Mini Projet iOS

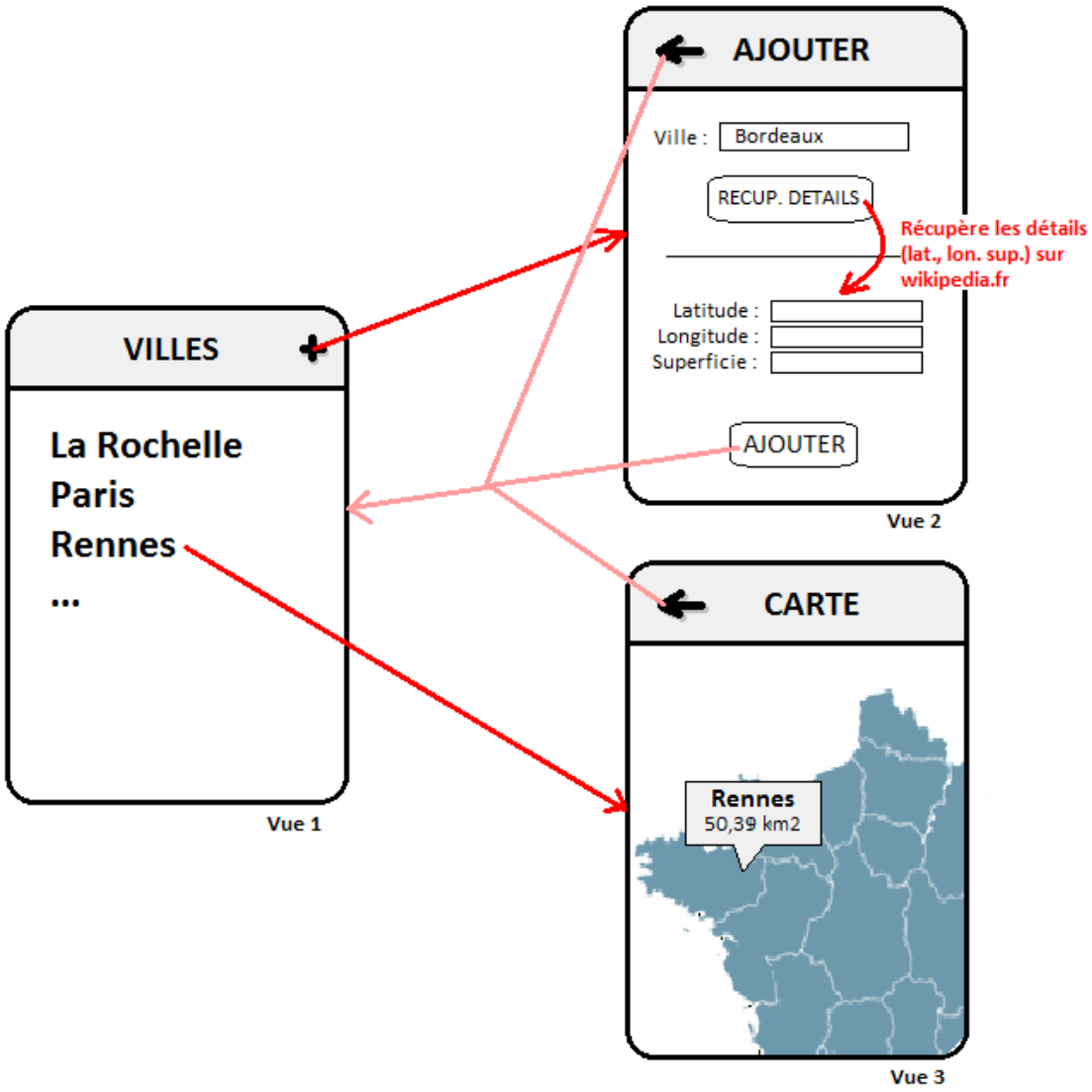
LP Informatique Répartie et Mobile, 2013/2014

guillaume.chiron@univ-lr.fr - michel.mernard@univ-lr.fr

Objectifs du projet :

- Structurer une application iOS de type MVC.
- Afficher les données sous différentes formes (liste, carte)
- Récupérer des données sur le web (via le code source HTML)
- Utiliser MapKit

Organisation de l'application :



Etapes par étapes :

Conseils

Bien lire l'énoncé, des détails précis sont donnés
Testez votre projet avec le simulateur après chaque question (quand c'est possible) et
Faites des sauvegardes après chaque étape pour pouvoir revenir en arrière en cas de nécessité

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** comprend 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 villes. La vue **Detail (UIView)** servira à afficher une ville sélectionnée sur une carte.

