الجمهورية الجزائرية الديمقراطية الشعبية
وزارة التعليم العالي و البحث العلمي

Université Abou Bekr Belkaid
Tlemcen Algérie
جامعة أبي بكر بلقايد
تلمسان الجزائر

République Algérienne Démocratique et Populaire
Université Abou Bakr Belkaid– Tlemcen
Faculté des Sciences
Département d'Informatique

MEMOIRE DE PROJET DE FIN D'ETUDES

Pour l'obtention du diplôme de licence en informatique

Thème

La mise en œuvre de la traduction alphanumérique pour effectuer un appel dans différentes plateformes

Réalisé par :

- **Mr.** Selmi Mohammed El-Amin.
- **Mr.** Yala Mohammed.

Encadré par :

- **Mr.** Lehsaini Mohamed.

Soutenu le : .06. 2015 devant le Jury :

- **Mr. BENAMAR Abdelkrim**
- **Mr. BENZIAN Yaghmoracene**

Année Universitaire : 2014-2015

* بسم الله الرحمان الرحيم *

Remerciements

Avant tout ; nous remercions notre **Dieu** de nous avoir aidé à faire notre thème de fin d'étude et Merci à nos parents qui nous soutiennent bien.

Au terme de ce travail, nous tiens à exprimer notre profonde gratitude et nos sincères remerciements à notre tuteur de notre projet de fin d'études à l'université Abou Bekr Belkaid Monsieur Lehsaini Mohammed qui a accepté d'encadrer nos travaux.

Nous tenons aussi à remercier vivement les membres de jury de leur extrême d'avoir accepté d'évaluer notre travail.

Nous remercierons également tous nos amis de la licence 3 nouveaux et DEUA.

Nos profonds remerciements vont à nos camarades surtout Meziani yasser Benaouda Memchaoui et Yahla Reda.

Nos plus vifs remerciements s'adressent aussi à tout le cadre professoral et administratif de l'Université Abou Bekr Belkaid Tlemcen.

Nos remerciements vont enfin à toute personne qui a contribué de près ou de loin à l'élaboration de ce travail.

* Et Merci*

Glossaire des Acronymes

API	Application Programmable Interface
CLI	Common Language Infrastructure
C #	C Sharp
DLL	Dynamic-Link Library
GUI	Graphical User interface
I/O	input/output
IDE	Integrated Development Environment
IL	Intermediate Language
IOs	iPhone Operating System
LINQ	Language Integrated Query
MVC	Model-View-Controller
MVVM	Model-View-ViewModel
.NET	**.**Network
OOP	Oriented Object Programming
OS	Operating System
PC	Perssonal Computer
PCL	Portable Class Library
SAP	Shared Asset Project
UI	User interface
XAML	eXtensible Application Markup Language
XML	eXtended Mark-up Language

Sommaire

Introduction Générale 1

Chapitre 1 Programmation orientée objet 2

1.1 Introduction 2

1.2 Définition de la POO 2

1.3 Paradigme de la POO 3

1.3.1 L'objet 3

1.3.2 Encapsulation 4

1.3.3 Le typage et le polymorphisme 4

1.3.4 Classe et prototype 5

1.4 Les avantages et les inconvénients de la POO 6

1.4.1 Les avantages de la POO 6

1.4.2 Les inconvénients de la POO 6

1.5 Conclusion 6

Chapitre 2 Xamarin et le concept multiplateforme 7

2.1 Introduction 7

2.2 Le concept multiplateforme 7

2.2.1 Définition 7

2.2.2 Xamarin 8

2.3 C # et .NET 14

2.4 Partage de code 14

2.5 Conclusion 17

Chapitre 3 Développement d'une application multiplateforme (Applr) 19

3.1 Introduction 19

3.2 Application Applr 19

3.2.1 Principe de fonctionnement 19

3.2.2 Outils de développement 20

3.2.3 Description du code de l'application 21

3.2.4 UI user interface 23

3.2.5 Partie interaction avec la UI 23

3.3 Conclusion 25

Conclusion Générale 26

Références bibliographiques 27

Liste des figures

Figure 1: Topologie de la programmation objet 3
Figure 2: Schéma illustre l'objet et les deux modes d'encapsulation 4
Figure 3: Structure d'une classe dans la POO 5
Figure 4: Xamarin utilisant le code pour les multiplateformes [4] 8
Figure 5: les paradigmes d'interface utilisateur dans IOs 9
Figure 6: Les paradigmes d'interface utilisateur sous Android 10
Figure 7: Paradigmes d'interface utilisateur dans Windows 10
Figure 8: Environnement de développement d'IOs 11
Figure 9: Environnement de développement d'Android 11
Figure 10: IDE de Windows phone 12
Figure 11: Une illustration de "UISwitch, Switch et ToggleSwitch" 13
Figure 12: Interrelations entre Visuel Studio et les API 15
Figure 13: Les packages de Xamarin pour résoudre les problèmes de la multiplateforme 16
Figure 14: Les trois plateformes IOs, Androïd et Windows phone 17
Figure 15: L'interface du Microsoft Visual Studio 2013 21
Figure 16: Interface de l'entrée du mot de téléphone sur Windows phone 24
Figure 17: Exemple d'appel téléphonique sur Windows phone 25

Introduction générale

Introduction Générale

La programmation orientée objet a contribué à une expansion significative des applications fonctionnant sur des plateformes différentes. Elle vise à faciliter l'utilisation de ces applications et de réduire leurs prix ainsi que de la rendre accessibles à tous les utilisateurs possédant des dispositifs intelligents. Il existe des entreprises spécialisées dans cette optique telles que Xamarin, [1] qui travaille pour aider les utilisateurs dans la création de leurs applications. Dans le cadre de ce projet de fin d'études nous proposons de développer une application permettant la translation des mots alphanumériques d'un téléphone en des chiffres numériques et cette application est portable sur plusieurs plateformes.

Le travail présenté dans ce mémoire entre dans le cadre de notre projet de fin d'études en cycle de Licence, option informatique générale, à Université Abou Bakr Belkaid de Tlemcen.

Ce travail est une mise en œuvre de la traduction alphanumérique pour effectuer un appel dans plusieurs plateformes indépendamment d'une plateforme bien particulière. Il est organisé en trois chapitres encadrés par une introduction générale et une conclusion.

- Le premier chapitre présente les concepts généraux de la programmation orientée objet.
- Le deuxième chapitre donne un aperçu sur Xamarin et le concept de multiplateforme.
- Le troisième chapitre est consacré à la présentation de notre application Applr d'une façon technique.
- Enfin, nous terminons ce rapport par une conclusion générale.

Chapitre 1

La programmation orientée objet

Chapitre 1
Programmation orientée objet

1.1 Introduction

Aujourd'hui, dans le langage informatique, la programmation orientée objet (POO) consiste en la mise en relation d'objets, c'est-à-dire d'éléments de programmation, avec un langage spécifique qui permet aux objets de communiquer entre eux.

Dans ce chapitre, nous commencerons tout d'abord par donner un aperçu sur la programmation orientée objet qui est le paradigme de programmation sur lequel se base l'application qu'on a développée.

1.2 Définition de la POO

La programmation orientée objet a été introduite par Alan Kay avec Smalltalk. Toutefois, ses principes n'ont été formalisés que pendant les années 80 et surtout en 1990. Par exemple le typage de second ordre, qui qualifie le typage de la programmation orienté objet (appelé Duck Typing par la communauté Ruby et Python), n'a été formulée qu'en 1995 par Cook [1].

La programmation orientée objet (POO) est un type de programmation qui a pour avantage de posséder une meilleure organisation, surtout dans les gros programmes. Ces derniers seront agencés de façon plus logique et seront donc plus facilement modifiables [2].

La programmation objet permet à un objet de raffiner la mise en œuvre d'un message défini pour des objets d'un type parent, autrement dit de redéfinir la méthode associée au message plus simple c'est le principe de redéfinition des messages. La figure 1 illustre la topologie de la programmation orientée objet.

