Sommaire :

Introduction :

Dans le cadre du cours sur l'introduction aux nouvelles technologies, nous avons choisi de développer une application IPhone pour le blog – http://www.reelgeneve.com/ -du journal des lettres R.E.E.L.

Le but était de récupérer les articles sur le site afin qu'ils soient tous accessibles facilement et rapidement sur l'IPhone. Afin de déterminer l'interface à concevoir, nous avons eu des contacts avec des personnes au sein du comité du journal qui nous ont proposé quelques éléments à intégrer à l'application.

Il est utile de rappeler qu'initialement, l'application devait être conçue sous Android. Malheureusement, par manque de documentation, nous avons décidé de changer de plateforme. Nous avons donc du implémenter notre application avec le langage exigé par Apple : « objective-c », qui est un dérivé du langage C. Nous avons aussi du nous procurer le programme permettant de développer des applications IPhone : « x-code », accessible gratuitement dans l'Appstore.

Afin de créer notre application, nous avons du nous poser la question de définir par quel moyen récupérer les informations sur le blog wordpress du journal. Notre première idée fut d'accéder à la base de donnée du blog contenant les articles. Cependant, nous avons rapidement compris que cette méthode n'était pas préconisée par Apple. Nous avons donc changé de direction pour nous orienter vers un analyseur syntaxique qui récupère les informations désirées, contenues dans les balises d'un document XML. Le rôle de l'algorithme étant, d'isoler les informations en fonction de leur contenu et de leur localisation dans le document grâce aux balises de début et de fin. En anglais, nous appelons cet outil un : « parseur ».