Nous allons nettoyer un peu le code :

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

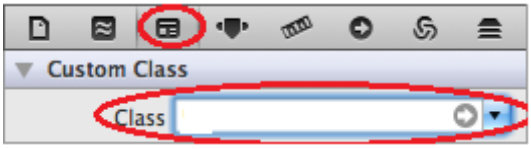
Nous allons ajouter une 3ème vue au **Storyboard** qui permettra l'ajout d'une ville via un formulaire :

- Ajouter via l'interface graphique un nouvel objet **View Controller** dans le **Storyboard**.
- Ajouter sur cette nouvelle vue les contrôles comme indiqué sur l'illustration ci-après (les boutons « Récup Détail » et « Ajouter », ainsi que les labels « Nom », « Latitude », « Longitude », « Superficie » et les champs de saisie correspondant.
- Ajouter un **Button** dans la barre de navigation de la vue **Master**.
- Lier ce bouton au nouveau **View Controller** par un lien de type **Push** (CTRL+Click+Déplacer).

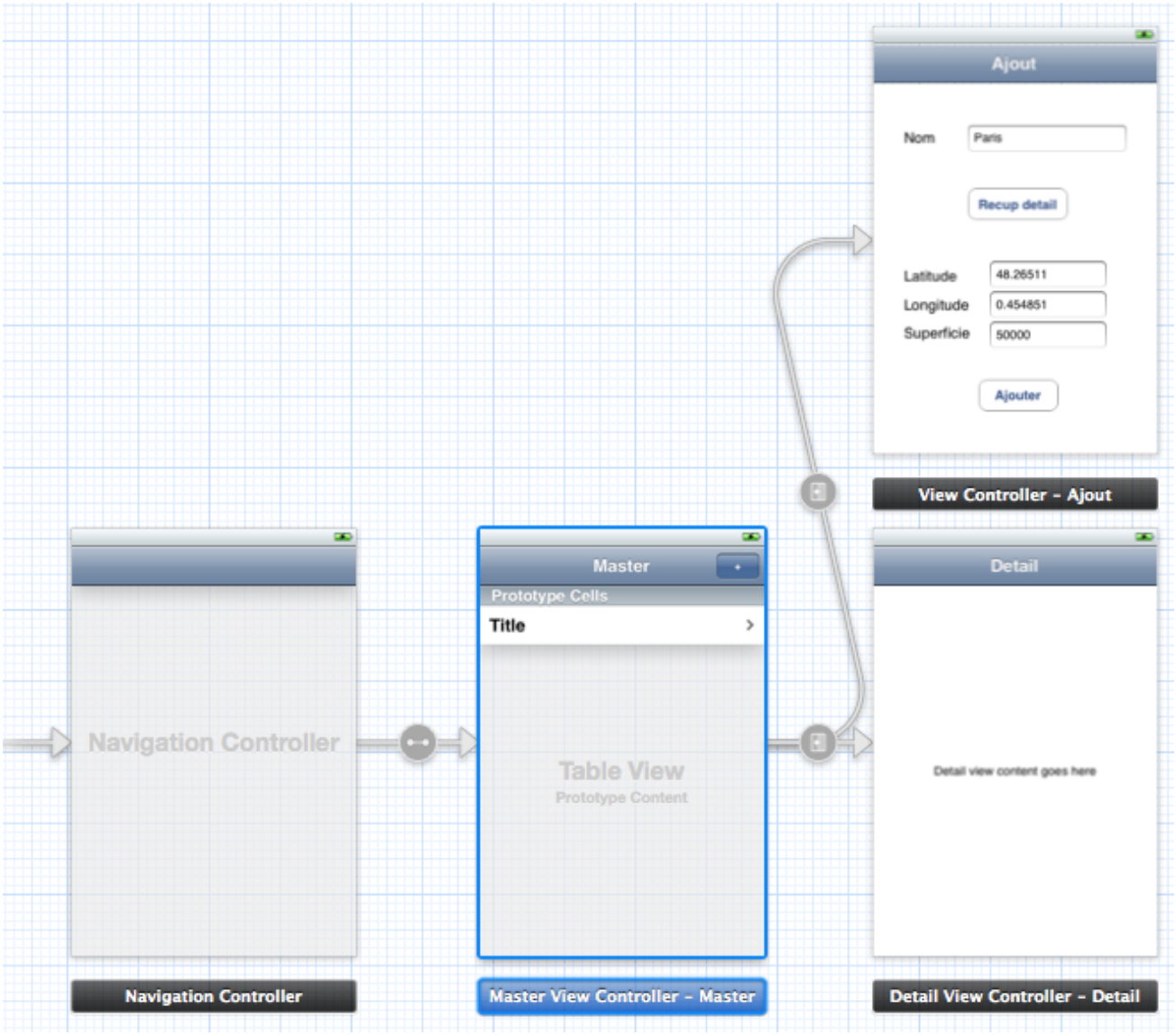
Pour pouvoir contrôler les actions de la nouvelle vue, il faut créer le contrôleur associé :

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

- Ensuite, via l'interface (Barre latérale droite->3ème onglet "Identity inspector"), définir la propriété **Custom class** de la nouvelle vue comme étant de type **AjoutViewController.**



Vous devez avoir un story-board semblable :



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

Etape 2 : Gestion de la liste des villes (<u>vue/contrôleur Master</u>)

Pour manipuler nos informations sur les villes, nous allons créer une classe **Ville**.