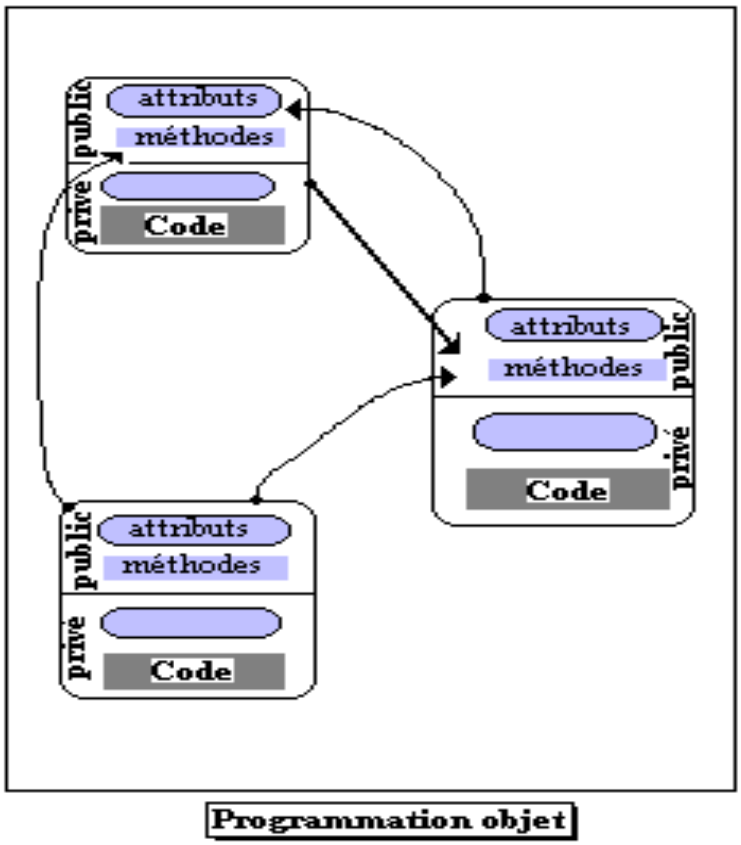

Figure 1: Topologie de la programmation objet

1.3 Paradigme de la POO

En programmation orientée objet, les programmes informatiques sont l'interaction des " objets ". Un objet à " variables d'instance " et " méthodes " valorise le record de variables d'instance associé à l'objet et les méthodes sont des fonctions qui impliquent un certain aspect de l'objet.

Les éléments de base de la POO sont comme suit :

1.3.1 L'objet

Un objet est une structure de données valuées et cachées qui répond à un ensemble de messages. Il est composé d'attributs et de méthodes où les données qui décrivent sa structure interne sont appelées ses attributs. L'ensemble des messages forme ce que l'on appelle l'interface de l'objet.

Certains attributs et/ou méthodes sont cachés, c'est le principe d'encapsulation et le programme peut modifier la structure interne des objets ou leurs méthodes associées sans avoir d'impact sur les utilisateurs de l'objet.

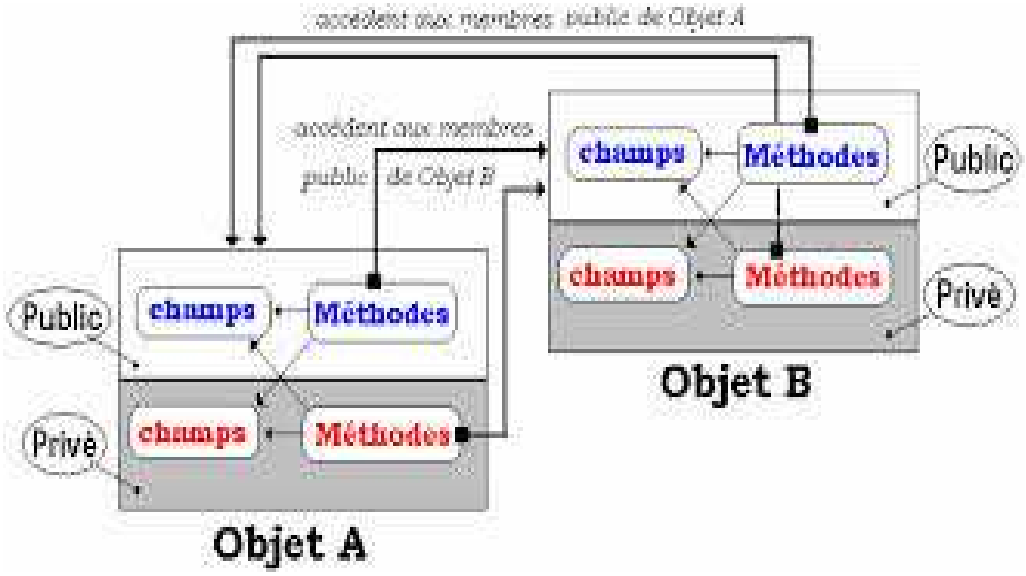

Figure 2: Schéma illustre l'objet et les deux modes d'encapsulation

1.3.2 Encapsulation

C'est le fait de réunir à l'intérieur d'une même entité (objet) le code (méthodes) et les données (champs). Il est donc possible de masquer les informations d'un objet aux autres objets.

Il y a deux niveaux d'encapsulation :

a) Privé

Les champs et les méthodes masqués sont dans la partie privée de l'objet.

b) Public

Les champs et les méthodes visibles sont dans la partie interface de l'objet.

1.3.3 Le typage et le polymorphisme

Le polymorphisme signifie qu'une entité peut prendre plusieurs formes. Cette caractéristique essentielle de la programmation orientée objet la caractérise. On distingue généralement trois types de polymorphisme :

- Le polymorphisme ad hoc (également surcharge)
- Le polymorphisme d'héritage (également redéfinition, spécialisation)
- Le polymorphisme paramétrique (également généricité ou template)

Chaque objet est un type différent c'est-à-dire le type est le nom de sa classe et il définit ses propres attributs et ses méthodes, et il est utile d'un point de vue sémantique dans le code.

Dans la programmation par objet, chaque objet est typé et il y a deux mécanismes de typage :

- le typage dynamique : le type des objets est déterminé à l'exécution lors de la création des dits objets (Smalltalk, CLOS, Python, PHP…).

- le typage statique : le type des objets est vérifié à la compilation et est soit explicitement indiqué par le développeur lors de leur déclaration (C++, Java, C#, Pascal…), soit déterminé par le compilateur à partir du contexte (Scala, OCaml, Haskell…).

De même, deux mécanismes de sous-typage existent : l'héritage de type simple (Smalltalk, Java, C#) et l'héritage multiple (C++, Python, CLOS, Eiffel, WLangage).

1.3.4 Classe et prototype

a) La Classe

La classe est une structure informatique particulière dans le langage objet. Elle décrit la structure interne des données et elle définit les méthodes qui s'appliqueront aux objets de même famille (même classe) ou type. Elle propose des méthodes de création des objets dont la représentation sera donc celle donnée par la classe génératrice.

Une classe est une sorte de moule ou de matrice à partir duquel sont engendrés les objets réels qui s'appellent des instances de la classe considérée.

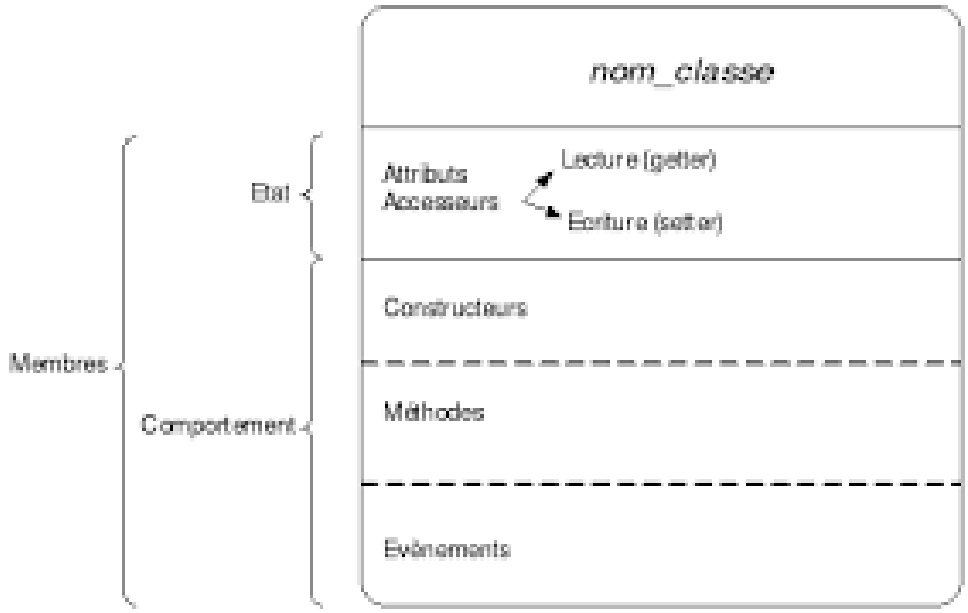

Figure 3: Structure d'une classe dans la POO

b) Le prototype

Le prototype est un objet à part entière qui sert de prototype de définition de la structure interne et des messages. Les autres objets de mêmes types sont créés par clonage. Dans le prototype, il n'y a plus de distinction entre attributs et messages : ce sont tous des slots.

1.4 Les avantages et les inconvénients de la POO

1.4.1 Les avantages de la POO

Le but de la POO est de rendre la programmation complexe plus facile c'est-à-dire, la rendre plus familière à l'homme. En outre, la POO nous aide à concrétiser certaines notions qui seraient difficiles à coder en langage procédural.

