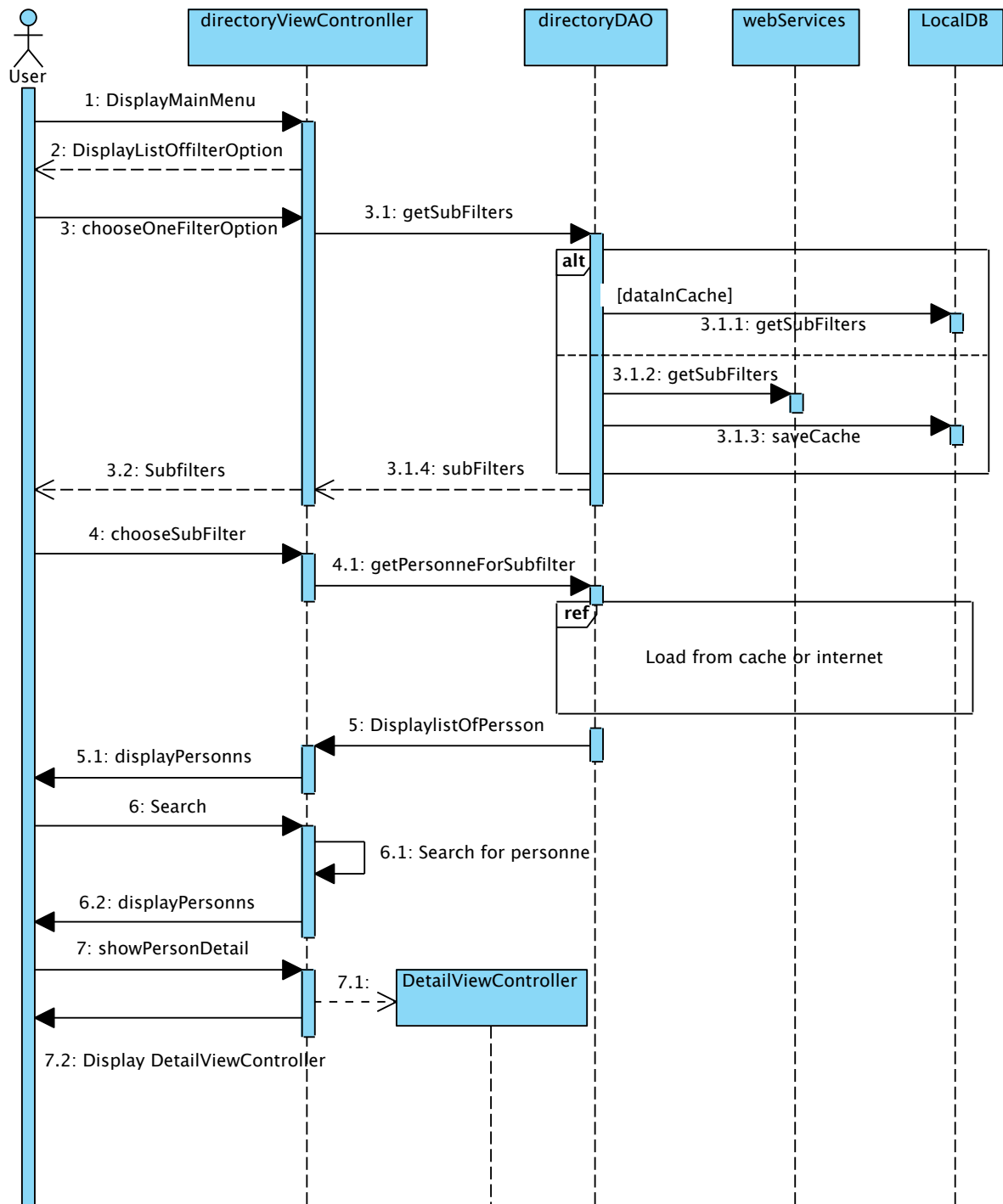


FIGURE 17: Exemple de séquence concernant choix d'un filtre d'affichage et ensuite l'affichage d'une personne de l'annuaire

