Projet iOS – Stade Rochelais

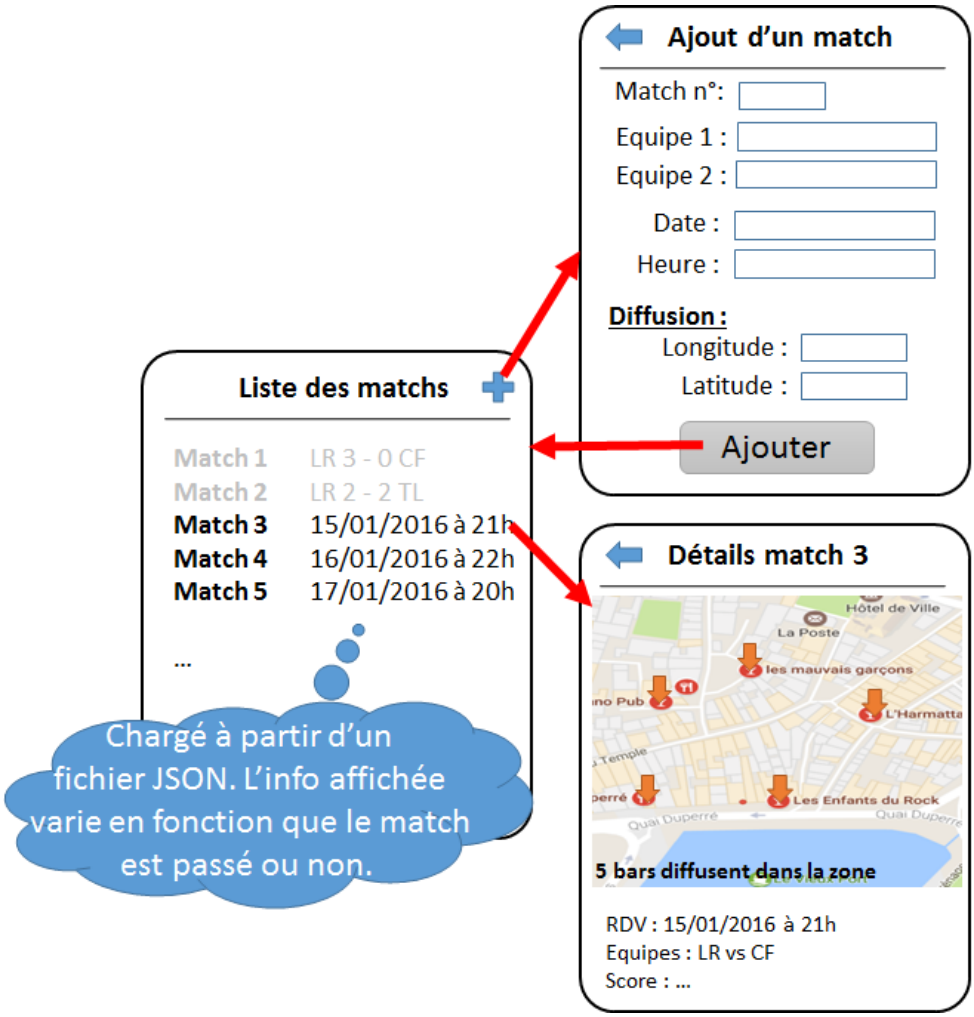
LP Informatique Répartie et Mobile, 2017
guillaume.chiron@univ-lr.fr - michel.mernard@univ-lr.fr

Objectifs du projet :

Mise en place d'une structure d'application qui pourra servir de base à votre application « Stade Rochelais », à savoir :

- Structurer une application iOS de type MVC.
- Afficher les données sous différentes formes (liste, carte, champs)
- Intégrer des données issues d'un fichier JSON (local, et en ligne).
- Utiliser MapKit

Organisation de l'application :



Etapes par étapes :

L'énoncé apporte un fil conducteur détaillé (pour rassurer les plus frileux), cependant libre à vous de parvenir à réaliser les étapes suivantes par vos propres moyens.

Conseils

- Lisez chaque étape en entier avant de la réaliser.
- Testez votre projet avec le simulateur après chaque question (quand c'est possible) ;
- Faites des sauvegardes après chaque étape pour pouvoir revenir en arrière en cas de nécessité ;
-Google (ou autres moteurs de recherche) sont vos amis !!!

