Dossier technique

# Présentation du projet

# Tâches réalisées

# Android :

## Fonctionnalités de l’application

## Description de l’interface

## Règle de nommage

## Description des objets

## Description des algorithmes

## Fonctionnement de l'application

# iOS

## Fonctionnalités de l’application

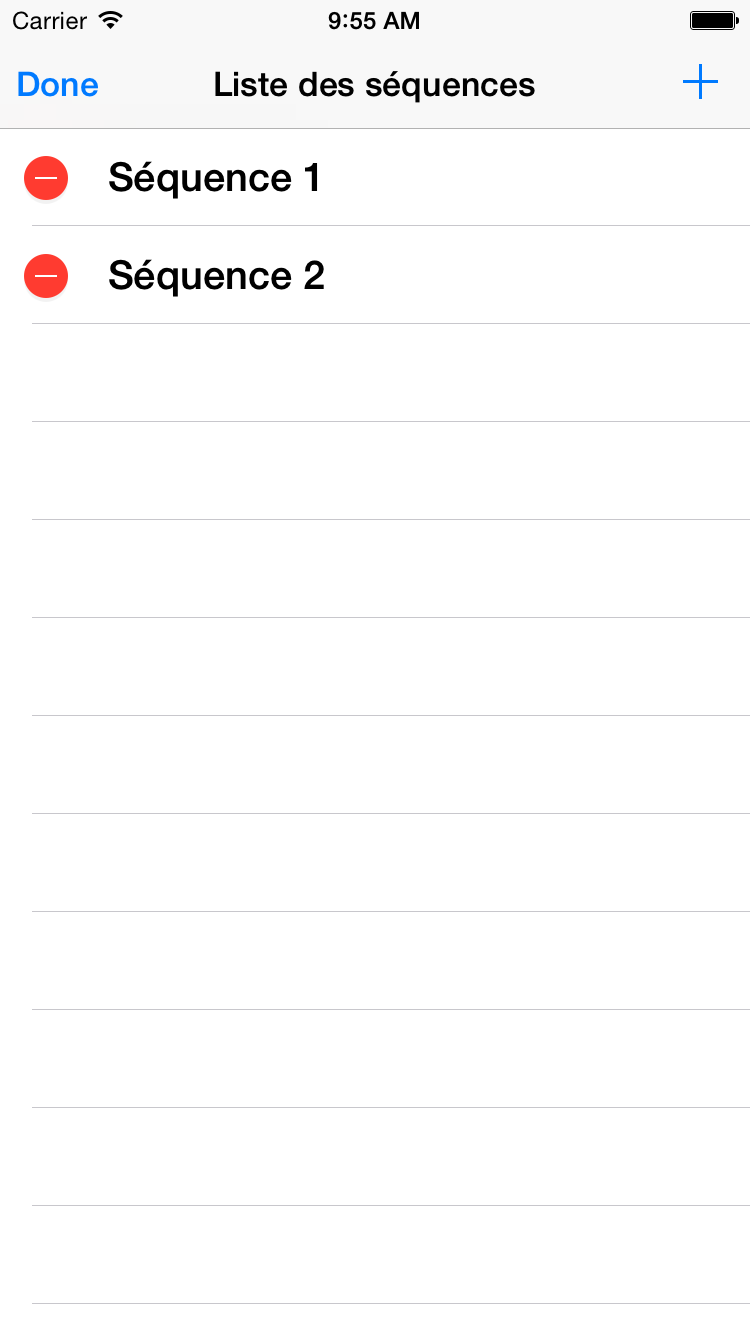
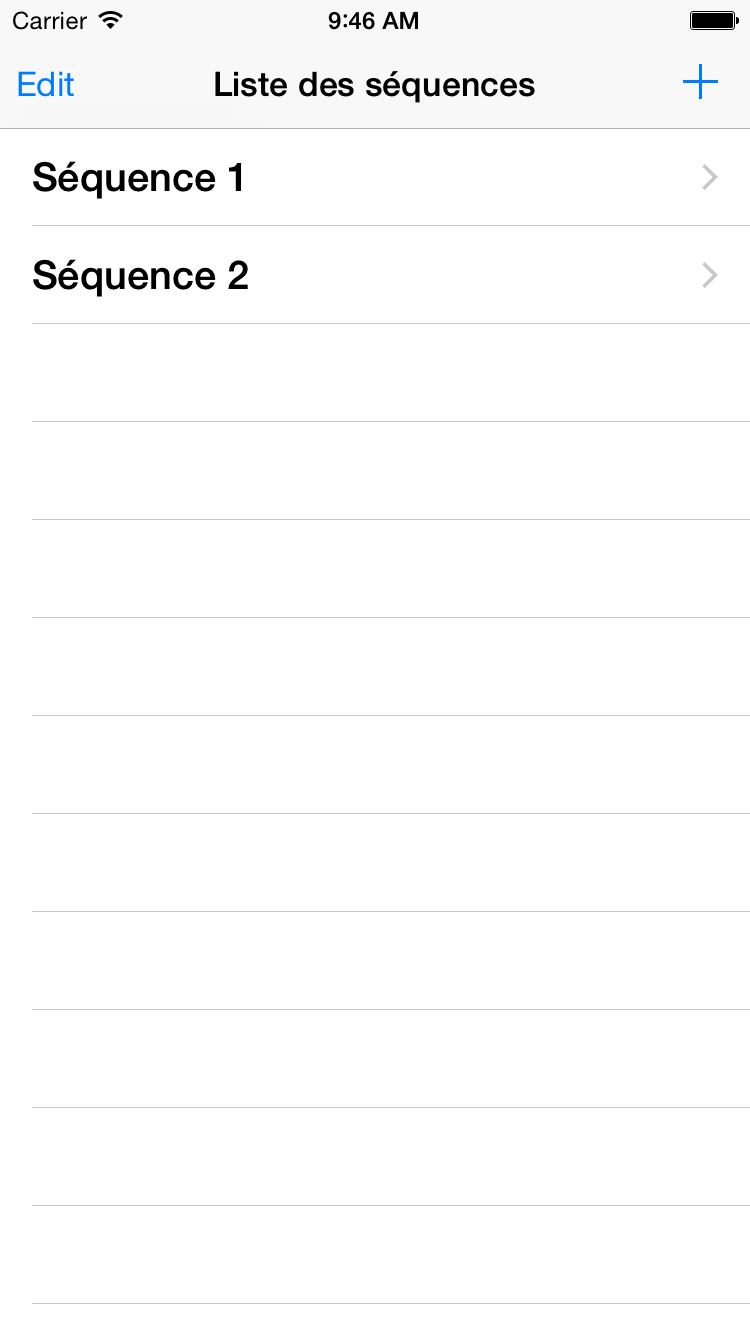
Comme convenu dans le cahier des charges, l’application ne sera pas entièrement porté sur iOS. Les fonctionnalités de l’application sur iOS seront les suivantes.