de ce composant dans une partie de l'écran (et non en plein écran) n'est pas possible. De ce fait, une interface qui répond à nos besoins a été conçue. Cette interface nous permet d'avoir un élément à droite pour l'affichage de contenu et à gauche une zone réservée pour la navigation. Les effets de transition ont été faits à l'aide des fonctions d'animation de la classe UIView.

```
-(void) animateView:(UIView *) toAnim{
    // The first anim of the size to 50 x 50 and his position at x = 100 y =200;
    [UIView animateWithDuration:0.5 delay:0 options: UIViewAnimationCurveEaseOut
        animations:^(
            CGRect rect = CGRectMake(100, 200, 50, 50);
            toAnim.frame= rect;
        )
        completion:^(BOOL finished){
            NSLog(@"First anim finish")
        }];
    // The second anim of the alpha value to 0 (transparent)
    // The delay propriete help us to sync the animations
    [UIView animateWithDuration:0.5 delay:0.5 options: UIViewAnimationCurveEaseOut
        animations:^(
            toAnim.alpha= 0;
        )
        completion:^(BOOL finished){
            NSLog(@"Second anim finish")
        }];
    [UIView commitAnimations]; // Starting the animation.
}
```

Listing 7: Exemple de 2 animations à l'aide de la classe UIView. La première change la taille et l'emplacement d'un élément graphique et la deuxième change sa transparence.

Il aurait été intéressant d'avoir le temps pour rendre ce composant plus générique et de le publier sur internet pour ainsi éviter à d'autres utilisateurs de devoir refaire le même travail.

Recherche : Pour offrir à l'utilisateur la fonction chercher, deux méthode s'offrent à nous :

1. La première consiste à utiliser les requêtes SQL-Lite pour faire la recherche dans la base de données et d'afficher le résultat.
2. La deuxième est de faire la recherche directement dans la liste d'objets actuellement affichés et de masquer les éléments qui ne répondent pas au texte de recherche.

La deuxième façon est celle conseillée par Apple. La première obligerait à chaque requête de réinitialiser chaque objet et serait trop coûteuse en matière de ressources système.

Voici le code qui nous permet de filtrer les éléments dans la liste :

```
-(void) searchTableView {
    NSString *searchText = searchBar.text;
```



```

NSMutableArray *searchArray = [[NSMutableArray alloc] init]; // Strings for searching

// _persons is an array with the current displayed list
for (Person * p in _persons)
{
    NSString * s = [NSString stringWithFormat:@"% @ %@" ,p.nom , p.prenom ,p.carriere];
    // Whe want to search in the fileds : nom , prenom ,carriere
    [searchArray addObject:s];
}
int i=0;
for (NSString *sTemp in searchArray) // We check each row
{
    NSArray* separatedWord = [searchText componentsSeparatedByString: @" "];
    for (NSString *word in separatedWord) {
        NSRange titleResultsRange = [sTemp rangeOfString:word options:
        NSCaseInsensitiveSearch];
        if (titleResultsRange.length > 0){
            [copyListOfItems addObject:[_persons objectAtIndex:i]];
            break;
        }
    }
    i++;
}
[searchArray release];
searchArray = nil;
[ self display:copyListOfItems];
}

```

Listing 8: Methode de recherche dans une UITableView.

Lancer un appel téléphonique : La philosophie d’Apple veut garder chaque appli-
cation dans son propre cadre et limiter la communication avec d’autres applications pour
des raisons de sécurité. Cependant, quelques tâches de base sont tout de même permises
et l’une de celles-là est de lancer un appel téléphonique depuis d’autres applications. Mais
une fois l’appel lancé, l’application est mise en background et elle ne sera pas remise
au premier plan à la fin de l’appel. Cette contrainte est connue et elle est impossible à
contourner.

```

- (void)calltoNum:(NSString *)telNumber
{
    NSString *s = [NSString alloc] initWithFormat:@"tel://%@" ,telNumber];
    [[UIApplication sharedApplication]openURL:[NSURL URLWithString:s]];
    [s release];
}

```

Listing 9: Lancement d’un appel téléphonique sur l’iPhone.