Etape 1: Création du projet, structuration de l'application

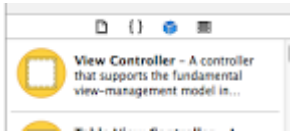
Dans Xcode, créer un nouveau projet de type **Master-Detail Application** avec un **Storyboard**. Par défaut, ce projet est composé d'un contrôleur principal **AppDelegate**, d'un **Storyboard** et de deux contrôleurs **MasterViewController** et **DetailViewController**. Contrairement aux .xib, le **Storyboard** est composé de plusieurs vues, il permet de définir de manière graphique l'enchaînement entre les différentes vues. La vue **Master (UITableView)** servira à afficher une liste des matchs. La vue **Detail (UIView)** servira à afficher les détails d'un match donné et aussi son ou ses lieux de diffusion.

Commençons par nettoyer un peu le code :

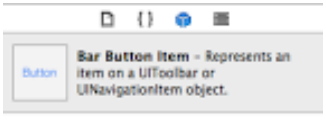
- Dans **MasterViewControler.m**, supprimer (ou mettre en commentaire) le code relatif à l'ajout des boutons « Edit » & « + » au niveau de la barre de navigation. (dans la méthode ***viewDidLoad***)

Nous allons ajouter une 3ème vue au **Storyboard** qui permettra l'ajout d'un match à liste :

- Ajouter via l'interface graphique un nouvel objet **View Controller** dans le **Storyboard**. Laisser cette vue vide pour le moment.



- Ajouter un **Bar Button Item** dans la barre de navigation de la vue **Master**. (*attention, il faut avoir mis le "focus" sur la vue en question pour pouvoir ajouter un élément dedans...*)

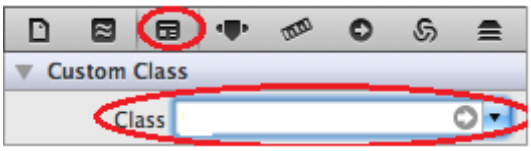


- Lier ce nouveau bouton au nouveau **View Controller** par un lien de type **Push** (*CTRL+Click+Déplacer*).

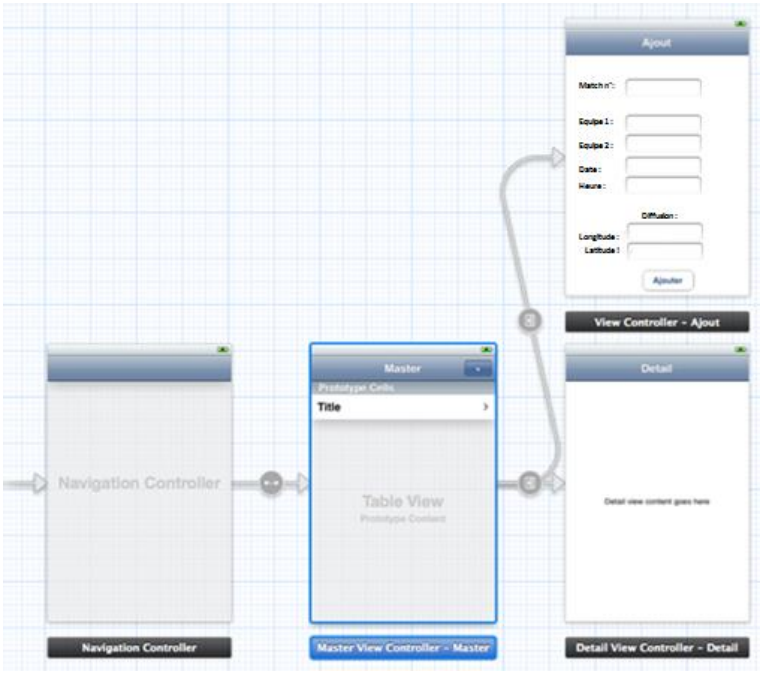
Pour pouvoir gérer les actions ou "messages" envoyées par la nouvelle vue, il faut créer le contrôleur associé à cette vue :

- Ajouter au projet une classe ayant le nom **AjoutViewController** (File->New file->**Objective-C Class,** type **subclass of UIViewControler**, sans xib) à votre projet.

- Ensuite, via l'interface de Xcode (Barre latérale droite->3ème onglet "Identity inspector"), définir la propriété **Custom class** de la vue "Ajout" comme étant de type **AjoutViewController.** (attention, il faut sélectionner la vue dans sa globalité, et pas seulement un élément contenu dans celle-ci)



Vous devez avoir un story-board semblable :



--------- FAIRE VALIDER L'ETAPE 1 PAR L'ENCADRANT ---------

Etape 2 : Gestion de la liste des matchs (vue/contrôleur Master)

Pour manipuler les informations concernant les matchs, nous allons créer une classe **Match**.