* Affichage de la liste des séquences
* Edition d’une séquence
* Edition d’un exercice
* Mémorisation des modifications dans la base de donnée interne du Smartphone
* Utilisation du vibreur

## Description de l’interface

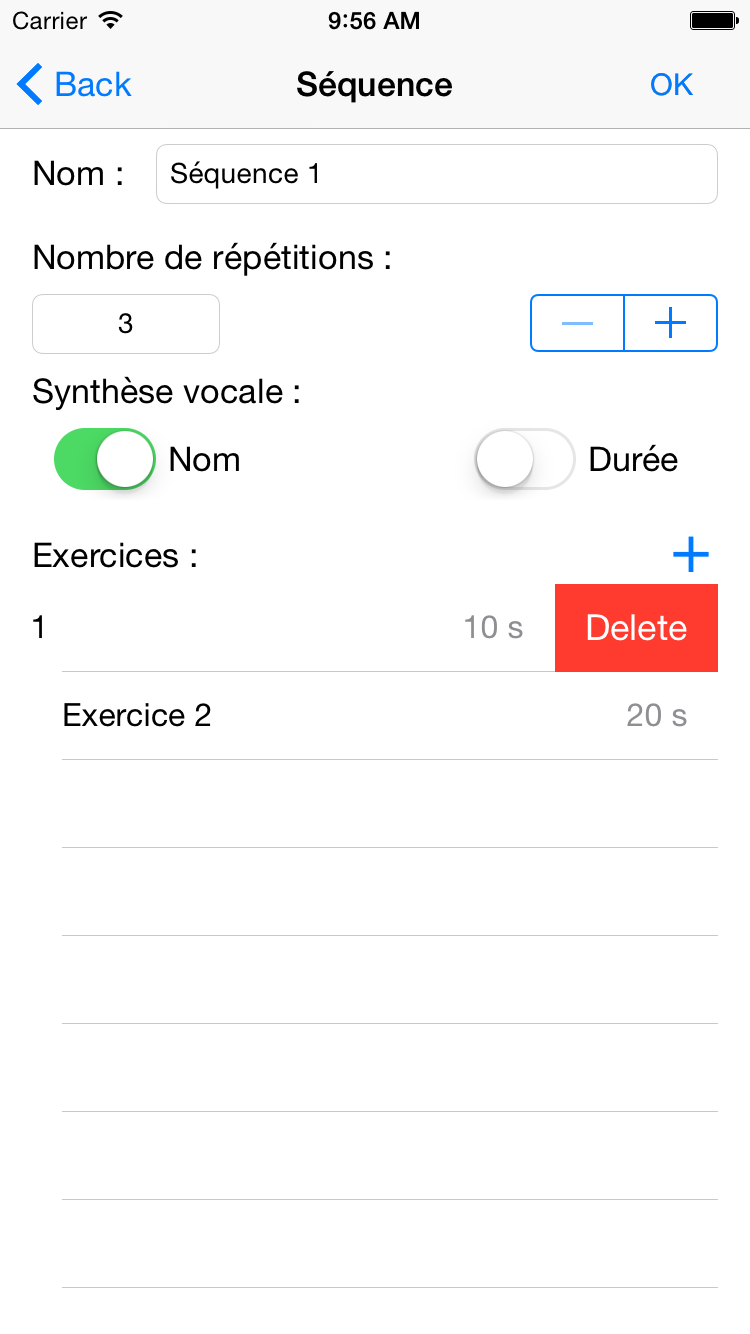
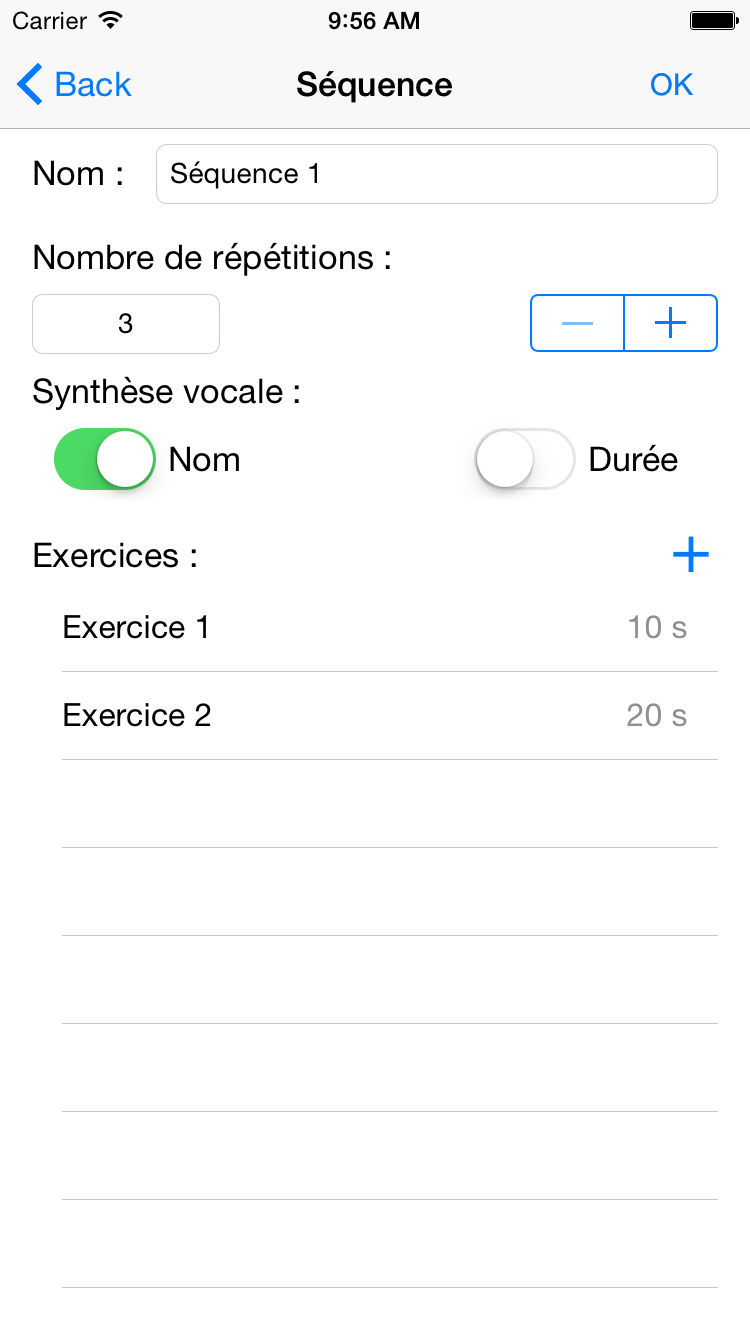
L’application sera composée de trois fenêtres