Écrire un e-mail : On peut bien sûr utiliser la même façon que pour l’appel téléphonique
(remplacer tel :// par mailto), mais il existe une autre variante plus élégante. Cette va-

riante nous permet de rester dans l'application et d'éviter qu'à la fin de l'écriture de l'e-mail l'utilisateur doive réouvrir l'application pour continuer son travail. Le composant MFMailComposeViewController fournit avec l'iOS nous permet de faire ceci.

```
– (void)sendEmailTo:(NSString *)destination
{
    if ([MFMailComposeViewController canSendMail]) {
        MFMailComposeViewController *mailComposer = [[MFMailComposeViewController alloc]
init];
        [[mailComposer navigationBar] setTintColor:[UIColor colorWithRed:0.03f green:0.03f blue
:0.03f alpha:1.0f]];
        mailComposer.mailComposeDelegate = self;
        [mailComposer setSubject:@"Subject"];
        [mailComposer setMessageBody:@"Sent from ESIB@PAD" isHTML:NO];
        [mailComposer setToRecipients:[NSArray arrayWithObject:destination]];
        [self presentViewController:mailComposer animated:YES];
        [mailComposer release];
    } else {
        UIApplication *app = [UIApplication sharedApplication];
        [app openURL:[NSURL URLWithString:
            [NSString stringWithFormat:@"mailto:%@?subject=%@&body=%@",
personInformation.email,@"Subject",@"Sent from ESIB@PAD"]]];
    }
}
```