Par exemple la modélisation d'une voiture dans les deux paradigmes de programmation : le langage procédural et la POO. Dans un langage procédural, le programmeur va devoir utiliser des outils abstraits alors qu'en POO, le programmeur pourra faire un objet "voiture" et lui donner ses caractéristiques, un objet "roue" par exemple, ainsi de suite, ce qui rend la programmation plus simple et surtout plus instinctive. Ceci est un avantage non négligeable dans le monde de la programmation, qui permet donc un gain de temps, de compréhension mais aussi d'efficacité.

1.4.2 Les inconvénients de la POO

Tous les programmes ne peuvent être modélisés avec précision par le modèle d'objets. Si vous voulez juste lire certaines données, faire quelque chose de simple et l'écrire revenir en arrière, vous n'avez pas besoin de définir des classes et des objets. Toutefois, dans certains langages de POO, vous pouvez avoir à effectuer cette étape supplémentaire.

Un autre inconvénient est que le concept d'un programmeur de ce qui constitue un objet abstrait peut ne pas correspondre à la vision d'un autre programmeur. [3]

1.5 Conclusion

Ce chapitre introductif a été consacré essentiellement à la présentation d'un type essentiel de la programmation en informatique sur lequel se base l'application qu'on va développer dans ce projet de fin d'études.

Chapitre 2

Xamarin et le concept multiplateforme

Chapitre 2
Xamarin et le concept multiplateforme

2.1 Introduction

Un logiciel multiplateforme qui est capable de fonctionner sur plusieurs systèmes d'exploitation et sur différents ordinateurs Smartphones et tablettes.

A l'heure de l'explosion du développement d'applications mobiles, Xamarin[1] [4] permet un concept très simple : développer une fois, déployer sur tous les équipements. La solution de cross-platform Xamarin permet en effet de développer des applications mobiles en C# qui peuvent être déployées par la suite aussi bien sur Windows Phone, Android et iOS.

Dans ce chapitre, nous présentons les concepts qui permettent de développer des applications mobiles et multiplateformes.

2.2 Le concept multiplateforme

2.2.1 Définition

Le concept multiplateforme concrétise la capacité d'un langage de programmation (comme c# dans notre projet) permettant aux programmeurs de développer des logiciels pour plusieurs plateformes concurrentes en écrivant un programme qu'une seule fois. Le logiciel multiplateforme peut fonctionner sur la plupart ou tous les systèmes avec peu ou pas de modification.

En informatique, une plateforme représente le binôme formé par l'ordinateur et son système d'exploitation, alors que multiplateformes est généralement utilisé pour décrire les logiciels informatiques.

[1] Xamarin : est un nouveau standard pour le développement des applications mobiles et multiplateformes.

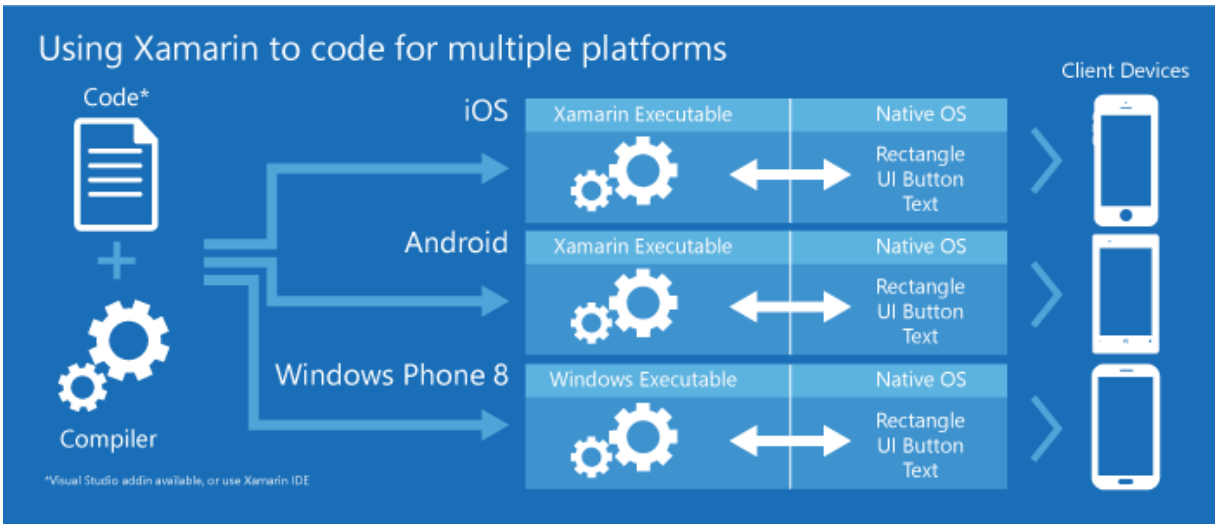

Figure 4: Xamarin utilisant le code pour les multiplateformes [4]

2.2.2 Xamarin

a) Présentation de Xamarin

Xamarin est une société de logiciels basée à San Francisco en Californie, créée en Mai 2011 par les ingénieurs qui ont créé Mono, MonoTouch et Mono pour Android qui sont mises en œuvre multiplateforme de la Common Langage Infrastructure (CLI) et les spécifications de langage commun (souvent appelé Microsoft .NET). Par ailleurs, Avec un C # partagé, les développeurs peuvent utiliser Xamarin pour iOS natives, Android, et Windows applications avec des interfaces utilisateur natifs et le code de l'action sur plusieurs plateformes.

b) Développement mobile et multiplateforme

L'industrie des ordinateurs personnels a connu un déplacement massif au cours des dernières années. Les ordinateurs de bureau existent encore, bien sûr, et ils demeurent vitaux pour les tâches qui nécessitent des claviers et des grands écrans. Par ailleurs, une grande partie de l'informatique personnelle se produit maintenant sur des petits appareils, en particulier pour les applications de consultation de l'information et de consommation des médias. Par exemple, les Smartphones ont un paradigme fondamentalement différent avec une interaction utilisateur basée principalement sur contact avec un clavier qui apparaît seulement en cas de nécessité.

c) Le paysage mobile

Bien que le marché du mobile ait le potentiel de changement rapide, il existe actuellement deux grands téléphones et des plateformes de tablettes:

- La famille Apple iPhones et iPads, dotée du système d'exploitation iOS.
- Le système d'exploitation Android développé par Google basé sur le noyau Linux, qui fonctionne sur une variété de téléphones et de tablettes.

Il y a aussi une troisième plate-forme de développement mobile qui n'est pas aussi populaire qu'iOS et Android, mais qui implique une société avec une forte tradition dans l'industrie de l'ordinateur personnel, il s'agit de Windows Phone de Microsoft.

Windows Phone a récemment obtenu un coup de pouce avec les versions récentes de Windows 8.1 et Windows Phone 8.1. Avec ces versions, une interface de programmation d'application unique appelée Windows Runtime est maintenant disponible pour les applications fonctionnant sur des machines de bureau, ordinateurs portables, tablettes et téléphones.

d) Problèmes de développement mobile et multiplateforme

- **Problème 1 (Différents paradigmes d'interface utilisateur) :** Les trois plateformes comportent des moyens similaires de présentation de l'interface utilisateur graphique (GUI) et les interactions avec le dispositif à travers la multitouche, mais en détail il y a beaucoup de différences. Chaque plateforme a différentes manières de naviguer dans les applications et les pages, différentes conventions pour la présentation des données, différentes façons d'invoquer et de menus d'affichage, et même des approches différentes au toucher.

Par ailleurs, les utilisateurs sont habitués à interagir avec des applications sur une plateforme particulière et s'attendent à tirer parti de cette connaissance avec les applications futures. Chaque plateforme acquiert sa propre "culture" de sortes, et ces conventions culturelles influencent alors les développeurs.

Figure 5: les paradigmes d'interface utilisateur dans IOs

Figure 6: Les paradigmes d'interface utilisateur sous Android

Figure 7: Paradigmes d'interface utilisateur dans Windows

- **Problème 2 (différents environnements de développement)** : Les programmeurs d'aujourd'hui sont habitués à travailler dans un environnement de développement intégré (IDE). Ces IDE existent pour tous les trois plateformes, mais bien sûr, ils sont différents :

  - La figure 8 illustre l'environnement de développement iOS, XCode sur Mac.