- **Diagramme des cas d'utilisations :**

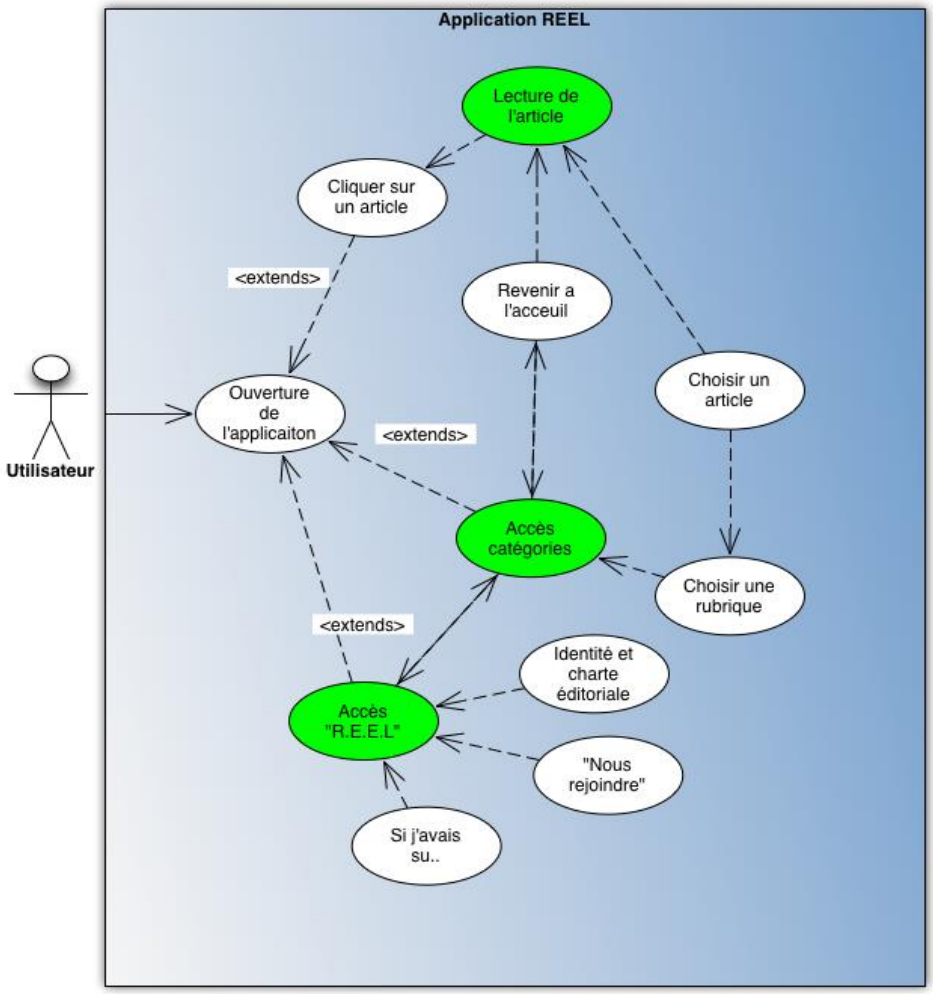
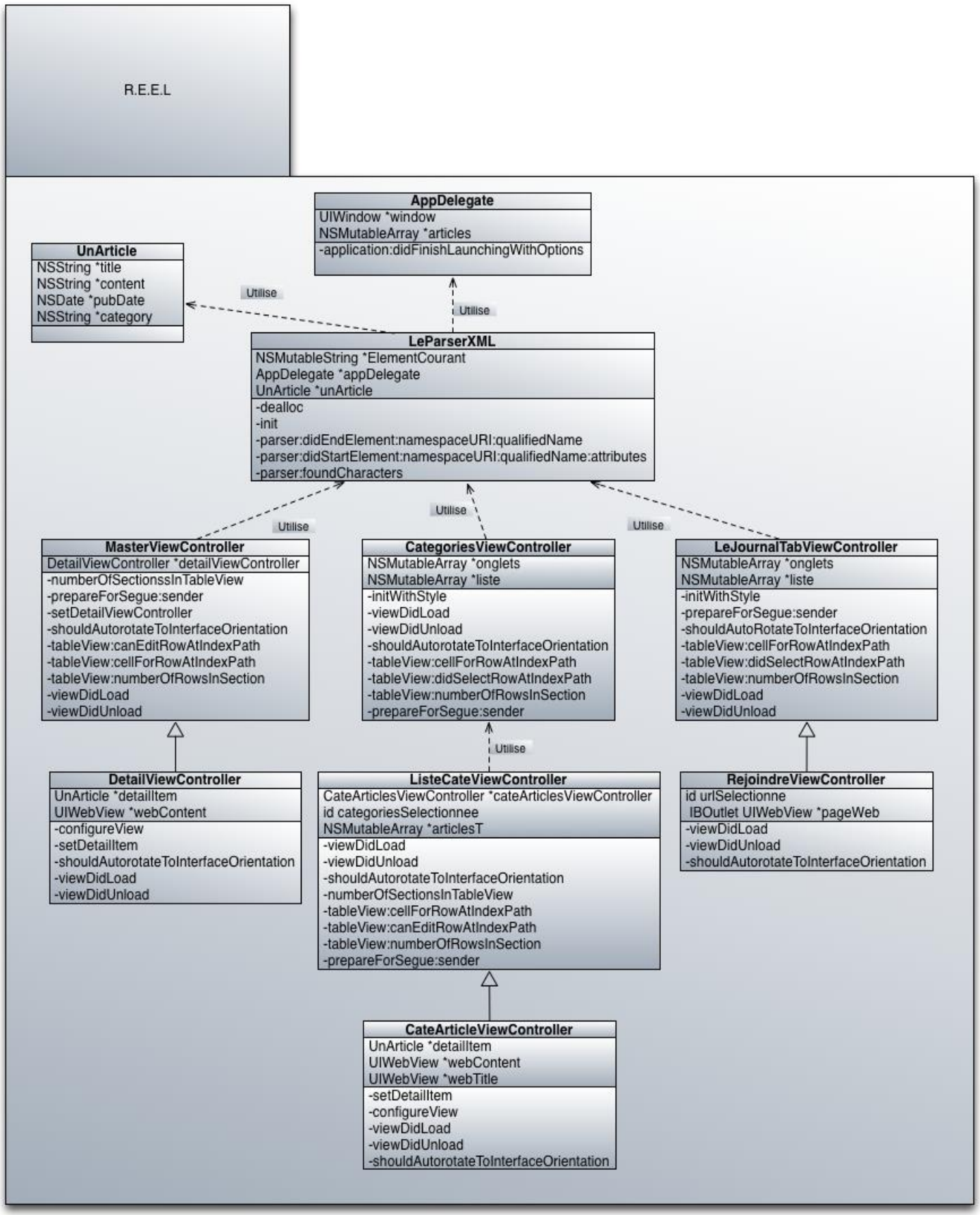