Listing 10: Ouverture de la fenêtre d'écriture d'e-mail

3.10 Calendrier

Diagramme de séquence

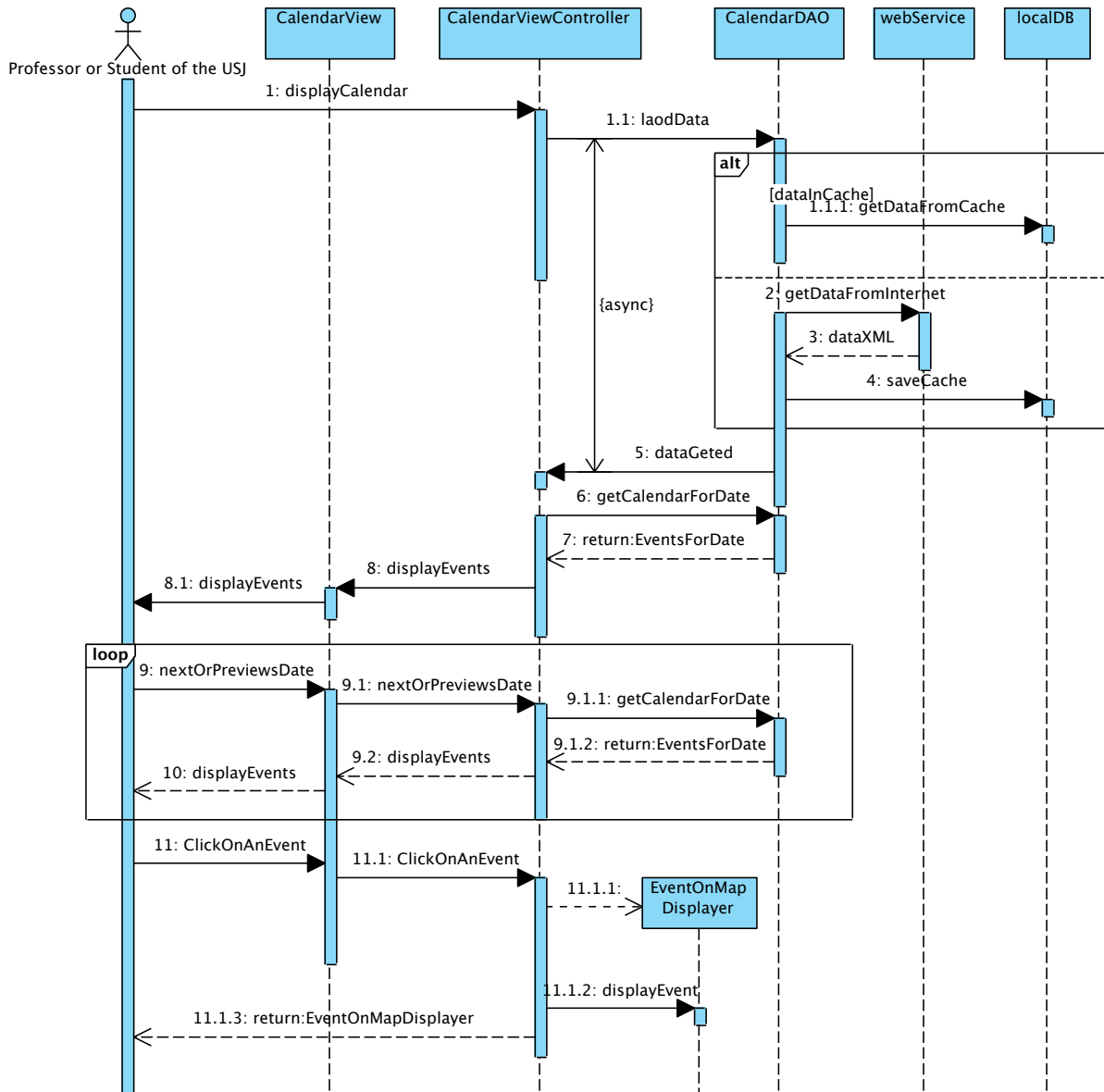


FIGURE 19: Exemple de séquence concernant l’affichage de l’horaire pour une journée et l’affichage du détail de l’évènement sur la carte

