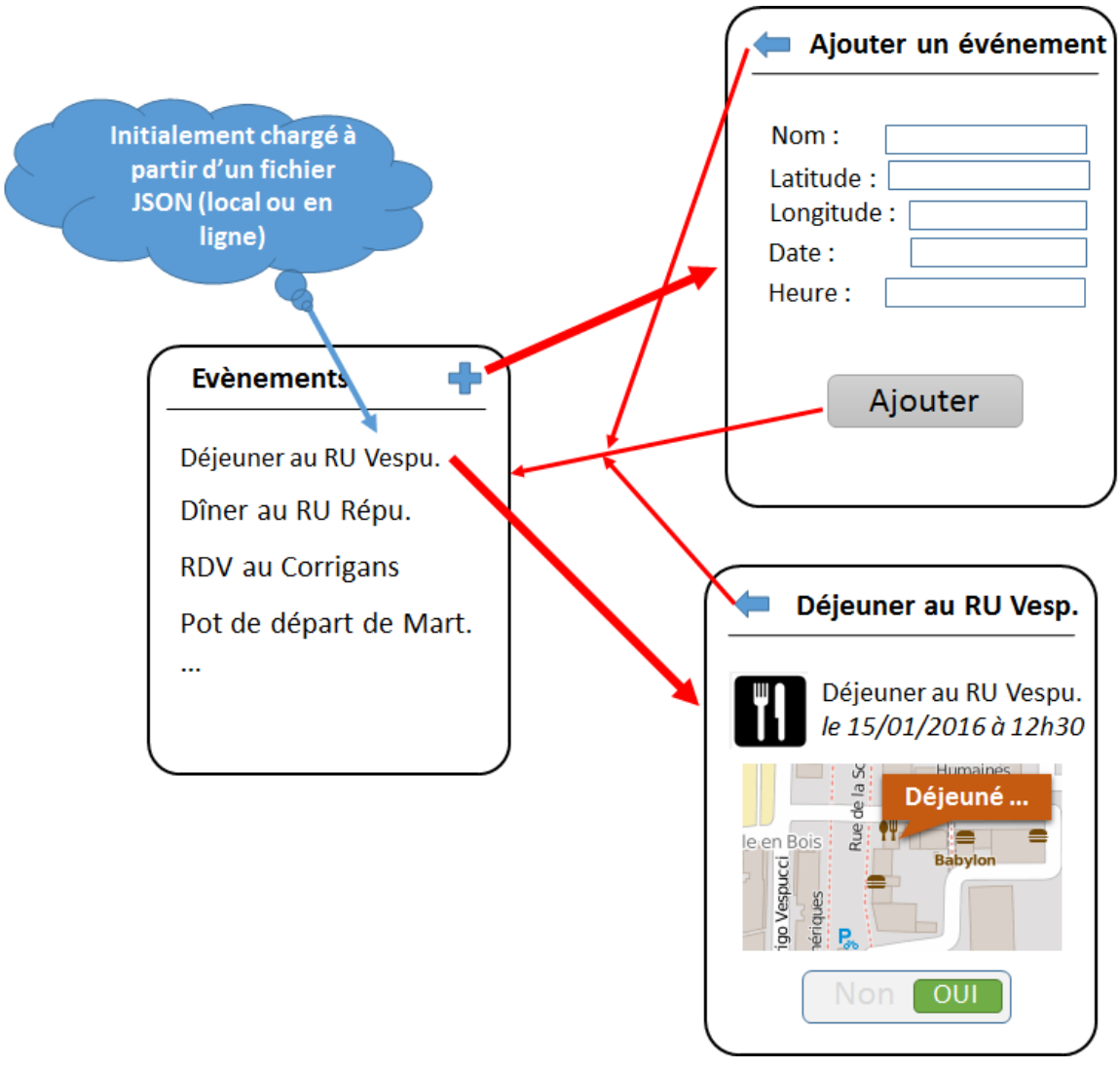
Projet iOS

LP Informatique Répartie et Mobile, 2016
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 QMV, 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

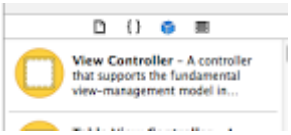
Dans Xcode, créer un nouveau projet de type **Master-Detail Application** avec un **Storyboard**. Par défaut, ce projet est composé d'un contrôleur principal **AppDelegate**, d'un **Storyboard** et de deux contrôleurs **MasterViewController** et **DetailViewController**. Contrairement aux .xib, le **Storyboard** 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 évènements. La vue **Detail (UIView)** servira à afficher les détails d'un évènement donné, notamment à l'aide d'une carte.