Diagramme des classes



Application PowerPoint :

Avant de réfléchir à l'algorithme, nous avons décidé de créer une version Powerpoint de notre application, afin de donner les premières pistes de l'aspect de notre future interface. Cela nous a été très utile, car nous avons ainsi pu observer si l'application était intuitive, rapide et claire.

Exemple d'un slide du PowerPoint :

Outils NTIC utilisés :

Comme nous l'avons introduit plus haut, nous avons utilisé un parser XML –celui fourni par le framework de base d'Apple - afin de rechercher les divers éléments utiles à notre application. Ces éléments sont des chaines de caractères contenues dans des balises xml. La forme du fichier xml dans un blog

Wordpress est la suivante (nous présentons ici uniquement les balises que nous avons parsé) :

<item>

<title> un titre d'article </title>

<pubDate> une date</pubDate>

<category> un nom de catégorie</category>

<content : encoded> le contenu de l'article </content : encoded>

</item>

L'idée principale de l'algorithme est de parcourir le fichier xml et lorsqu'il trouve une balise ouvrante <item>, il instancie un tableau dynamique d'articles. A partir de ce moment, on cherche dans un item toutes les balises qui définissent un article : <title>, <pubDate>, category> et <content : encoded>. Enfin, pour déterminer la fin d'un article, on recherche les fermetures de chaque balise, ainsi que celle d'un item. Le tableau final contient donc plusieurs articles, chacun avec les balises qui les définissent.

- **Description du programme :**

Afin de créer notre application, nous avons du utiliser le programme fourni par Apple, X-Code. Le langage, Objective-C, qui est un dérivé du C, est un langage orienté objet. Notre application est par conséquent constituée de plusieurs classes qui dialoguent entre elles. Comme en Java, la notion d'héritage est primordiale et est utilisée à tout moment. Notre travail s'articule donc en diverses classes dont les rôles sont bien définit et qui s'échangent des informations.

