Dezvoltarea aplicatiilor software pe platforma iOS

Student: Ene Andrei-Bogdan

Grupa 441A

Profesor coordonator: Stefan Stancescu

CUPRINS

Introducere

iOS desemnează sistemul de operare de la compania americană Apple Inc. pentru următoarele calculatoare și aparate "inteligente":

- **Phone - un *smartphone* de mare succes pe piața de telefoane mobile**
- **iPod touch - un player MP3**
- **iPad - un calculator tabletă**
- **Apple TV - un apara de tip *Settop box***

iOS este un sistem de operare de tip Unix, care încă în prima sa versiune a conținut multe elemente din Mac OS X, tot un sistem de operare de tip Unix de la Apple.

Versiunea actuală (18 septembrie 2013) este iOS 7. Ca noutate oferă de exemplu funcționalitatea unui punct de acces la Internet (*hotspot*) personal.

Funcționalitatea iOS poate fi întregită de către utilizator prin procurarea de aplicații suplimentare specializate numite *app*s în prăvălia online App Store a lui Apple. În mai 2011 stăteau la dispoziție acolo cca 350.000 de apps, din care unele sunt chiar gratuite. Exemple de *apps* gratuite: cumpănă "cu apă"; mici animale casnice mișcătoare care se lasă alintate etc.; pahar cu bere virtual. La cealaltă extremă stă aplicația *I'm rich* („Sunt bogat”) care nu poate decât să afișeze pe ecran un diamant rotitor, dar care în schimb costă circa 800 euro. Desigur însă că majoritatea aplicațiilor oferă o utilitate reală.

Din motive de politică a produsului, iOS nu sprijină aplicația multimedială Flash a companiei americane Adobe.

Prin funcționalitatea sa iOS este unul din factorii de succes primordiali al telefoanelor iPhone pe piața mondială.

Un concurent al lui iOS este sistemul de operare Android de la compania Google.

iOS SDK

Pentru scrierea unei aplicatii iOS , este necesar Xcode si iOS SDK. Acest SDK se poate descarca de pe platform Apple numai daca utilizatorul este inregistrat. Vom discuta despre Xcode 4.2 si iOS SDK pentru iOS 5,dar si celelalte versiuni sunt compatibile cu scrierea acestei aplicatii iOS. In cazul in care avem interfata diferita intr-o aplicatie iOS, atunci trebuie sa stim ca avem o alta versiune de Xcode.

Prima aplicatie iOS

Prima aplicatie iOS ne arata cum sa punem o fereastra neagra pe display-ul produsului Apple, si care ii va permite utilizatorului sa acceseze un buton in urma caruia va afisa un text.

Aplicatia este create ca atunci cand accesezi butonul din imagine , digitul 1 va aparea in casuta. Xcode este folosit pentru a intra in programe si pentru a le testa.

Apple a distribuit un simulator iPhone ca o parte a iOS SDK. Simulatorul este o replica a multor aplicatii iOS dar si a imaginii initiale ale produsului. Simulatorul face treaba

mai usoara atunci cand avem erori sau cand trebuie facut un debug la o aplicatie. Acest lucru salveaza destul timp.

Crearea unei noi aplicatii iOS

Dupa ce am instalat iOS SDK , trebuie pornit aplicatia Xcode.Acolo vom intalni diverse template-uri care le consideram un punct de start pentru diferite aplicatii.