- Ajouter à votre projet une classe **(**File->New file->**Objective-C class)**.
- Ajouter à la classe **Match** les 7 attributs suivants : numero (int), equipe1 (NSString *), equipe2 (NSString *), latitude (float), longitude (float), heure (NSString *), date (NSString *).
    - Implémenter une méthode constructeur « `initWithNum` » permettant l'instanciation d'un **Match.** Dans un premier temps, on y passe simplement 1 argument (le numéro du match) en paramètre.

Aide sur comment implémenter un classe :
https://openclassrooms.com/courses/programmez-en-objective-c/les-classes-1

- Instancier dans **MasterViewControleur.m**, un nouveau **Match** dès la fin du chargement de la vue en utilisant la méthode constructeur de **Match**. Attention à bien importer le fichier d'entête « Match.h »

```
ex: Match * nouveauMatch = [[Match alloc] initWithNum:1];
```

Nous allons maintenant gérer l'ajout & l'affichage des **Matchs** dans le **TableView**.

- Renommer la méthode **insertNewObject** (celle générée par défaut lors de la création du projet) en **insertNewMatch** et l'adapter afin qu'elle prenne en paramètre un objet de type **Match *** de manière à gérer l'ajout d'instances de **Match** dans le conteneur "_object" existant prévu à cet effet.
  A la fin de la méthode, ajouter **[self.tableView reloadData]** pour prévoir le rafraichissement du **tableView**.

- Adapter le contenu de la méthode **cellForRowAtIndexPath** pour gérer l'affichage des matchs dans le **tableView**. Il suffit pour cela de définir une entrée dans le **TableView** de type **UITableViewCellStyleSubtitle** (au lieu de **UITableViewCell**) et de lui associer un **textLabel** (nom du match), et d'un **detailTextLabel** (correspondant à la concaténation de la date et de l'heure).
  Voici un exemple de code :

```
- (UITableViewCell *)tableView:(UITableView *)tableView cellForRowAtIndexPath:(NSIndexPath *)indexPath {
UITableViewCell *cell = nil;
cell = [tableView dequeueReusableCellWithIdentifier:@"cell"];
if (cell == nil) {
 cell = [[UITableViewCell alloc] initWithStyle:UITableViewCellStyleSubtitle reuseIdentifier:@"cell"];
}
cell.textLabel.text = @"Title1";
cell.detailTextLabel.text = @"Subtitle 1";
return cell;
}
```

PS : Il faut régler dans l'interface de Xcode le style des cellules du **tableView** en "Subtitle".

--------- *FAIRE VALIDER L'ETAPE 2 PAR L'ENCADRANT* ---------

Etape 3 : Gestion de l'ajout d'un match (<u>vue/contrôleur Ajout</u>).

Cette vue permet d'ajouter un match en renseignant les informations nécessaires dans les champs de saisis.

- Mettre en place une méthode liée au bouton "Ajouter" (IBAction) dans laquelle une instance de **Match** sera créée, en se basant sur les infos fournies dans les champs de saisies.

- Pour ajouter un nouvel match au **tableView**, passer par la méthode **insertNewMatch** (définie dans **MasterViewController**)**.** Attention : Cela nécessite d'avoir accès aux méthodes membres de **MasterViewController** à partir son contrôleur fils (sur lequel vous travailler actuellement).
  Une manière de procéder est que le **MasterViewController** renseigne sa propre référence (self) au contrôleur fils lors du changement la vue associée (à savoir dans la méthode "**prepareForSegue**").
  En détails :
  - Dans **AjoutViewControleur.h,** ajouter "#import MasterViewController.h"
    et ajouter un attribut "MasterViewController * parentMc;"
  - Dans **AjoutViewControleur.m,** définir les accesseurs :
    ```
    -(void) setMC:(MasterViewController *) mc {
        parentMc =mc;
    }
    -(MasterViewController *) getMC:() {
        return parentMc;
    }
    ```
    Ajouter dans **AjoutViewControleur.h** les signatures :
    "-(void) setMC:(**MasterViewController ***) mc;" et
    "-(**MasterViewController ***) getMC();"

  - Ensuite, dans l'interface de Xcode (en mode StoryBoard), cliquer sur le lien entre le **MasterView** et **AjoutView** et définir un identifiant de transition (ex: "*passToAjout*").