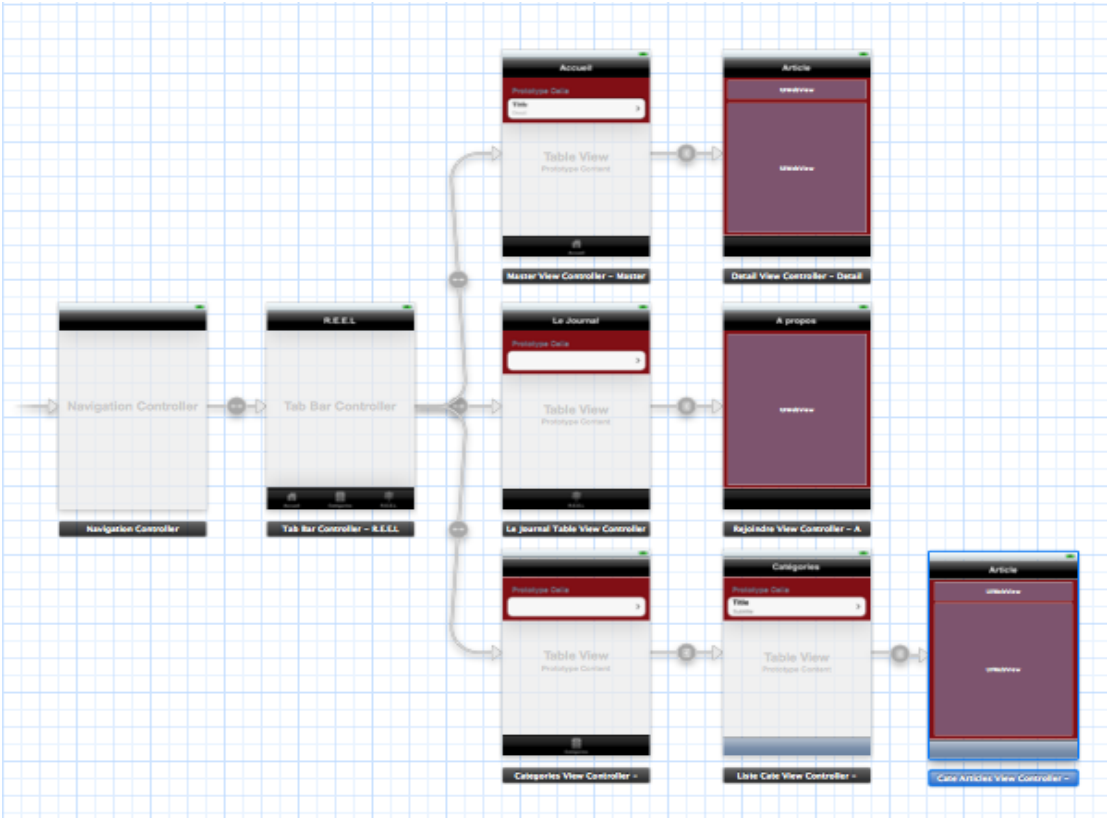
La première dont nous allons discuter s'appelle AppDelegate et son utilité est primordiale dans n'importe quelle application IPhone. Elle sert de lanceur et

initialise le programme. Dans notre cas, elle permet aussi de charger l'adresse URL qui contient le flux XML à parser.

Viennent ensuite les classes contenant l'interface graphique. Celles qui apparaissent à l'ouverture de l'application sont de type UITableViewController. Cela signifie qu'elles contiennent une vue avec divers contrôles qu'elles vont distribuer par la suite à d'autres classes de détail. Dans notre cas, ces classes contiennent une vue avec des cellules remplies dynamiquement avec des chaines de caractères, comme par exemple, le titre d'un article. Il y a aussi les classes de type UIViewController. Celles-ci ne contiennent pas d'informations tabulaires comme les dernières, mais permettent d'afficher des données sur toute la surface de la vue. Nous les utilisons par exemple, pour afficher le contenu d'un article. Enfin, nous avons des classes de type NSObject qui servent à définir des objets qui seront instanciés dans nos vues.