### Liste des séquences



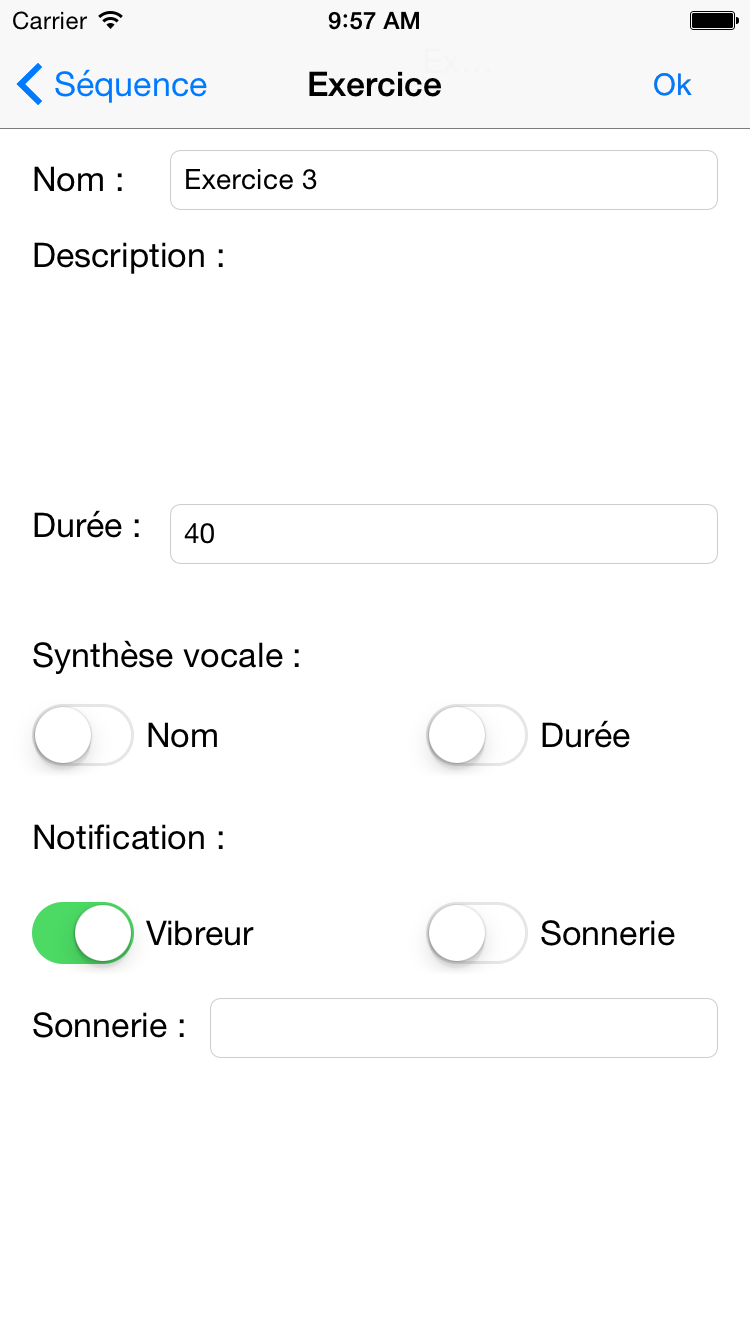
Dans cette vue, l’utilisateur peut lister les séquences enregistrées dans la base de donnée et les supprimer.

### Edition de la séquence



Cette vu permet l’édition des caractéristiques d’une séquence. On peut y modifier le nom, le nombre de répétitions, le type de synthèse vocale et on aussi peut y ajouter ou supprimer un exercice. L’appui sur le bouton OK permet d’enregistrer les modifications dans la base de donnée et le bouton back permet de retourner à la fenêtre précédente. Un appui sur un exercice permet d’accéder à la fenêtre d’édition de l’exercice et un glissement du doigt permet de supprimer un exercice.

### Edition de l’exercice



Dans cette fenêtre, l’utilisateur peut modifier les détails d’un exercice.

* Le nom
* La description
* La durée en seconde
* Le type de synthèse vocale
* Le type de notification
* Le nom du fichier de la sonnerie

Un appui sur le bouton OK permet d’enregistrer les modification et un appui sur le bouton Séquence permet de retourner à la fenêtre d’édition de la séquence.

## Règle de nommage

Les règles de nommage utilisées sont les suivantes :

* Pour les classes :
  + Nom de la classe commençant par une majuscule
  + Données membres de la classe commençant par la lettre m suivit d’une écriture du nom de la donnée membre en Camel case
  + Fonctions commençant par une lettre minuscule et nom parlant de la fonction en Camel case
* Pour les variables internes :
  + nom clair et parlant en Camel case et commencant par une lettre en minuscule
  + éviter les caractères numériques et spéciaux dans les noms
* Pour les classes d’interface
  + Nom clair de la classe
  + Le nom se termine par View ou ViewController

## Description des objets

### Classes Model





Ces classes permettent la mise en forme des données .

### Classe VueModel

#### MasterViewController



Cette classe contient le code utilisé par la première fenêtre de l’application. Elle permet de récupérer la liste des séquences de la base de donnée et gère son affichage et ses modifications.