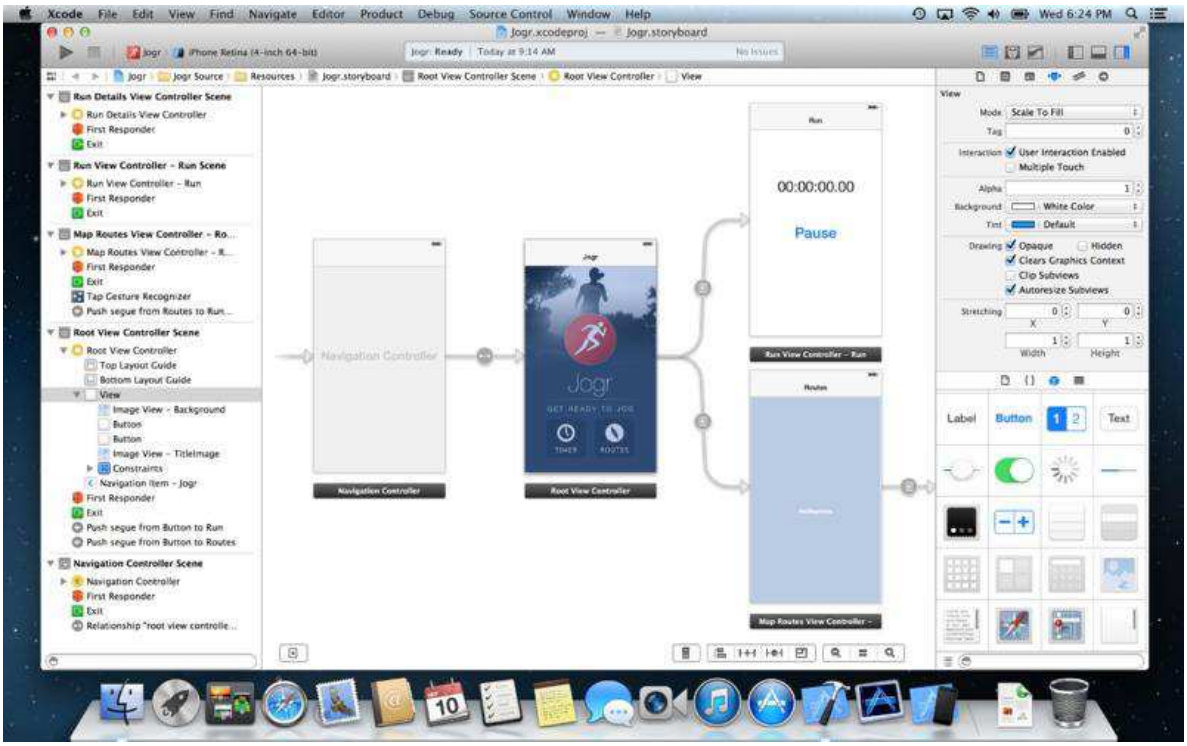

Figure 8: Environnement de développement d'IOs

- La figure 9 illustre l'environnement de développement Android, Eclipse sur une variété de plateformes.

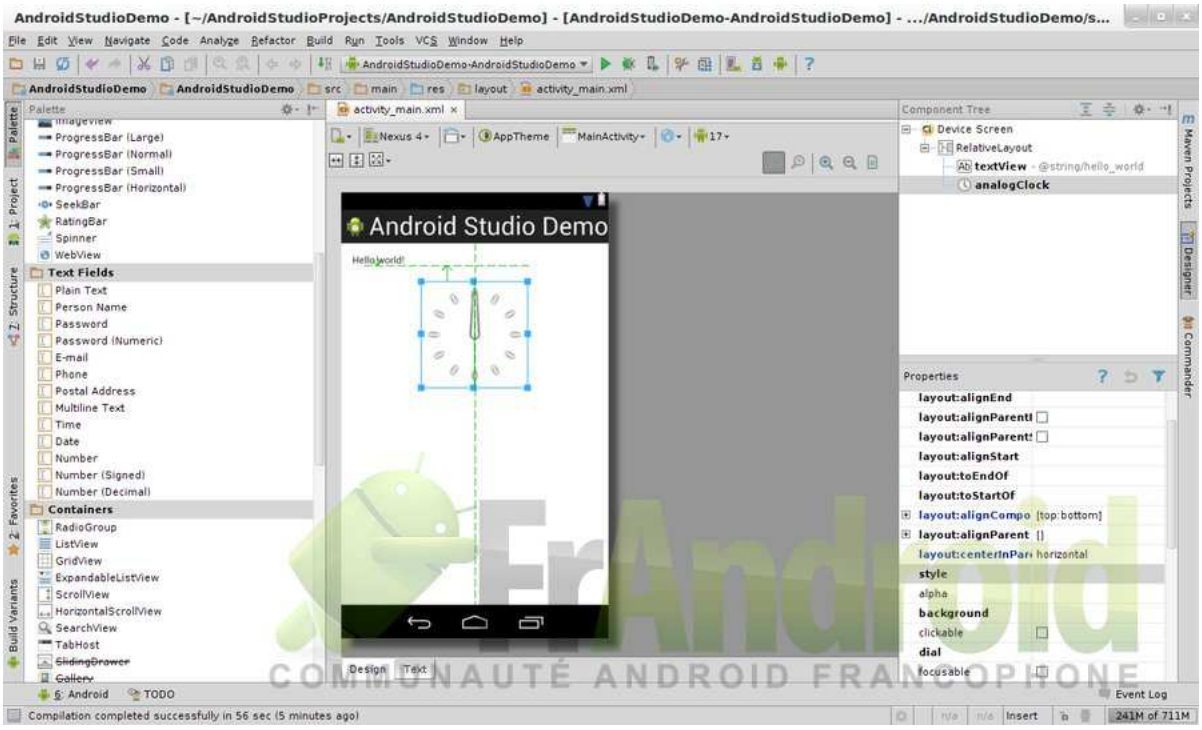

Figure 9: Environnement de développement d'Android

- La figure 10 illustre l'environnement de développement de Windows Phone, Visual Studio sur PC.

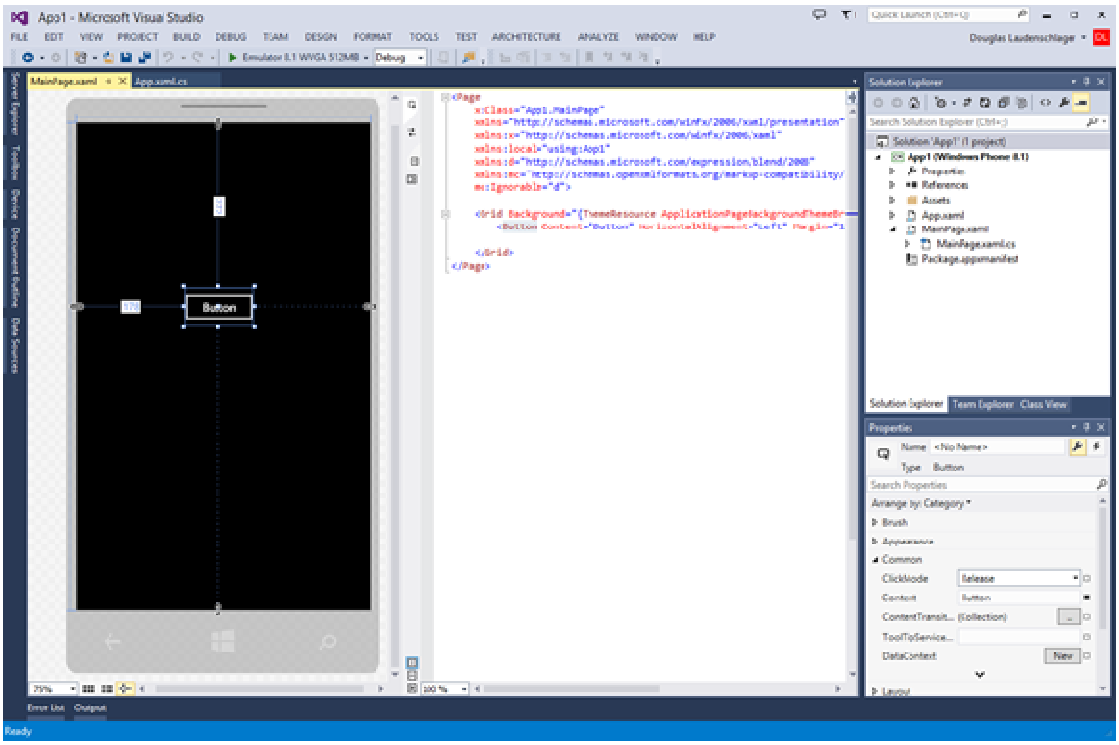

Figure 10: IDE de Windows phone

- **Problème 3 (Différentes interfaces de programmation)** : Tous les trois IDE de ces plateformes sont basés sur différents systèmes d'exploitation avec différentes API. Dans de nombreux cas, les trois plateformes mises en œuvre tous les types d'objets de l'interface utilisateur, mais avec des noms différents. Par exemple, toutes les trois plateformes ont quelque chose qui permet à l'utilisateur de basculer entre deux états:

  - Sur iPhone, c'est une "vue" appelée UISwitch.
  - Sur les périphériques Android, c'est un "widget" appelé Switch.
  - Sur Windows Phone, une possibilité est "un contrôle" appelé ToggleSwitchButton du package Windows Phone ToolkitNuGet.