Diagramme de classe

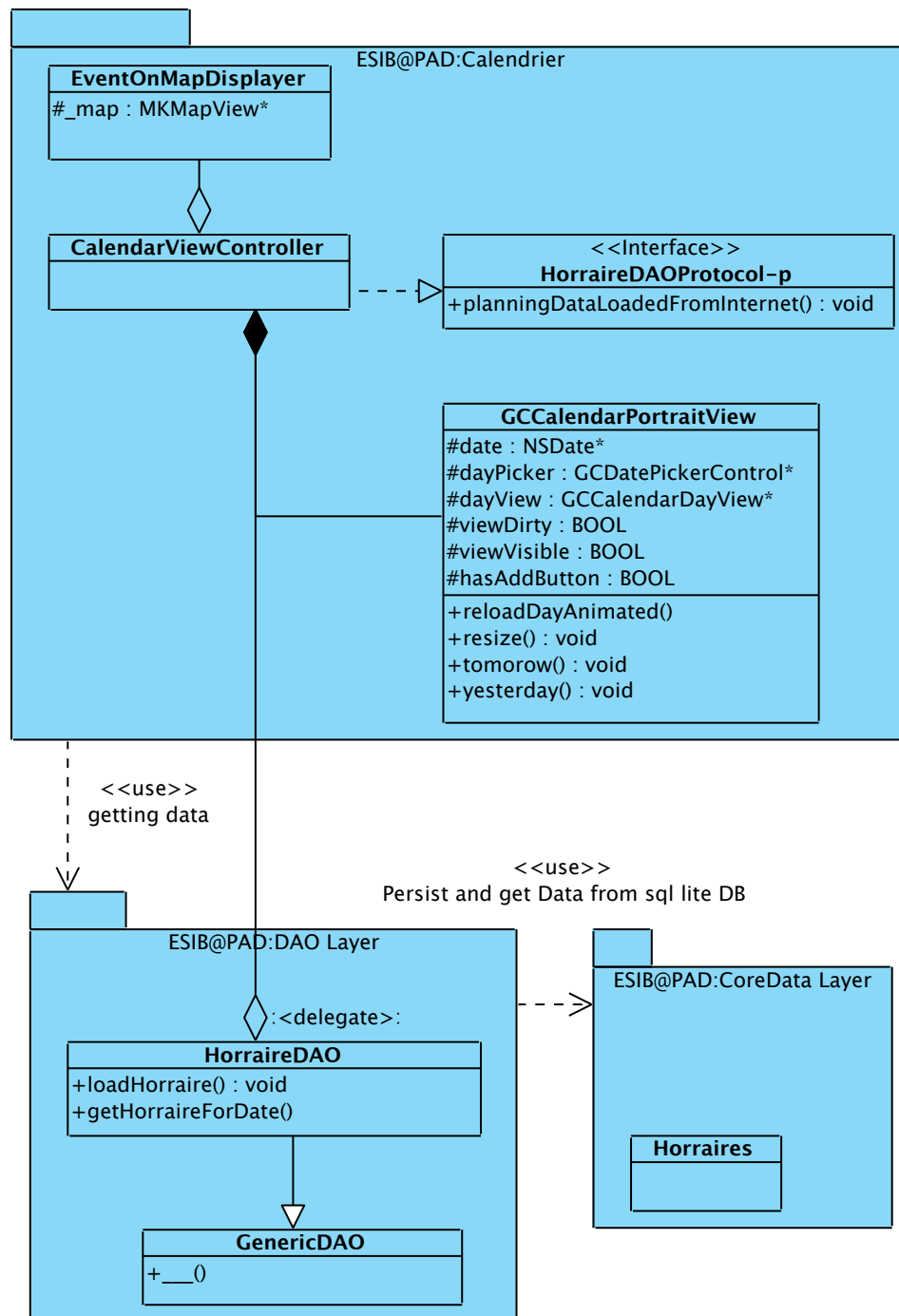


FIGURE 20: Diagramme de classe du composant calendrier

Discussion

GCCalendarPortraitView est un composant Opensource téléchargé depuis internet et qui permet d'afficher une journée d'un calendrier avec des événements. Ce composant était de base uniquement compatible en mode plein écran sur iPhone et ne supportait pas la rotation de l'écran. Les modifications nécessaires ont été faites pour pouvoir l'utiliser dans une partie spécifique de l'écran(plus grand pour l'iPad). L'ajout de la possibilité de passer au jour suivant, précédent grâce au mouvement de glissement du doigt a aussi été ajouté.

```
-(void) viewDidLoad{
    [super viewDidLoad];
    // Swipe Right notification
    UISwipeGestureRecognizer *swipeGesture = [[UISwipeGestureRecognizer alloc] initWithTarget:self
        action:@selector(swipe:)];
    swipeGesture.direction = UISwipeGestureRecognizerDirectionRight;
    [dayView addGestureRecognizer:swipeGesture];
    [swipeGesture release];

    // Swipe Left notification
    UISwipeGestureRecognizer *swipeGestureLeft = [[UISwipeGestureRecognizer alloc] initWithTarget
        :self action:@selector(swipe:)];
    swipeGestureLeft.direction = UISwipeGestureRecognizerDirectionLeft;
    [dayView addGestureRecognizer:swipeGestureLeft];
    [swipeGestureLeft release];
}
// Event sent when the swipe mouvement is recognized. -
-(void)swipe:(UISwipeGestureRecognizer *)swipe{

    if (swipe.direction == UISwipeGestureRecognizerDirectionLeft){
        [self tomorrow];
    }else if (swipe.direction == UISwipeGestureRecognizerDirectionRight){
        [self yesterday];
    }
}
```

Listing 11: Enregistrement pour les notifications du mouvement glissement du doigt et réception de l'événement

Affichage de l'événement sur la carte Le même framework MapKit utilisé dans le composant Map est réutilisé ici pour afficher l'emplacement de l'événement sur la carte.