Nous allons nettoyer un peu le code :

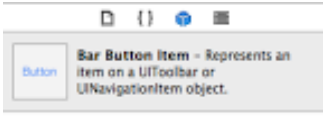
- Dans **MasterViewControler.m**, commenter (ou supprimer) le code relatif à l'ajout des boutons « Edit » & « + » au niveau de la barre de navigation. (dans la méthode ***viewDidLoad***)

Nous allons ajouter une 3ème vue au **Storyboard** qui permettra l'ajout d'un évènement via un formulaire :

- Ajouter via l'interface graphique un nouvel objet **View Controller** dans le **Storyboard**.



- Ajouter sur cette nouvelle vue les contrôles comme indiqué précédemment sur l'illustration de l'application (le bouton « Ajouter », ainsi que les labels « Nom », « Latitude », « Longitude », « Date » et « Heure », ainsi que les champs de saisie correspondants.
- 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...*)

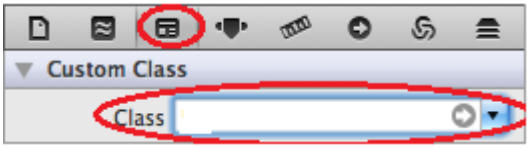


Lier ce nouveau bouton au nouveau **View Controller** par un lien de type **Push** (*CTRL+Click+Déplacer*).

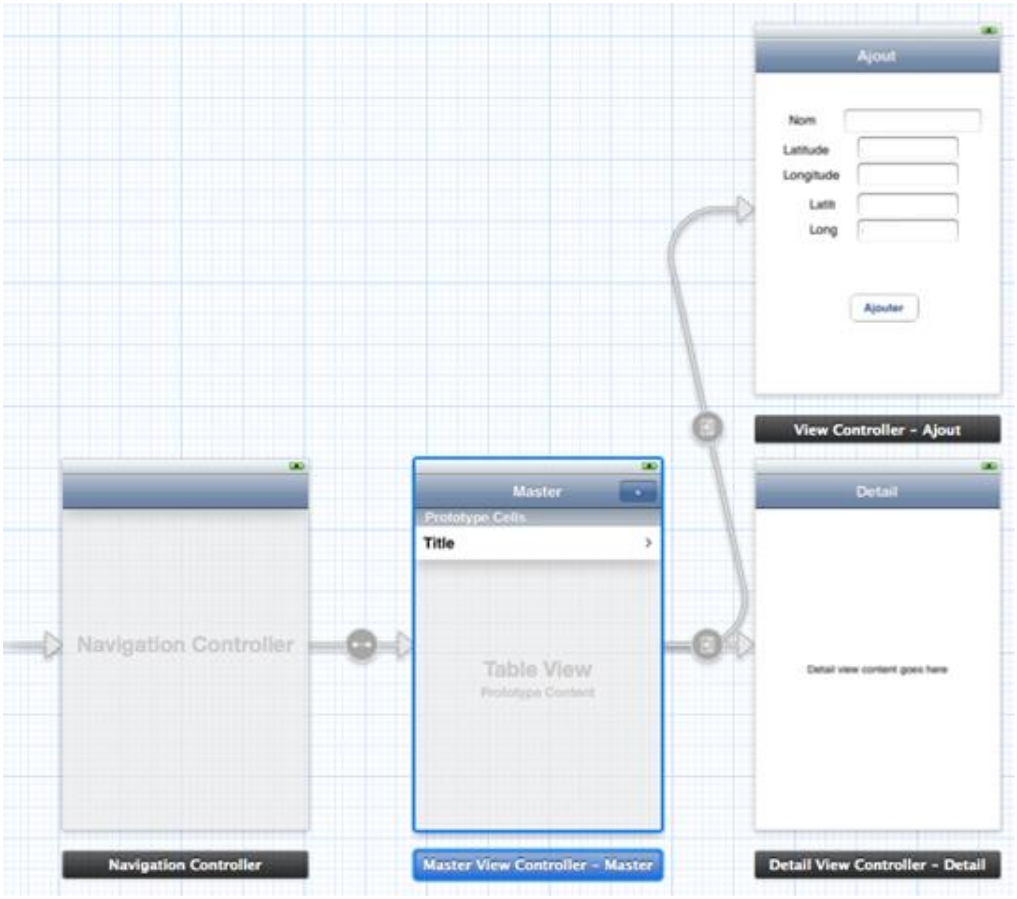
Pour pouvoir gérer les actions ou "messages" envoyées par la nouvelle vue, il faut créer le contrôleur associé à cette vue :

- 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 évènements (vue/contrôleur Master)

Pour manipuler les informations concernant les évènements, nous allons créer une classe **Evenement**.