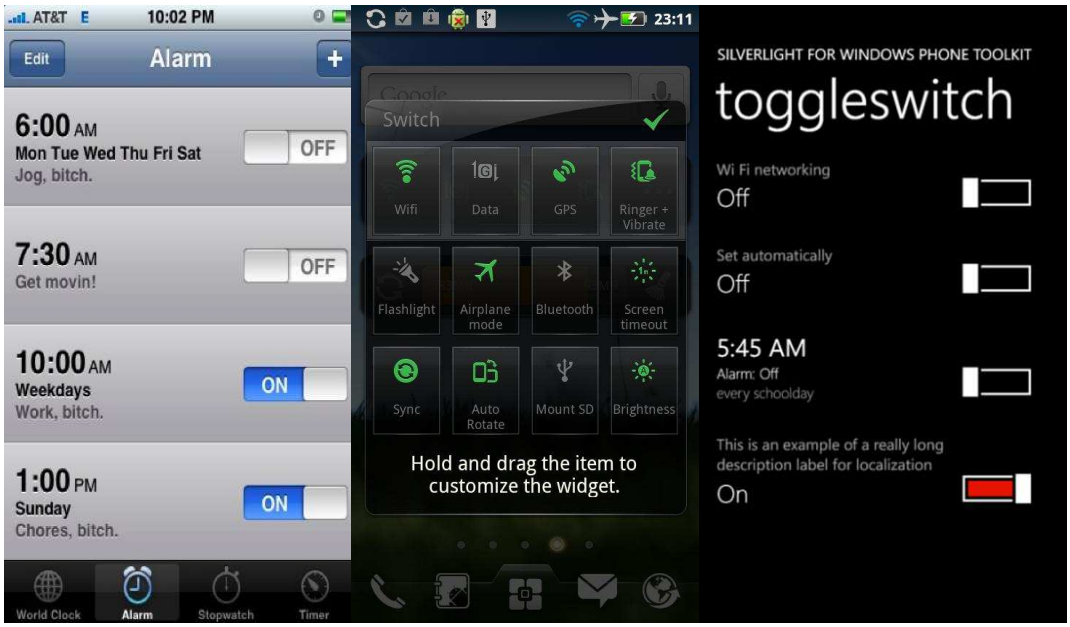

Figure 11: Une illustration de "UISwitch, Switch et ToggleSwitch"

- **Problème 4 (Différents langages de programmation) :** Les développeurs ont une certaine souplesse dans le choix d'un langage de programmation pour chacun de ces trois plateformes, mais en général, chaque plateforme est très étroitement associée à un langage de programmation bien particulier :

  - Objective-C pour l'iPhone.
  - Java pour les appareils Android.
  - C # pour Windows Phone.

Ces trois langages sont étroitement liés puisqu'ils sont tous les descendants orientées objet de C, mais ils sont devenus des cousins éloignés plutôt. Pour ces raisons, une entreprise qui souhaite cibler de multiples plateformes pourrait très bien employer trois différentes équipes de programmeurs, chaque équipe qualifiée et spécialisée dans un langage et API qui sont particuliers.

Par ailleurs, ce problème de langage est particulièrement le plus important, mais c'est le problème qui nécessite d'être résolu. Si nous pouvons utiliser le même langage de programmation pour ces trois plateformes, nous pourrons au moins avoir un code partagé entre les plateformes. Ce code partagé n'aurait probablement pas être impliqué dans l'interface utilisateur parce que chaque plateforme a différentes API, mais il pourrait bien être le code d'application qui ne touche pas l'interface utilisateur du tout.

Un langage unique pour toutes les trois plateformes serait certainement commode.

2.3 C # et .NET

C# dévoilé par Microsoft en 2000, est un assez nouveau langage de programmation, au moins en comparaison avec Objective-C et Java. Dans un premier temps, C # semblait être un langage assez simple fortement typé, impératif et orientée objet, certainement influencé par C ++ (et Java ainsi), mais avec une syntaxe beaucoup plus propre que C ++ et aucun des bagages historique.

En outre, la première version de C# avait le soutien de niveau langage pour les propriétés et les événements, qui s'avèrent à être les types de membres qui sont particulièrement adaptés pour les interfaces utilisateur graphiques de programmation. Mais C# a continué à croître et s'améliorer au fil des ans. Le soutien des génériques, lambda fonctions, LINQ, et les opérations asynchrones a élevé avec succès C# pour être classé comme un "langage multi-paradigme de programmation". Le code C# peut être traditionnellement impératif, ou enrichi avec paradigmes de programmation déclaratifs ou fonctionnels.

Le 3 Avril 2014, Anders Hejlsberg a publié une version open-source du compilateur C # appelée plateforme .NET Compiler (anciennement connu sous son nom de code " Roslyn"). Par ailleurs, depuis sa création, C# a été étroitement associé avec Microsoft .NET Framework. Au plus bas niveau, ".NET" fournit une infrastructure pour les types de données de base C# (int, double, chaîne, etc.). Mais ".NET" fournit également une bibliothèque de classes ".NET" Framework pour de nombreuses tâches courantes rencontrées dans de nombreux types de programmation. Ceux-ci comprennent: Math, Mise au point, Réflexion, Collections, Mondialisation, fichier I/O, Réseautage, Sécurité et Threading.

- Des services Web.
- Traitement de données.
- La lecture et l'écriture de XML.

2.4 Partage de code

L'avantage de cibler de multiples plateformes avec un seul langage de programmation vient de la capacité à partager le code entre les applications.

Avant que le code puisse être partagé, il doit être structuré à cette fin. En particulier depuis la généralisation de l'utilisation d'interfaces utilisateur graphiques, les programmeurs ont compris l'importance de la séparation du code de l'application en couches fonctionnelles. Elle s'était peut-être la division la plus utile entre le code utilisateur et l'interface.

L'architecture MVC (Modèle-Vue-Contrôleur) officialise la séparation en un modèle (les données sous-jacente), la vue (La représentation visuelle des données), et le contrôleur (qui traite les entrées de l'utilisateur). La version originale de MVC date des années 1980. Plus récemment, l'architecture MVVM (Model-View-ViewModel) a efficacement modernisé MVC

en ajoutant des interfaces graphiques modernes. MVVM sépare le code dans le modèle (la donnée sous-jacente), la vue (l'interface utilisateur, notamment visuels et entrée), et le ViewModel qui gère le passage de données entre le modèle et la vue.

Lors de l'élaboration d'un programme qui cible de multiples plateformes mobiles, l'architecture de MVVM aide le développeur à séparer le code de la plateforme spécifique View où le code nécessite d'interagir avec API et ViewModel indépendant de la plateforme. Souvent ce code indépendant de la plateforme doit accéder aux fichiers, ou utilise des collections ou filetage. Normalement, ces emplois seraient considérés comme faisant partie d'une API du système d'exploitation, mais ils sont aussi des emplois qui peuvent faire usage de la bibliothèque de classes ".NET" Framework, et si la bibliothèque de classe est disponible sur chaque plateforme, il sera effectivement indépendant de la plateforme.

Quelle que soit la méthode utilisée, ce code commune a un accès à la bibliothèque de classes .NET Framework, il peut effectuer des I/O, gérer la mondialisation, les services web, décomposer XML, et ainsi de suite. Cela signifie que nous pouvons créer une seule solution Visual Studio qui contient quatre projets C# ciblant les trois principales plateformes mobiles, ou nous pouvons utiliser Xamarin Studio pour iPhone et Android. La figure 12 montre les interrelations entre Visual Studio ou Xamarin Studio, les bibliothèques Xamarin, et les API de la plateforme:

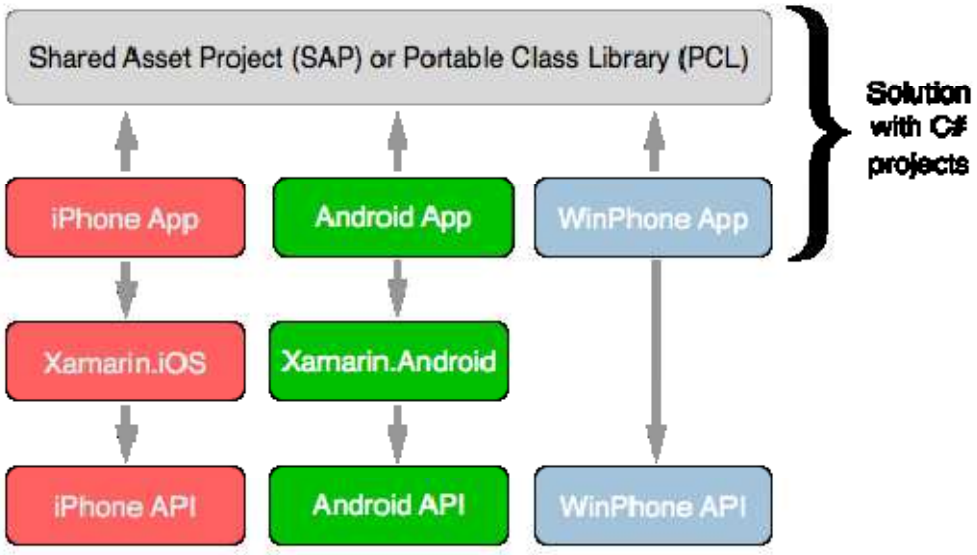

Figure 12: Interrelations entre Visuel Studio et les API

Les boîtes de la deuxième rangée sont les applications spécifiques à la plateforme réelle. Celles-ci font des appels au projet commun et aux bibliothèques de Xamarin pour la mise en œuvre des API de la plateforme native.