- Ajouter au votre projet un classe **Ville (**File->New file->**Objective-C class)**.
- Implémenter dans la classe **Ville** les 4 attributs suivants : nom (NSString *), latitude (float), longitude (float) et superficie (float).
- Implémenter 2 méthodes :
    - 1 constructeur permettant l'instanciation de la ville (en passant le nom en paramètre)
    - 1 méthode pour mettre à jour les paramètres: latitude, longitude et superficie

<u>RAPPEL:</u>

<u>Dans le .h</u>

@interface Ville : NSObject {
liste des attributs
}

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

liste des méthodes

@end

<u>Dans le .m</u>

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

définition des méthodes *déclarés dans le .h*

- instancier dans **MasterViewControleur.m**, une nouvelle **Ville** manuellement dès la fin du chargement de la vue en utilisant la méthode constructeur de **Ville**. Attention à bien inclure « Ville.h »

```
ex: Ville * nouvelleVille = [[Ville alloc] initWithName:@"Paris"];
```

Nous allons maintenant gérer l'ajout & l'affichage des **Villes** dans le **Tableview**.

- Renommer la méthode **insertNewObject** en **insertNewVille** et l'adapter afin qu'elle prenne en paramètre un objet de type **Ville *** pour l'ajouter dans le conteneur "_object" déjà prévu à cet effet. A la fin de la méthode, ajouter **[self.tableView reloadData]** pour prévoir le rafraichissement du **tableView**.

- Adapter la méthode **cellForRowAtIndexPath** pour gérer l'affichage des villes dans le **tableView**. Définir une entrée dans le **TableView** de manière à avoir un libellé principal (nom de la ville), et d'un sous-libellé (superficie). Attention, pour afficher la superficie, il faudra convertir le nombre en chaine de caractère. Exemple : **NSString** *str = [NSString stringWithFormat:@"%f", myFloat];

--------- FAIRE VALIDER L'ETAPE 2 PAR L'ENSEIGNANT ---------

Etape 3 : Gestion de l'ajout d'une ville (vue/contrôleur Ajout).

Cette vue permet d'ajouter une ville en renseignant les informations nécessaire. Dans un premier temps (étape 3a), l'utilisateur entrera toutes les informations relatives à la ville (nom, longitude, latitude, superficie) dans des champs prévus à cet effet. Dans un second temps (étape 3b), l'utilisateur pourra se contenter d'entrer seulement le nom de la ville. L'application remplira automatiquement les autres champs en allant chercher les informations relatives sur Wikipedia.

Etape 3a : Ajout d'une ville manuellement

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

- Pour ajouter une nouvelle ville au **tableView**, passer par la méthode **insertNewVille** (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 (voir "**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;
    }
    ```
  - Ensuite, dans l'interface, 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 destinationViewControlleur] setMC:self];
    }
    ```

- Appeler ensuite la méthode **insertNewVille** via le **MasterViewController** (que vous avez référencé dans la variable locale **"_mc"** précédament) en lui passant en paramètre l'instance de la **Ville** que vous avez crée.

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

--------- *FAIRE VALIDER L'ETAPE 3a PAR L'ENSEIGNANT* ---------

Etape 3b : Récupération d'info sur le web

- Utiliser le bouton « Recup. Détails » pour automatiser la récupération de code source HTML de la page http://fr.wikipedia.org/wiki/nomVille, où « nomVille » est le nom de la ville que l'utilisateur aura renseigné dans le champ correspondant. Vous pouvez utiliser la classe **RequeteManager** présentée en annexe (voir lien donnée en TP) pour gérer la connexion à une ressource web. Dans un premier temps, vous pouvez afficher le code HTML récupéré dans la console (via **NSLog**). Eviter dans un premier temps les cas compliqués, comme les noms de villes avec des espaces (ex: "La Rochelle")...

- Trouver les 3 expressions régulières qui permettent d'isoler les informations (latitude, longitude et superficie) à partir du code source HTML. Une aide mémo sur les expressions régulières est fournie en annexe.

En guise d'exemple, voici la première expression régulière (formaté pour Objective C) permettant de récupérer la latitude :

```
REGEXP :  @"params=\\d+\\.\\d+_._"

=> info trouvé dans un lien dans le code HTML de la page wikipedia
pointant vers "toolserver.org/~geohack/geohack.php?..."
```

Attention à bien utiliser \\ à la place de \ dans la définition des expressions régulières en objective C.