  - Ajouter "#import AjoutViewControleur.h" dans **MasterViewController.m**
  - Puis dans **MasterViewController.m,** ajouter à la méthode **prepareForSegue** :
    ```
    if ([[segue identifier] isEqualToString:@"passToAjout"]) {
      [[segue destinationViewController] setMC:self];
    }
    ```

- Maintenant que votre instance de « AjoutViewControleur » connaît son **MasterViewController,** vous pouvez appeler la méthode **insertNewMatch** (que vous avez déjà implémenté dans **MasterViewController**) au travers de la variable locale **"mc"**). Il suffit alors de lui passer en paramètre une instance de **Match**.

- Dans la vue « Ajout », utiliser la méthode **popViewControllerAnimated** pour que l'application retourne automatiquement sur la vue **Master** une fois le match ajouté dans le conteneur de **Matchs** ("_object" dans **MasterViewController**).

--------- *FAIRE VALIDER L'ETAPE 3 PAR L'ENCADRANT* ---------

Etape 4 : Chargement des matchs à partir d'un fichier JSON.

Au chargement de l'application, remplir le containeur de matchs servant à peupler le **tableView.** Pour cela, vous pouvez vous inspirer du code suivant. A vous de le tester/décortiquer/adapter… Vous pouvez également vous inspirer du TP précédents.

```
    NSData *matchsData = [NSData dataWithContentsOfFile:@"/Users/.../Matchs.json"];
    NSError * error;
    NSDictionary * matchsDictionary = [NSJSONSerialization JSONObjectWithData: matchsData
options:kNilOptions error:&error];

    for (NSString* keyMatch in matchsDictionary) {
       NSDictionary * detailsMatchDictonary = [matchsDictionary objectForKey: keyMatch];
        (NSString *) e1 = [detailsMatchDictonary objectForKey:@"equipe1"];
        NSLog(@"Match %@ = %s", keyMatch, e1);
    }
```

--------- FAIRE VALIDER L'ETAPE 4 PAR L'ENCADRANT ---------

Etape 5 : Gestion de la carte (vue/contrôleur Detail Map)

Ajoutez la référence « MapKit » dans les paramètres de votre projet.

- Ajouter les composants (MKMapView**,** boutons, labels…) à la vue comme présenté dans l'illustration de l'application.

- Remplir les champs (ou labels selon ce que vous avez choisi d'utiliser pour cette vue) à l'aide des attributs de l'objet **Match** passé en paramètre lors du passage de la vue Master à la vue Détail.

- Dès le chargement de la vue terminé, faire que la carte se centrer sur la latitude/longitude d'un point de diffusion du match en question, et zoomer sur une zone de 1km².

- Ajouter une punaise (**MKAnnotation**) à l'emplacement du bar diffusant le match. Voir plus d'infos dans le TP précédent.

- Faire apparaitre les données concernant le bar (ex : long, lat…) lors du clique sur une punaise.

--------- FAIRE VALIDER L'ETAPE 5 PAR L'ENCADRANT ---------

Etape 6 : Récupération du fichier JSON en ligne

- Utiliser la classe **RequeteManager (présentée en annexe et disponible sur moodle)** pour gérer la connexion à une ressource web. La récupération se fera au sein de la classe **MasterViewController** lors du chargement de l'application. Pour que la requête HTTP puisse être gérée de manière asynchrone, **RequeteManager** utilise la notion de « délégué », ce qui lui permet de notifier à ses abonnées qu'elle a fini d'acquérir la ressource demandée. Dans notre cas, la classe **MasterViewController** qui est l'abonné, donc elle doit implémenter le protocole **RequeteManagerDelegate** via la notation suivante :
  `@interface MasterViewController … … <RequeteManagerDelegate>`

- Ainsi, il est maintenant nécessaire de définir la méthode `-(void)requeteManagerResponse:(NSString *) htmlsource` dans la classe **MasterViewController** qui sera en charge du traitement du fichier JSON renvoyé par ***RequeteManager.***

- Dans un premier temps, vous pouvez afficher le contenu du fichier récupéré dans la console (via **NSLog**).