Lorsque l'application iPhone est construite, le compilateur génère Xamarin C# et C# Intermediate Language (IL) comme d'habitude, mais elle fait appel à l'utilisation du compilateur Apple sur MAC pour générer le code machine iPhone native tout comme le

compilateur Objective-C. Les appels à partir de l'application pour l'iPhone API sont les mêmes que si la demande était écrite en Objective-C.

Pour les applications sur Android, le compilateur C # Xamarin génère IL, qui fonctionne sur une version de Mono sur l'appareil à côté du moteur Java, mais les appels à partir de l'API des applications sont à peu près les mêmes que si les applications ont été écrites en Java.

Le 28 mai 2014, Xamarin introduit Xamarin.Forms dans le cadre d'une collection d'améliorations à la plateforme Xamarin. Xamarin.Forms nous permet d'écrire du code de l'interface utilisateur qui peut être compilé pour iPhone, Android et Windows Phone. Une application Xamarin.Forms permet de générer trois projets distincts pour les trois plateformes avec un quatrième projet contenant du code qui es en commun. Cependant, dans une application Xamarin.Forms, les trois projets sont généralement très petits, souvent consistant seulement des fragments avec un peu de code passe-partout de démarrage. La figure 13 illustre cette description.

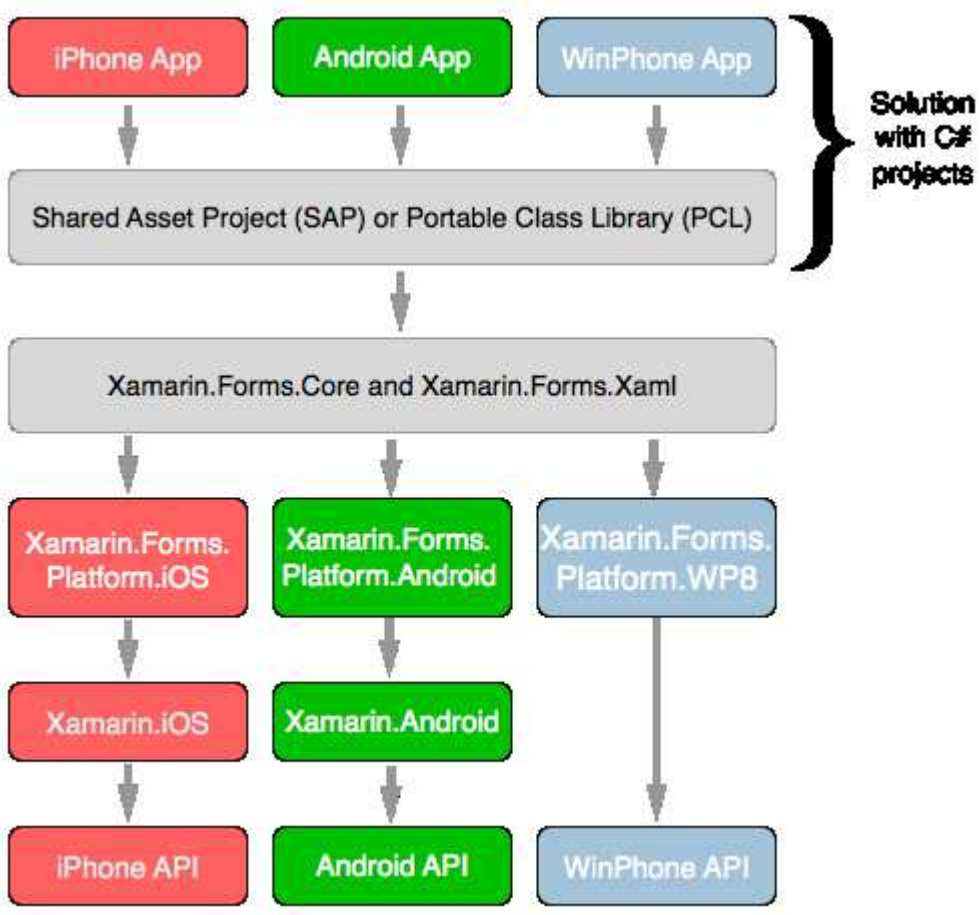

Figure 13: Les packages de Xamarin pour résoudre les problèmes de la multiplateforme

Les bibliothèques Xamarin.Forms.Core et Xamarin.Forms.Xaml implémentent l'API Xamarin.Forms. En fonction de la plateforme, Xamarin.Forms.Core permet alors l'utilisation de

l'un des Xamarin.Forms. Ces bibliothèques sont la plupart du temps une collection de classes appelées équarrisseurs qui transforment l'interface utilisateur Xamarin.Forms en une interface utilisateur spécifique à la plateforme.

Xamarin.Forms.Core contient également une classe nommée curseur pour afficher une barre horizontale que l'utilisateur manipule pour choisir une valeur numérique. Dans les moteurs des bibliothèques spécifiques à la plateforme, cela estampé à un UISlider sur l'iPhone, une barre de recherche sur Android, et un téléphone coulissant sur Windows. Cela signifie que lorsque nous écrivons un programme Xamarin.Forms qui a un commutateur ou un curseur, ce qui est effectivement affiché, est l'objet correspondant mis en œuvre dans chaque plateforme.

Par exemple la photo qui apparait dans la figure 14 illustre un petit programme de Xamarin.Forms contenant une étiquette lecture "Bonjour, Xamarin.Forms!", Un Bouton disant "Cliquez-moi!", un commutateur et un curseur. Le programme est exécuté (de gauche à droite)

L'iPhone, Android et Windows Phone:

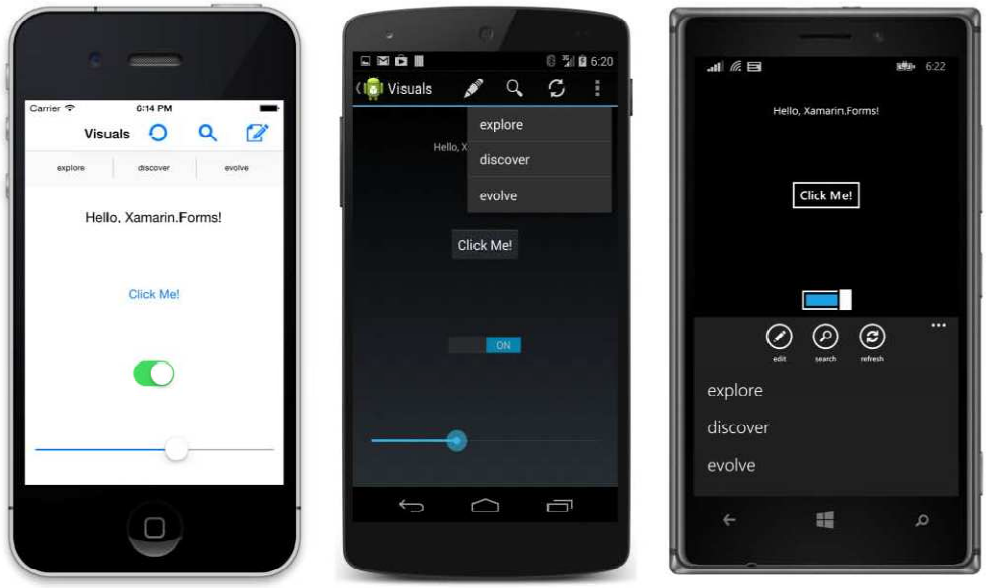

Figure 14: Les trois plateformes IOs, Androïd et Windows phone

2.5 Conclusion

Ce chapitre présente les concepts et les outils nécessaires pour les applications mobiles et multiplateformes. Puis on a mis l'accent sur l'outil Xamarin et ses composants. Pour une meilleure compréhension de ce nouveau paradigme de programmation qui vise à ne pas lier l'application à la plateforme de développement.

Le chapitre suivant est une illustration de ces concepts à travers une application de la téléphonie. Cette application permet de faire une conversion d'une chaîne alphanumérique en une chaîne numérique correspondante à un numéro de téléphone.