3.11 ExamResult

Diagramme de séquence

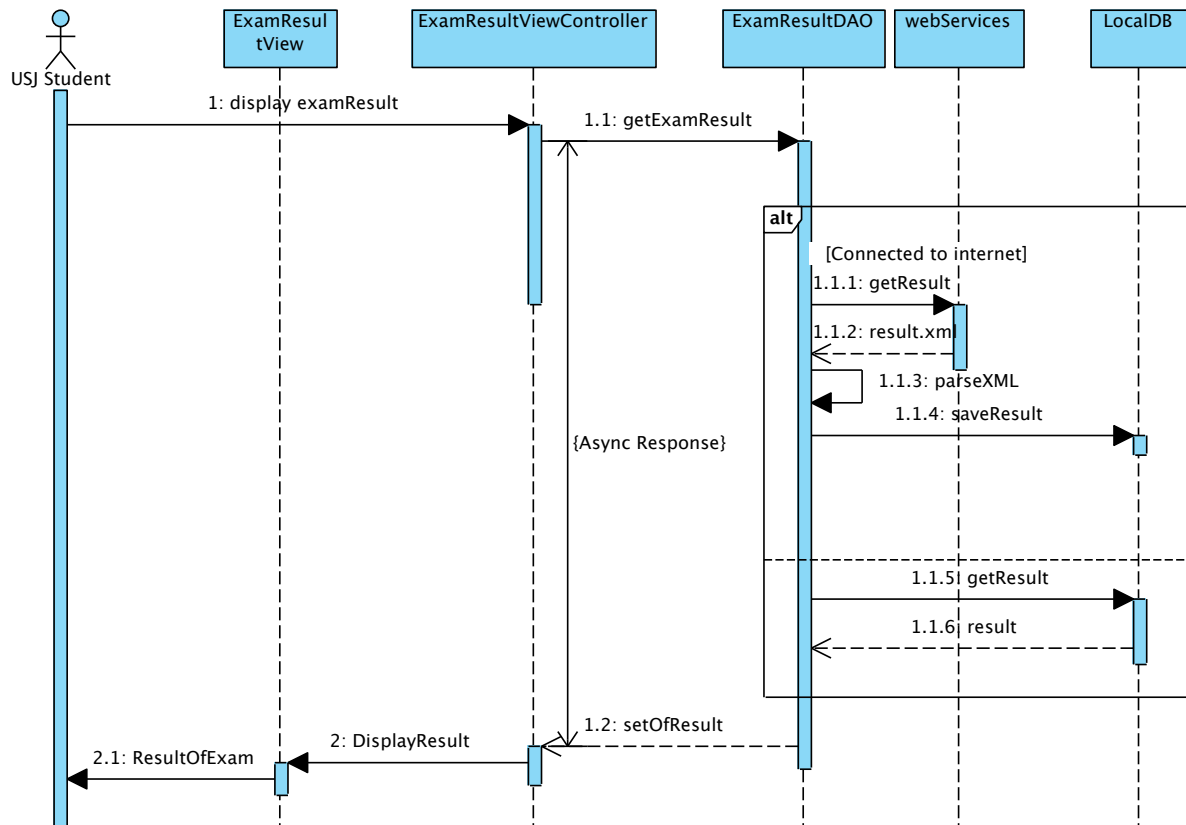


FIGURE 21: Exemple de séquence concernant l’affichage de l’horaire pour une journée et l’affichage du détail de l’évènement sur la carte