- Ajouter à votre projet une classe **Evenement (**File->New file->**Objective-C class)**.
- Ajouter à la classe **Evenement** les 5 attributs suivants : nom (NSString *), latitude (float), longitude (float), heure (NSString *), date (NSString *).
    - Implémenter une méthode constructeur permettant l'instanciation de l'**Evenement** (en passant le nom en paramètre)

RAPPEL

Dans le .h

@interface Evenement : NSObject {
liste des attributs
}

liste des @property *(ex: copy, weak, strong...) => règles à suivre pour la synthèse des accesseurs*

liste des méthodes

@end

Dans le .m

#import "Evenement.h"
@implementation Evenement

liste des @synthesize *pour chaque attributs (permet la synthèse automatique des accesseurs).*

définition de méthodes déclarées *dans le .h*

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

```
ex: Evenement * nouvelEvenement = [[Evenement alloc] initWithName:@"RDV là bas"];
```

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

- Renommer la méthode **insertNewObject** existante en **insertNewEvenement** et l'adapter afin qu'elle prenne en paramètre un objet de type **Evenement *** de manière à gérer l'ajout d'instances d'**Evenement** dans le conteneur "_object" existant déjà 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 évènements 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 de l'évènement), et d'un **detailTextLabel** (correspondant à la concaténation de la date et de l'heure).
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 évènement (vue/contrôleur Ajout).

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

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

- Pour ajouter un nouvel évènement au **tableView**, passer par la méthode **insertNewEvenement** (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 (à savoir dans la méthode "**prepareForSegue**").
  En détails :
  - Dans **AjoutViewControleur.h,** ajouter "#import MasterViewController.h",
    et déclarer un attribut via "@property (retain,strong) MasterViewController * mc;"
  - Dans **AjoutViewControleur.m,** définir une nouvelle méthode :
    ```
    -(void) setMC:(MasterViewController *) mc {
        _mc=mc;
    }
    ```
    Ajouter sa signature "-(void) setMC:(**MasterViewController ***) mc;" dans le .h.
  - Ensuite, dans l'interface de xcode, cliquer sur le lien entre le **MasterView** et **AjoutView** et définir un identifiant de transition (ex: "*mon_Identifier*").
  - Ajouter "#import AjoutViewControleur.h" dans **MasterViewController.m**
  - Ajouter dans la méthode **prepareForSegue** :
    ```
    if ([[segue identifier] isEqualToString:@"mon_Identifier"]) {
      [[segue destinationViewController] setMC:self];
    }
    ```

- Appeler ensuite la méthode **insertNewEvenement** via le **MasterViewController** (que vous avez référencé dans la variable locale **"_mc"** précédemment) en lui passant en paramètre l'instance de l'**Evenement** que vous avez créée.

- Dans la vue « Ajout », utiliser la méthode **popViewControllerAnimated** pour que l'application retourne automatiquement sur la vue **Master** une fois l'évènement ajouté dans le conteneur d'**Evenements**.

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

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

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

```
    NSData *eventsData = [NSData dataWithContentsOfFile:@"/Users/.../evenementsQVM.json"];
    NSError * error;
    NSDictionary * eventsDictionary = [NSJSONSerialization JSONObjectWithData: eventsData
options:kNilOptions error:&error];

    for (NSString* keyEvent in eventsDictionary) {
       NSDictionary * detailsEventDictonary = [eventsDictionary objectForKey: keyEvent];
        float lat = [detailDictonary objectForKey:@"lat"];
        NSLog(@"Latitude de %@ = %f", keyEvent, lat);
    }
```

--------- *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 **Evenement** 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 de l'évènement en question, et zoomer sur une zone de 1km².

- Ajouter une punaise (MKAnnotation) à l'emplacement de l'évènement. (plus d'infos dans le TP précédent)

- Faire apparaitre les données concernant l'évènement (ex : Nom de l'évènement & date, heure) 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 bonus : Notifications !

- Lors d'un clique sur un bouton (ou autre évènement de votre choix), faire apparaitre une notification affichant les détails du prochain évènement.

--------- *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
```

Données au format JSON


```
{
   "Déjeuner au RU Vespucci":
   {
      "lat":46.1520087,
      "lon":-1.1561935,
      "date":"15/01/2016",
                  "heure":"12:30",
                  "type":"manger"
   }
   ,
   "Dîner au RU République":
   {
      "lat":46.1388953,,
      "lon":-1.1539113,
      "date":"15/01/2016",
                  "heure":"19:30",
      "type":"manger"
   }
   ,
   "RDV au Corrigans":
   {
      "lat":46.1624567,
      "lon":-1.1485574,
      "date":"16/01/2016",
                  "heure":"21:00",
      "type":"boire"
   }
   ,
   "Pot de départ de Martine":
   {
      "lat":46.1476946,
      "lon":-1.1572324,
      "date":"20/01/2016",
                  "heure":"17:30",
      "type":"goûter"
   }
   ,
   "Galette des rois de l'IUT Info":
   {
      "lat":46.1419244,
      "lon":-1.1534127,
      "date":"27/01/2016",
                  "heure":"11:45",
      "type":"goûter"
   }
   ,
   "Sortie The Roof":
   {
      "lat":46.1519373,
      "lon":-1.1579042,
      "date":"04/02/2016",
                  "heure":"19:00",
      "type":"sport"
   }
}
```