Chapitre 3
Développement d’une application multiplateforme (Applr)

Chapitre 3
Développement d'une application multiplateforme (Applr)

3.1 Introduction

Avec la présence de plusieurs plateformes et de plusieurs types d'équipements (ordinateurs, Smartphones et tablettes), le monde d'exploitation des logiciels a besoin de ne pas être esclave seulement à une plateforme bien particulière et en absence de cette plateforme le logiciel devient inutile. Dans cette optique, les développeurs des logiciels ont pensé au développement de logiciels qui ne doivent pas dépendre d'une plateforme bien spécifique mais qui peuvent être exploitables sur différentes plateformes et différents équipements. Ces applications sont dites applications multiplateformes.

Dans ce chapitre nous développons un prototype d'une application qui est multiplateforme. Cette application consiste à traduire une chaîne alphanumérique en une chaîne numérique qui représente un numéro de téléphone.

3.2 Application Applr

Le nom de notre application est divisé en deux parties. La première partie c'est *Appl* qui exprime le mot application et la lettre *r* qui exprime le mot "rappeler" qui est dans le même sens du mot "envoyer".

3.2.1 Principe de fonctionnement

L'application consiste à faire un décodage des numéros téléphoniques à partir des chaînes alphanumériques, puis effectuer un appel téléphonique au numéro décodé indépendamment de la plateforme utilisée.

Le code sharing ou bien le code partagé avait pris lieu ces dernières années, et la technologie Xamarin n'est pas une exception. Dans ces technologies, le développeur a le choix à différents types ou bien stratégies à suivre pour partager son code cross-platform. Dans le cadre de notre projet nous avons développé l'application "Applr". Pour cela, on a utilisé les PCL qui nous donne la possibilité de coder les parties communes de l'application (souvent la logique de l'App) et puis générer au même temps DLL pour chaque ".Net Platform".

Les PCL offre une grande flexibilité et accélère le développement de l'App en centralisant une grande partie du code qui doit être codée et testée, et même modifiée une seule fois pour chaque sous projet.

3.2.2 Outils de développement

Dans cette section, nous présentons les outils logiciels nécessaires pour le développement de l'application

a) Choix du langage

L'application développée dans le cadre de ce projet, est une application cross-plateformes (multiplateformes). Pour cela nous avons choisi un langage qui a ce profile représenté par C# sous Microsoft Visual Studio.

b) Présentation de C#

Le langage de programmation C# (C dièse en français, ou prononcé C-Sharp en anglais) a été développé par la société Microsoft, et notamment un de ses employés, Anders Hejlsberg, pour la plateforme .NET (point NET / dot NET).

Ce langage est un langage orienté objet, avec un typage fort. Il est très proche du langage Java [5].

C# est un langage orienté objet de type sécurisé et élégant qui permet aux développeurs de générer diverses applications sécurisées et fiables qui s'exécutent sur le .NET Framework. Nous pouvons utiliser le langage C# pour créer entre autres des applications clientes Windows, des services Web XML, des composants distribués, des applications client-serveur et des applications de base de données.

c) Visual Studio 2013

Visual Studio 2013 est un ensemble d'outils complet qui permet de générer des applications bureautiques et le développement en équipe d'applications Web d'entreprise. Nous pouvons donc non seulement générer des applications bureautiques de haute performance, mais aussi tirer parti des outils puissants de développement à base de composants et autres technologies telles que Visual Studio. Cette technologie pourra mettre à notre disposition des outils pour simplifier la conception, le développement et le déploiement de solutions d'entreprise en équipe.

d) L'interface de Visual Studio 2013

Comme chaque langage le Visual studio est constitué de plusieurs éléments :

File (Fichier), Edition, Built (construire), Project (projet) qui définit par nom_project.cs cs qui signifié C Sharp, Debug ou débuguer, team ou composant, SQL c'est la partie de base de

données, Tools sont les outils…etc. La figure 15 montre l'interface du Microsoft Visual studio 2013.

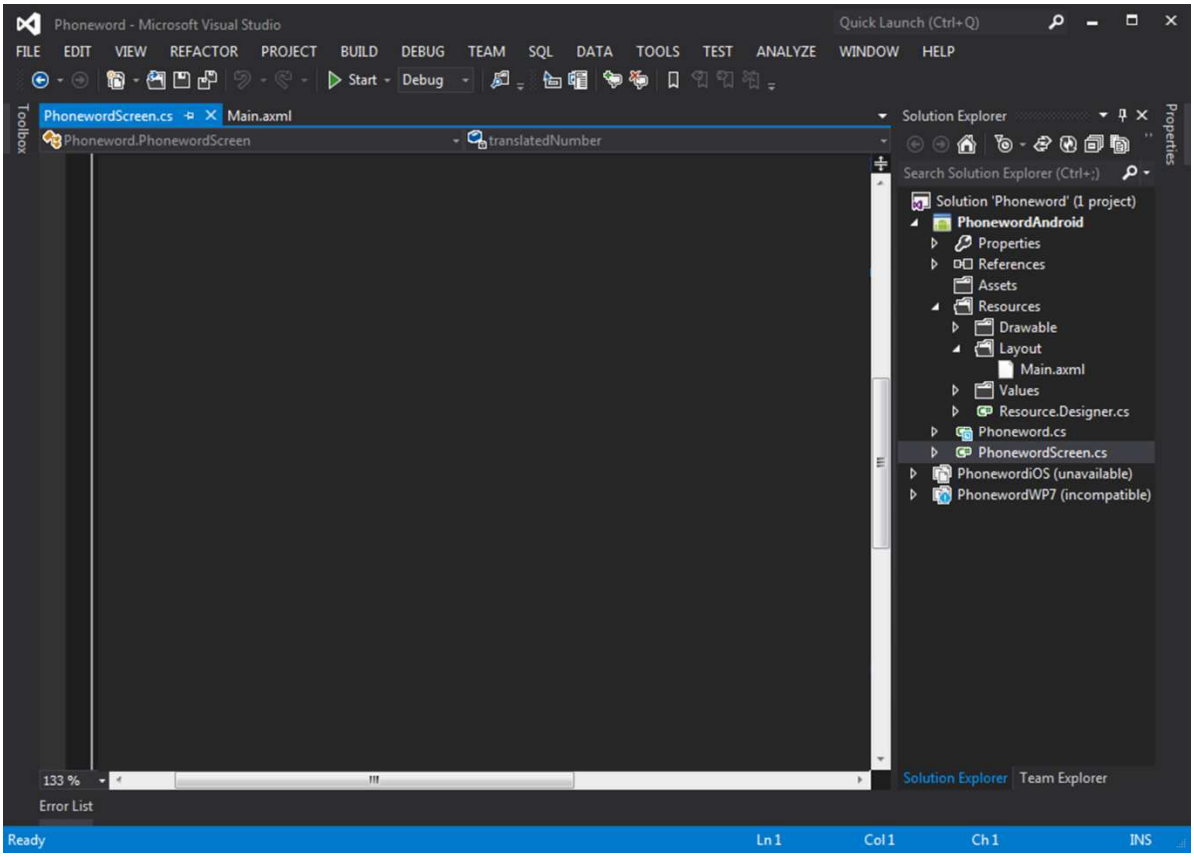

Figure 15: L'interface du Microsoft Visual Studio 2013

3.2.3 Description du code de l'application

Dans cette section, nous présentons comment se fait la conversion d'une chaîne alphanumérique en un numéro de téléphone.

```
---------------------------------------------------------------------------------------------------------
using System.Text;
using System;
//sont des bibliothèques

namespace Applr
{
  public static class PhonewordTranslator// la class PhonewordTranslator
  {
     public static string ToNumber(string raw)
     {
         if (string.IsNullOrWhiteSpace (raw))
            return "";
         else
            raw = raw.ToUpperInvariant();

         var newNumber = new StringBuilder();
        foreach (var c in raw)
```

```
        {
            if (" -0123456789".Contains(c))
                newNumber.Append(c);
            else {
                var result = TranslateToNumber(c);
                if (result != null)
                    newNumber.Append(result);
            }
        }

        return newNumber.ToString();
    }
    static bool Contains (this string keyString, char c)
    {
        return keyString.IndexOf(c) >= 0;
    }
static int? TranslateToNumber(char c)
    {
        if ("ABC".Contains(c))
            return 2;
        else if ("DEF".Contains(c))
            return 3;
        else if ("GHI".Contains(c))
            return 4;
        else if ("JKL".Contains(c))
            return 5;
        else if ("MNO".Contains(c))
            return 6;
        else if ("PQRS".Contains(c))
            return 7;
        else if ("TUV".Contains(c))
            return 8;
        else if ("WXYZ".Contains(c))
            return 9;
        return null;

    }
  }
}
```

Le code source ci-dessus représente la partie majeure qui fait le processus de décodage, on avait nommé la fonction *ToNumber* qui doit rendre le numéro après traitement du code passé en paramètre dans la méthode:

1- Vérification logique des entrées de l'utilisateur, gestion des exceptions et des cas indéfinis.

2- Séparation de la partie alphabétique.

3- Identification des nombres obtenus par une fonction fils *TranslateToNumber*.

4- Retour des résultats d'identification et concaténation du numéro final.