#### EditionSequenceView



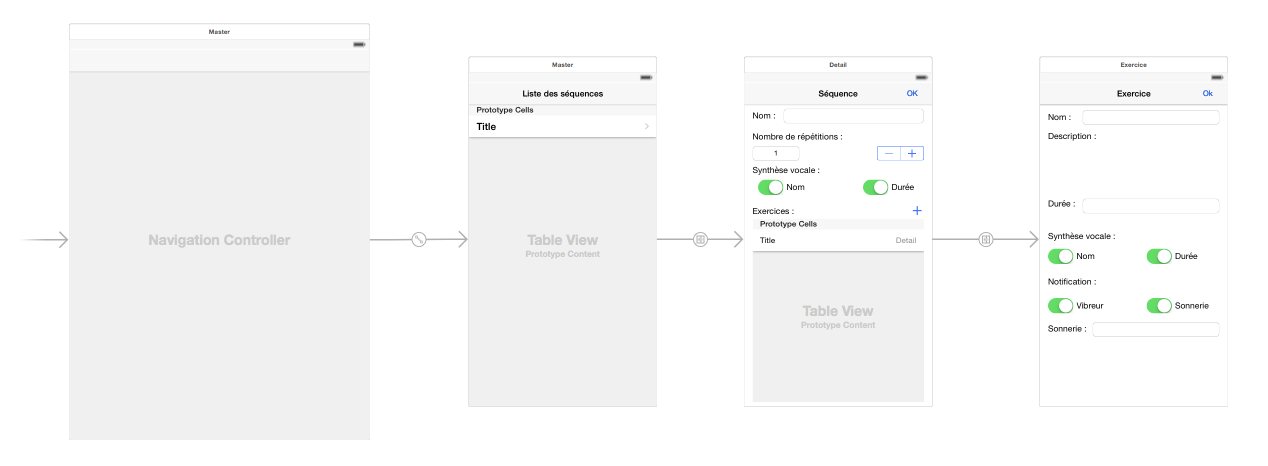
Cette classe gère l’interface d’édition de séquence. Elle permet l’affichage et la modification des caractéristiques d’une séquence. Elle permet aussi de lister, d’ajouter, de supprimer et de modifier les exercices composant une séquence.

#### EditionExerciceView



Cette classe gère l’interface d’édition de l’exercice, elle permet l’affichage et la modification des caractéristiques d’un exercice.

### Storyboard Vue



Ce storyboard représente la partie Vue de mon application. Il regroupe les différentes fenêtres, les contrôles qui les composent et les interactions entres elles.

Il est composé d’un Navigation Controller qui sert de base à l’interface et de plusieurs fenêtres :

* Liste des séquences
* Séquence
* Exercice

Les flèches entre les différentes interfaces sont des Segues , ils permettent de définir un lien entre deux interfaces et la façon dont se fait la transition entre ces deux interfaces.

## Description des algorithmes

#### Utilisation du vibreur

- (IBAction)onSwVibreurChange:(id)sender {

if(swNotifVibreurEx.on)

{

AudioServicesPlaySystemSound(kSystemSoundID\_Vibrate);

}

}

Ce code permet d’utiliser le vibreur quand l’utilisateur active la notification par vibreur dans le code.

#### Ajout d’un nouvel exercice à une séquence

- (IBAction)onBtnAddClick:(id)sender {

   NSManagedObjectContext \*context = [self.fetchedResultsControllerEx managedObjectContext];

   NSEntityDescription \*entity = [[self.fetchedResultsControllerEx fetchRequest] entity];

   NSManagedObject \*newExercice = [NSEntityDescription insertNewObjectForEntityForName:[entity name] inManagedObjectContext:context];

   [newExercice setValue:@"Nouvel exercice" forKey:@"nomEx"];

   [newExercice setValue:@"" forKey:@"descriptionEx"];

   [newExercice setValue:@10 forKey:@"dureeEx"];

   //récupération de la position de l'exercice dans la liste d'exercices de la séquence

   id <NSFetchedResultsSectionInfo> sectionInfo = [self.fetchedResultsControllerEx sections][0];

   long nbreExercicesDansSequence = [sectionInfo numberOfObjects];

   [newExercice setValue:[NSNumber numberWithLong:nbreExercicesDansSequence] forKey:@"positionDansSequenceEx"];

   //mise en place de la relation entre l'exercice créé et la séquence dans detailItem

   [newExercice setValue:self.detailItem forKey:@"sequence"];

   NSMutableSet \*exercices = [self.detailItem mutableSetValueForKey:@"exercices"];

   [exercices addObject:newExercice];

   //sauvegarde les données mises à jour

   NSError \*error = nil;

   if (![self.managedObjectContext save:&error]) {

       // Replace this implementation with code to handle the error appropriately.