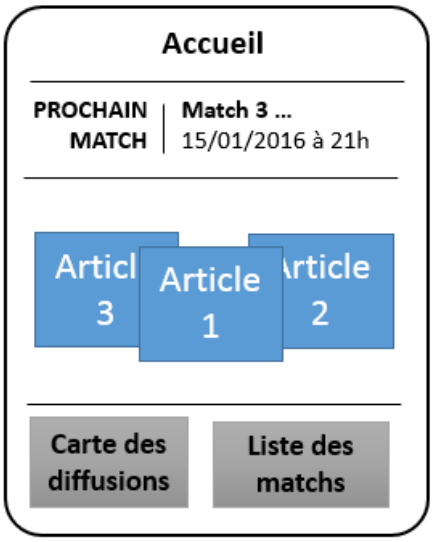
--------- *FAIRE VALIDER L'ETAPE 6 PAR L'ENCADRANT* ---------

Etape 7 :

- Gérer dans le fichier JSON et sur la carte la diffusion d'un match dans plusieurs bars. Afficher (et actualiser lors du déplacement de la carte) dans un label le nombre de bars qui diffusent et qui sont visibles sur la carte. Voir schéma de l'application au début de l'énoncé.

--------- *FAIRE VALIDER L'ETAPE BONUS PAR L'ENCADRANT* ---------

Etape 8 (bonus) :



Ajouter une vue d'accueil (voir ci-contre) à l'application.

- Un clique sur « Carte des diffusions » affiche une carte avec tous les bars diffusant tous les matchs.
- Un clique sur « Liste des matchs » affiche le Tableview implémenté précédemment montrant la liste des matchs.
- Le Carousel (liste de articles) n'est pas un composant par défaut d'iOS, vous devez utiliser une bibliothèque externe du type : https://github.com/nicklockwood/iCarousel
(voir iCarousel/Examples/Advanced iOS Demo)

--------- *FAIRE VALIDER L'ETAPE BONUS PAR L'ENCADRANT* ---------

Annexes :

Connexion à un serveur Web

*La classe **RequeteManager** permet de récupérer le code source HTML d'une URL de manière asynchrone. La réponse est renvoyée via la méthode déléguée **requeteManagerResponse**. Avant l'envoi d'une requête par le contrôleur via **RequeteManager**, il faut définir le delegate comme étant le contrôleur lui-même (voir ci-dessous):*

```
// ----- RequeteManager.h --------
#import <Foundation/Foundation.h>

@protocol RequeteManagerDelegate <NSObject>
@required
    - (void)requeteManagerResponse:(NSString *) htmlsource;
@end

@interface RequeteManager : NSObject {
    id <RequeteManagerDelegate> delegate;
    NSMutableData * receivedData;
}

@property (retain) id delegate;
-(void)lancerRequete:(NSString *)url;

@end
```

EXEMPLE D'APPEL

```
- (IBAction)getInfos:(id)sender {

    RequeteManager * rm = [[RequeteManager alloc] init];
    rm.delegate = self;
    [rm lancerRequete:@"http://fr.wikipedia.org/wiki/Paris"];

}
```

EXEMPLE DE REPONSE

```
- (void)requeteManagerResponse:(NSString *) htmlsource {
    ...
}
```

```
// --------- RequeteManager.m ----------------
#import "RequeteManager.h"

@implementation RequeteManager
@synthesize delegate;

-(void)lancerRequete:(NSString *)url {

    NSURLRequest *theRequest=[NSURLRequest requestWithURL:[NSURL URLWithString:url]
                                              cachePolicy:NSURLRequestUseProtocolCachePolicy
                                          timeoutInterval:60.0];

    NSURLConnection *theConnection=[[NSURLConnection alloc] initWithRequest:theRequest delegate:self];

    if (theConnection) {
        receivedData = [[NSMutableData data] retain];
    } else {
        NSLog(@"Erreur de connexion");
    }
}

- (void)connection:(NSURLConnection *)connection didReceiveResponse:(NSURLResponse *)response {
    [receivedData setLength:0];
}

- (void)connection:(NSURLConnection *)connection didReceiveData:(NSData *)data {
    [receivedData appendData:data];
}

- (void)connection:(NSURLConnection *)connection didFailWithError:(NSError *)error {


    NSLog(@"Erreur");
}

- (void)connectionDidFinishLoading:(NSURLConnection *)connection {
    NSString * receivedString = [[NSString alloc] initWithData:receivedData encoding:NSASCIIStringEncoding];
    [[self delegate] requeteManagerResponse:receivedString];


}
@end
```