Les classes font parties de l'espace du nommage Applr qui les rends accessibles aux autres assemblées en partie UI.

3.2.4 UI user interface

La partie interface graphique diffère d'une plateforme à une autre, mais le codage se fait en utilisant de langage XAML de Markup très proche en syntaxe au HTML.

Cette partie peut être faite sans aucun savoir préalable du XAML puisque les IDE utilisés que ce soit Xamarin studio ou VS15 offre des designers drag & drop qui facilitent la tâche au programmeur et lui booste l'étape du design.

3.2.5 Partie interaction avec la UI

a) Exemple Windows phone

- La première partie

```
using Applr;
namespace Applr_WindowsPhone
{
  [Activity(Label = "Phoneword", MainLauncher = true)]
  public class PhonewordScreen : Activity
  {
string translatedNumber;
      protected override void OnCreate(Bundle bundle)
      {
        base.OnCreate(bundle);
                SetContentView(Resource.Layout.Main);
                 Button TranslateButton = FindViewById<Button>(Resource.Id.TranslateButton);
      Button CallButton = FindViewById<Button>(Resource.Id.CallButton);
        EditText PhoneNumberText = FindViewById<EditText>(Resource.Id.PhoneNumberText);

         TranslateButton.Click += delegate
        {

            // *** SHARED CODE
            translatedNumber = Applr.PhonewordTranslator.ToNumber(PhoneNumberText.Text);

            if (translatedNumber == "")
{
                CallButton.Text = "Call";
                CallButton.Enabled = false;
            } else {
                CallButton.Text = "Call " + translatedNumber;
                CallButton.Enabled = true;
            }
        };
```

Nous simplifions cette partie du code par avec l'image significative présentée dans la figure 16. Cette interface représente le choix de l'appel du mot alphanumérique qui est transformé au numérique avec une seule action en appuyant sur le bouton" traduire".

Figure 16: Interface de l'entrée du mot de téléphone sur Windows phone

- **La deuxième partie**

```
CallButton.Click += delegate {
            var callIntent = new Intent(Intent.ActionCall);
            new AlertDialog.Builder(this)
              .SetMessage("Call " + PhoneNumberText.Text + "?")
              .SetNeutralButton("Call", delegate {
                 callIntent.SetData(Android.Net.Uri.Parse("tel:" + translatedNumber));
                 StartActivity(callIntent);
              })
              .SetNegativeButton("Cancel", delegate{})
              .Show() ;
         };
     }
  }
}
```

Cette partie présente la dernière étape qui consiste de faire cet appel par le numéro que nous avons obtenu à travers la traduction du mot alphanumérique. La figure 17 illustre cette étape (lancement d'un appel).

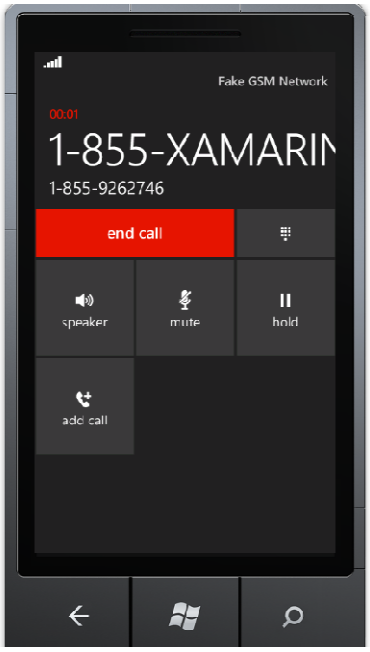

Figure 17: Exemple d'appel téléphonique sur Windows phone

3.3 Conclusion

Dans ce chapitre nous avons présenté en détails la démarche suivie pour le développement de notre application et les outils nécessaires pour sa mise en place. Puis nous avons présenté quelques exemples d'exécutions pour illustrer comment elle fonctionne. Nous avons pris comme exemple Windows phone pour son exploitation mais on pourra l'exploiter sur d'autres plateformes.

Conclusion générale

Conclusion Générale

Durant le présent projet de fin d’études, nous avons basé sur un meilleur type de programmation c’est la poo avec un langage que nous le trouvé très facile c’est le C# et nous avons tracé un but pour développer nos capacités de création d’une application à l’aide des Solutions qui nous sont fournis par Xamarin et qui spécifie surtout pour la multi plate-forme et pour faciliter a l’utilisateur d’avoir cette application dans les différents systèmes exploitations comme Android iOS et Windows phone et en fin nous espérons que nous atteignons nos objectifs de nous permettre aux utilisateurs de profiter aux des applications, peu importe comment les différents leurs systèmes et Contribuer à la réduction de prix d'applications et de faciliter la disponibilité pour l'utilisateur,Dernier but qui est d'élever l'autonomisation éducative de nous-mêmes et de notre niveau de développement dans le domaine de la programmationspécialisée surtout pour la multi-plates-formes .

Références bibliographiques

Références bibliographiques

[1] www.microsoftvirtualacademy.com/.../developper-une-application-cross-, Développer une application cross-plateformes avec Xamarin , mise à jour Septembre 2014.

[2] https://www.diigo.com/list/magalib/sci6373 14 mars 2012

[3] https://www.depannetonpc.net/.../lire_56_programmation-**orientee-objet**.html

[4] https://fr.wingwit.com/**programmation**/computer-programming.../87470.html

[5] http://www.microsoftvirtualacademy.com/training-courses/developper-une-application-cross-plateformes-avec-xamarin, Date de mise à jour Septembre 2014.

[6] https://msdn.microsoft.com/en-us/wordament2-msdn.aspx

[7] c#, https://fr.**wiki**books.org/**wiki**/**Programmation**_C_sharp/Version_imprimable

ملخص

تعرف تكنولوجيا المعلوماتية تطورا هاما خصوصا في البرمجة التي تعرف بOOP حيث يشهد العالم تنافسا غير مسبوق في هذا النوع من البرمجة و استعمالها في منصات متعددة او ما يسمى ب (cross-Platform). هذا لإنشاء تطبيقات مخصصة لمختلف الهواتف المحمولة الذكية و أجهزة الكمبيوتر المتنوعة و كذلك أجهزة أخرى ذكية و ذلك يتم باستعمال لغات البرمجة الخاصة بهذا المجال والملائمة لهذه المنصات عموما. من أهم الشركات المتخصصة في هذا المجال شركة اكزامارين (Xamarin) و تعتبر الرائدة فيه خصوصا لإتاحتها و طرحها حلول في هذا المصب و بتنوع التطبيقات في شتى الأجهزة و لاحتياج العالم إلى بعض التطبيقات التي تتماشى مع العديد من المنصات خصوصا المتعلقة بالهواتف الذكية و لهذا قررنا إنشاء تطبيق من هذا السياق التي تعتمد على نوع البرمجة OOP و كلغة البرمجة اخترنا # C و بمعدات مختلفة من فيزيال استيديو و اكزامارين. فورم تتمكن تطبيقتنا من العمل في منصات مختلفة وتقوم بتحويل كلمة الاتصال الهاتفي من الأبجدي الرقمي إلى الرقمي و هذا ما سنراه في مشروعنا.

Résumé

La technologie de l'information connaît un développement important surtout dans la programmation comme la POO, où le monde connaît une concurrence sans précédent dans ce type de programmation qui utilise différentes plateformes (cross platform). Ceci pour créer des applications personnalisées pour différents Smartphones et ordinateurs, ainsi que d'autres dispositifs intelligents. Dans ce type d'applications, on utilise différents langages de programmation appropriés supportés par plusieurs plateformes. Compagnie (Xamarin) est l'une des entreprises les plus spécialisées dans ce domaine. Elle propose des solutions en aval pour les applications dans diverses configurations. Dans la réalité, nous avons besoin de certaines applications qui sont en ligne et qui supportent plusieurs plateformes sur les téléphones intelligents. Pour cela nous avons proposé de développer une application qui se base sur la programmation POO et C # comme langage de programmation avec Visual studio et Xamarin Forms. Ces éléments permettent à notre application de fonctionner sur plusieurs plateformes et convertir le mot alphanumérique au numérique.

Mots clés : Xamarin, application multiplateforme, C#, POO.

Abstract

The information technology is experiencing a significant development especially in programming as OOP, when the world is highly competitive unprecedented in this type of programming and use in multiple platforms (cross platform) and this to create custom applications for various smart phones, various computers and many other intelligent devices. Therefore, in this kind of applications we use appropriate programming languages which support various platforms. The Company (Xamarin) is one of the most specialized companies in this field. It proposes downstream solutions for several applications in various configurations. In reality, we need some applications that are online with many platforms especially on smart phones. For that, we propose to develop an application based on OOP programming and use C # as programming language with Visual Studio and Xamarin Forms. These elements enable for our application to use various platforms and convert alphanumeric message to digital form.

Keywords: Xamarin, cross-platform application, C#, POO.