Diagramme de classe

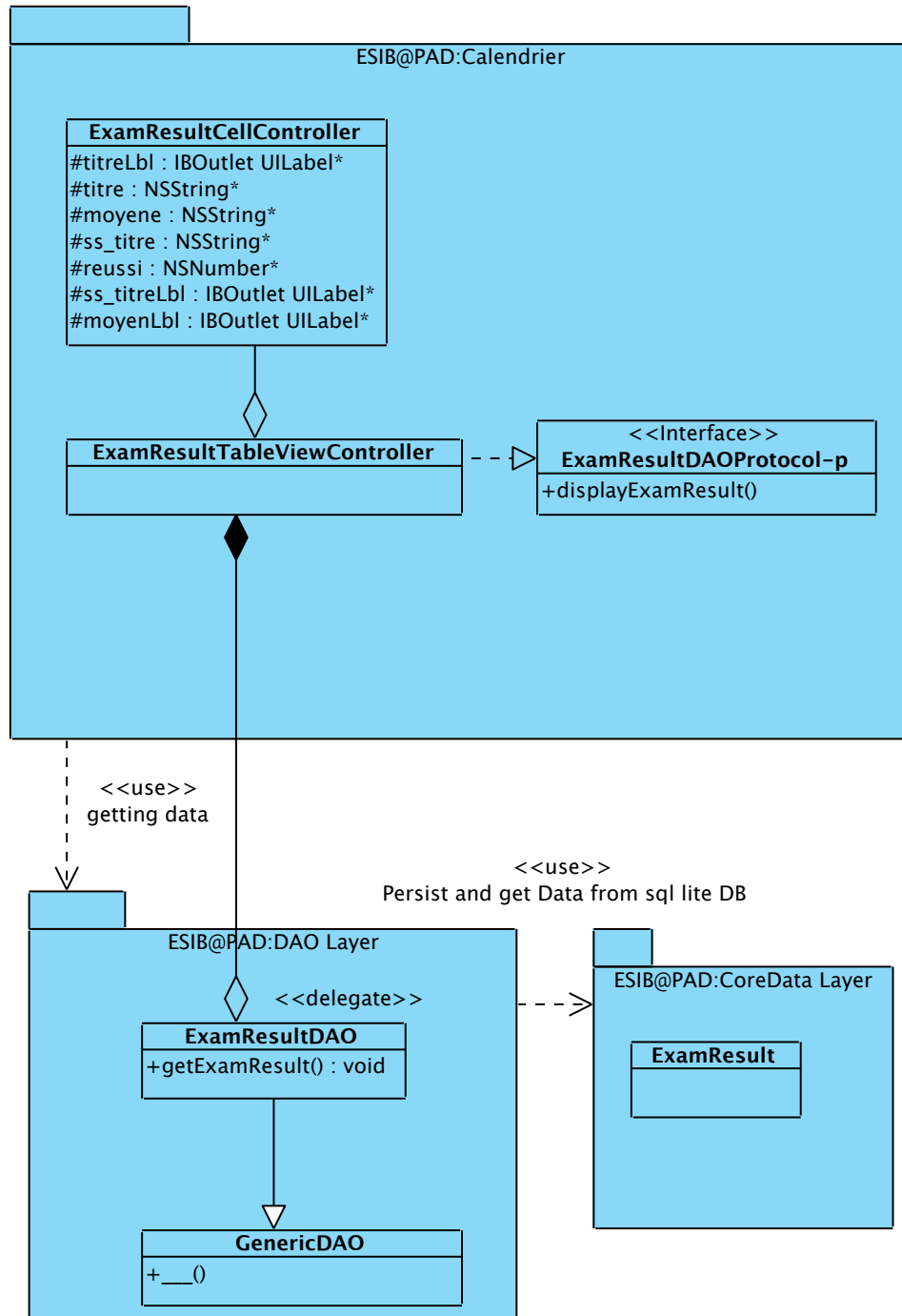


FIGURE 22: Diagramme de classe du composant calendrier

Discussion

Tout comme les news , si il y a une connexion internet, les données seront directement téléchargées depuis internet et non pas prises depuis le cache.

Glossary

Core Data Core Data is part of the Cocoa API in Mac OS X first introduced with Mac OS X 10.4 Tiger and for iOS with iPhone SDK 3.0.[2] It allows data organised by the relational entity-attribute model to be serialised into XML, binary, or SQLite stores.
(Source wikipedia)http://en.wikipedia.org/wiki/Core_Data . 11

XCode XCode est un environnement de développement pour Mac OS X.<http://fr.wikipedia.org/wiki/Xcode> . 11