       // abort() causes the application to generate a crash log and terminate. You should not use this function in a shipping application, although it may be useful during development.

       NSLog(@"Unresolved error %@, %@", error, [error userInfo]);

       abort();

   }

   [self.lstExercices endUpdates];

}

Ce code permet l’initialisation et l’ajout d’un exercice dans la séquence actuelle et le sauvegarde dans CoreData.

#### Mise à jour des caractéristiques d’une séquence

- (IBAction)onBtnOkClick:(UIButton \*)sender {

   //met à jour les données dans self.detailItem

   [self.detailItem setValue:txtNomSeq.text forKey:@"nomSeq"];

   long nbreRepetitions =[txtNbreRepSeq.text integerValue] ;

   [self.detailItem setValue:[NSNumber numberWithLong:nbreRepetitions] forKey:@"nombreRepetitionsSeq"];

   if(swSynthNomSeq.on)

   {

       [self.detailItem setValue:@true forKey:@"synthVocNomSeq"];

   }

   else

   {

       [self.detailItem setValue:@false forKey:@"synthVocNomSeq"];

   }

   if(swSynthDureeSeq.on)

   {

       [self.detailItem setValue:@true forKey:@"synthVocDureeSeq"];

   }

   else

   {

       [self.detailItem setValue:@false forKey:@"synthVocDureeSeq"];

   }

   //sauvegarde les données mises à jour

   NSError \*error = nil;

   if (![self.managedObjectContext save:&error]) {

       // Replace this implementation with code to handle the error appropriately.

       // abort() causes the application to generate a crash log and terminate. You should not use this function in a shipping application, although it may be useful during development.

       NSLog(@"Unresolved error %@, %@", error, [error userInfo]);

       abort();

   }

}

Cette fonction est appelée quand l’utilisateur clique sur le bouton OK. Elle met à jour les caractéristiques de l’exercice et sauvegarde les modifications dans CoreData.

## Fonctionnement de l'application

A son lancement, l’application va récupérer les séquences enregistrées dans CoreData via la fonction *fetchedResultsController.* Une fois les séquences récupérées, elles sont ajoutées dans la *UITableView*.

Lorsque l’utilisateur supprime une séquence, la suppression est automatiquement transmise à CoreData en plus de la *UITableView.*

Quand l’utilisateur sélectionne une séquence, l’application change de fenêtre et affiche la fenêtre d’édition de séquence en lui transmettant la séquence à modifier dans *detailItem*. L’app va récupérer les caractéristiques de la séquence ainsi que les exercices la composant, via les *NSFetchedResultsController* *fetchedResultsControllerSeq* et *fetchedResultsControllerEx* et met à jour l’affichage.

Un clic sur un exercice ouvre la fenêtre d’édition d’exercice en transmettant l’exercice dans *detailItem*. L’app va ensuite récupérer les caractéristiques de l’exercice dans CoreData via *fetchedResultsControllerEx* et les afficher.

Si l’utilisateur modifie les caractéristiques d’un exercice ou d’une séquence, les changements sont automatiquement répercutés à CoreData via le *NSFetchedResultsController* correspondant.

# Windows Phone

## Fonctionnalités de l’application

## Description de l’interface

## Règle de nommage

## Description des objets

## Description des algorithmes

## Fonctionnement de l'application

# Remarques personnelles