- **Description de l'interface administrateur et utilisateur :**

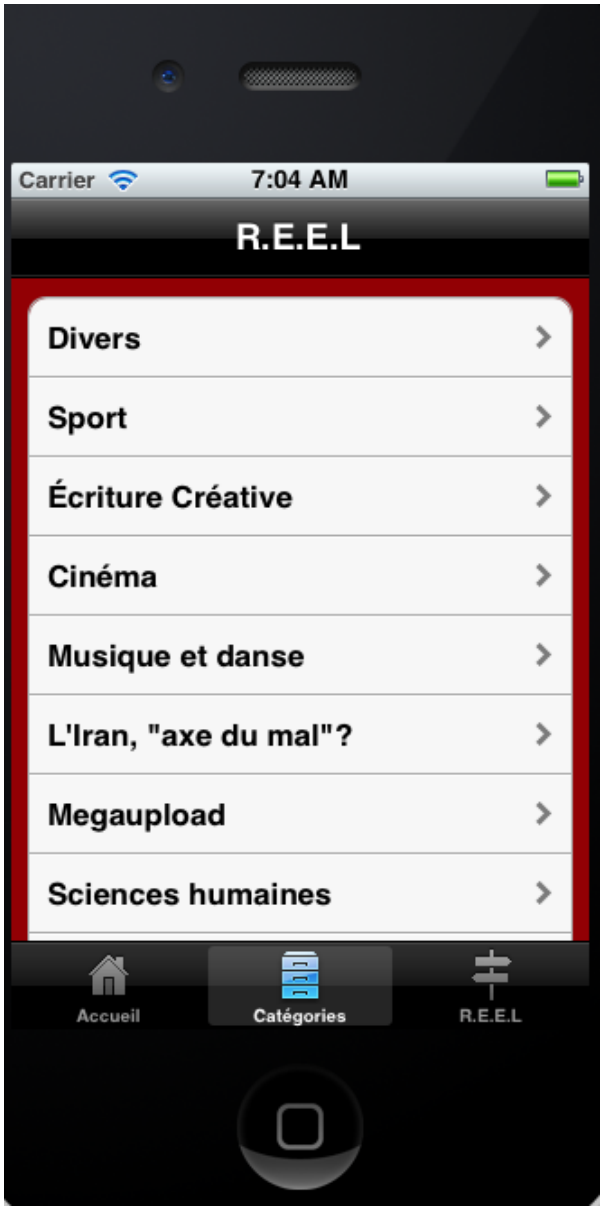
  Voici une capture de l'interface administrateur :



On peut voir que les vues sont reliées entre elles. Ces liens, sont soit des liens de relations qui permettent de transmettre des contrôles entre les vues, soit des passages d'une vue à l'autre. Par exemple, le TabBarController est en relation avec les tables master, rubriques et leJournal, auxquelles elle transmet la barre en bas de la vue. En revanche, le lien entre la vue master et la vue détail est un lien de passage de la première à la seconde, l'idée étant de cliquer sur la cellule pour accéder à la vue de détail.

Concernant l'interface utilisateur, il n'y a que peu de différence, si ce n'est que les vues et les cellules sont remplies de texte.

Un exemple ici :



- **Les difficultés rencontrées :**

Créer le parseur fut très difficile, car il en existe de nombreux, chacun avec ses spécificités. Après avoir réussi le parsing, la difficulté principale fut d'extraire les catégories avec leurs articles respectifs. Pour classer par catégorie, nous avons regardé pour chaque article, dans quelle catégorie il se trouvait et l'avons classé dans un tableau correspondant à la catégorie désirée. Un autre problème survenu tout au long du projet, fut la difficulté de naviguer entre les classes. En effet, chacune est constitué en deux partie, une entête contenant les déclarations des variables et un corps, contenant l'implémentation des diverses méthodes.

- **Conclusions et perspectives d'avenir:**

Pour terminer, nous avons pu remarquer que créer une application IPhone, bien que rudimentaire, prend un temps conséquent. En effet, étant donné que les applications pour IPhone ont une grande côte depuis quelques années, il existe de nombreuses manières de parvenir au même résultat. La difficulté est donc croissante en rapport à la documentation disponible. Le résultat atteint pour notre application, est par conséquent à certains niveaux, en deçà de notre idée de départ. La base désirée, est tout de même acquise. Il ne manque que certains détails qui permettraient un peu plus de fluidité et d'interactivité à l'utilisateur. Pour citer quelques exemples d'améliorations possibles, on peut imaginer une interface plus conviviale, avec des éléments plus structurés, ou encore d'autres onglets avec des fonctions qui peuvent être utiles à l'utilisateur.
Pour terminer, nous sommes satisfaits du résultat final, puisqu'il correspond dans les grandes lignes à notre idée de base. Quand au nombre d'heures que nous avons passé pour effectuer ce travail, on peut aisément prétendre avoir dépassé les 150 heures par personne.