- Utiliser les expressions régulières avec la classe **NSRegularExpression** pour récupérer les infos (latitude, longitude et superficie) et remplir automatiquement les 3 champs de saisie correspondants.

N'hésitez pas à vous inspirez d'exemples trouvés sur internet !

--------- *FAIRE VALIDER L'ETAPE 3b PAR L'ENSEIGNANT* ---------

Etape 4 : Gestion de la carte (vue/contrôleur Detail Map)

Ajoutez la référence « **MapKit** » à votre projet.

- Ajouter un contrôle de type **MKMapView** sur la vue **Détail**.

- Dès le chargement de la vue terminé, faire que la carte se centrer sur la latitude/longitude de la ville sélectionnée dans la liste.

- Zoomer sur une zone de 1km².

- Ajouter une punaise (via **MKAnnotation**) à l'emplacement de la ville. (plus d'infos dans le TP précédant)

- Faire apparaitre les données sur la ville (ex : Nom de la ville & superficie) lors du clique sur une punaise.

--------- *FAIRE VALIDER L'ETAPE 4 PAR L'ENSEIGNANT* ---------

Etape bonus

Q19 : Récupérer le nombre d'habitants de la ville sur Wikipédia et l'afficher sur la carte.

--------- *FAIRE VALIDER LE BONUS PAR L'ENSEIGNANT* ---------

Annexes :

Connexion à un serveur Web

*La classe **RequeteManager** récupère le code source HTML d'une URL de manière asynchrone. La réponse est renvoyée via la méthode déléguée **requeteManagerResponse**. Exemple : Un contrôleur implémente le protocole <**RequeteManagerDelegate**>. 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):*

```objc
// ----- 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

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

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

}
```

EXEMPLE DE REPONSE

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

```objc
// --------- 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 {
    [connection release];
    [receivedData release];
    NSLog(@"Erreur");
}

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

Expressions régulières

Les métacaractères / ancres

- ^ accent circonflexe
  marque le début d'une chaîne(voir classe aussi)
- $ dollar
  marque la fin d'une chaîne

Memo:
^chat$ reconnaît chat seul
^$ reconnaît chaîne vide
^ début de chaîne
$ fin de chaîne

L'alternative

- | barre verticale
  marque l'alternative

Memo:
L[yi]s reconnaît Lys ou Lis
^(De|Sujet|Date):@ reconnaît tout ce qui commence par De:@ ou Sujet:@ ou Date:@

Les quantificateurs

- ? point d'interrogation
  Facultatif
  zéro ou une occurrence
- * étoile
  Facultatif
  zéro, une ou plusieurs occurrences
- + signe plus
  Obligatoire
  une ou plusieurs occurrences
- {x}accolade + nombre
  Obligatoire restrictif
  doit apparaître exactement x fois
- {x,}accolade + nombre
  Obligatoire non restrictif
  doit apparaître au moins x fois
- {x,y}accolade + nombre
  Obligatoire restrictif
  doit apparaître exactement x fois et maximum y fois

Memo:
a? reconnaît 0 ou 1 a
a* reconnaît 0 ou plusieurs a
a+ reconnaît 1 ou plusieurs a

Les classes de caractères

- [ ] les crochets
  indique une classe
- - le tiret
  indique l'intervalle dans une classe

Memo:
[a-z] reconnaît les lettres de a à z
[Yy]ves un mot avec ou sans majuscule
<h[1-6]> une balise de titre par exemple

La classe complémentée

- [^...] au lieu de [...]
  indique une classe complémentée
  reconnaît tout caractère qui n'est pas énuméré

Memo:
[^0-9] reconnaît tout ce qui n'est pas des chiffres
[^1-6] reconnaît tout sauf les chiffres de 1 à 6
Rappel : l'accent circonflexe est un métacaratère à l'intérieur de la classe. A l'extérieur, c'est une ancre qui signifie le début de...

Le tiret

- - indique un intervalle dans une classe [0-9]

  Rappel: Le tiret est un métacaractère à l'intérieur d'une classe à condition qui exprime bien un intervalle.
  Pour utiliser le tiret en tant que littéral à l'intérieur d'une classe, soit le placer au début, soit en fin de classe [-0-9] ou [0-9-]
  A l'extérieur d'une classe le tiret est un caractère normal.

Classe abrégée	Correspondance
\d	[0-9]
\D	[^0-9]
\w	[a-zA-Z0-9_]
\W	[^a-zA-Z0-9_]
\t	Tabulation
\n	Nouvelle ligne
\r	Retour chariot
\s	Espace blanc (correspond à \t \n \r)
\S	Ce qui n'est PAS un espace blanc (\t \n \r)
.	Classe universelle

Testez vos expressions régulières sur http://regexpal.com/