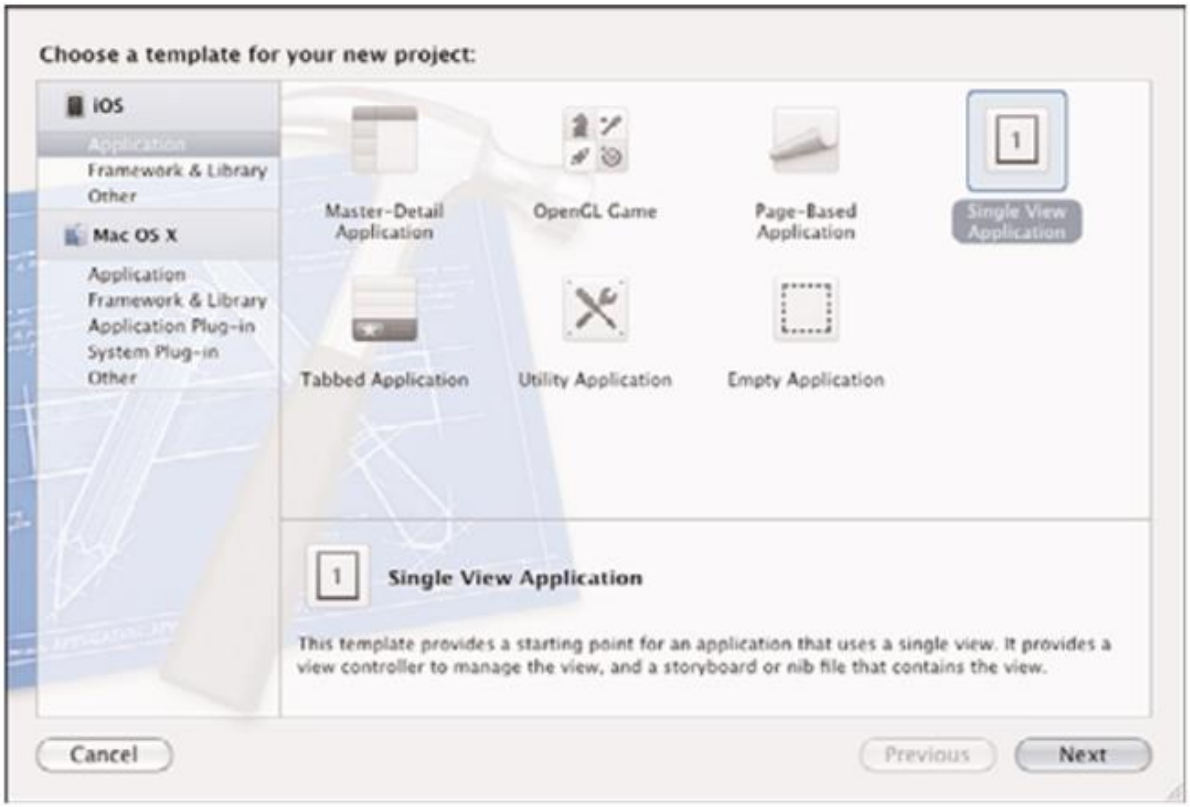


Figure 21.3 Starting a new iOS project

Template-urile Aplicatiei iOS

Tipul aplicatiei	**Descriere**
Master-Detail	Pentru aplicatiile care folosesc un control pentru navigatie.Genereaza o

	impartire a ecranului pentru produsele cu ecran mai mare (iPAD)
OpenGL Game	Este folosit pentru aplicatii grafice si dinamice cum ar fi jocurile
Page-Based Application	Pentru aplicatiile care folosesc un controller pentru vizualizarea paginilor si numerotarea acestora.
Single-View Application	Pentru aplicatiile care incep cu o singura imagine. Se poate desena pe aceasta imagine dupa care se poate vizualiza intr-o alta fereastra
Tabbed Application	Pentru aplicatii care folosesc bara de instrumente.(cum ar fi aplicatiile pentru muzica)
Utillity Application	
Emty Application	Pentru aplicatiile care pornesc doar cu prima fereastra a unui produs Apple. Se poate folosi pentru a configura pornirea.

Crearea unei ferestre

Aplicatiile iOS sunt construite folosind patern-uri de tip MVC. Implementarea aplicatiei (ex:) "AppDelegate" adaugata de un template creaza fereastra aplicatiei in care este loc pentru o singura aplicatie iOS.

Vom folosi codul:

```
public override bool FinishedLaunching(UIApplication app, NSDictionary
options)
{
    // create a new window instance based on the screen size
    window = new UIWindow(UIScreen.MainScreen.Bounds);

    // make the window visible
    window.MakeKeyAndVisible();
```

```
    return true;

}
```

Codul va produce un ecran blank la fel ca in imagine:

Adaugarea unui Controller

La acest rezultat vom adauga un controller prin crearea unui "UIVIEWCONTROLLER" si va fi setat pe "window.RootViewController" :

```
public override bool FinishedLaunching(UIApplication app, NSDictionary
options)
{
    // create a new window instance based on the screen size
    window = new UIWindow(UIScreen.MainScreen.Bounds);

    controller = new UIViewController();
    controller.View.BackgroundColor = UIColor.White;
    window.RootViewController = controller;

    // make the window visible
    window.MakeKeyAndVisible();

    return true;
}
```

Fiecare controller are o imagine asociata care va putea fi accesata din butonul “view”. Codul va schimba imaginea “BackgroundColor” in “UIColor.White” , si va fi afisata:

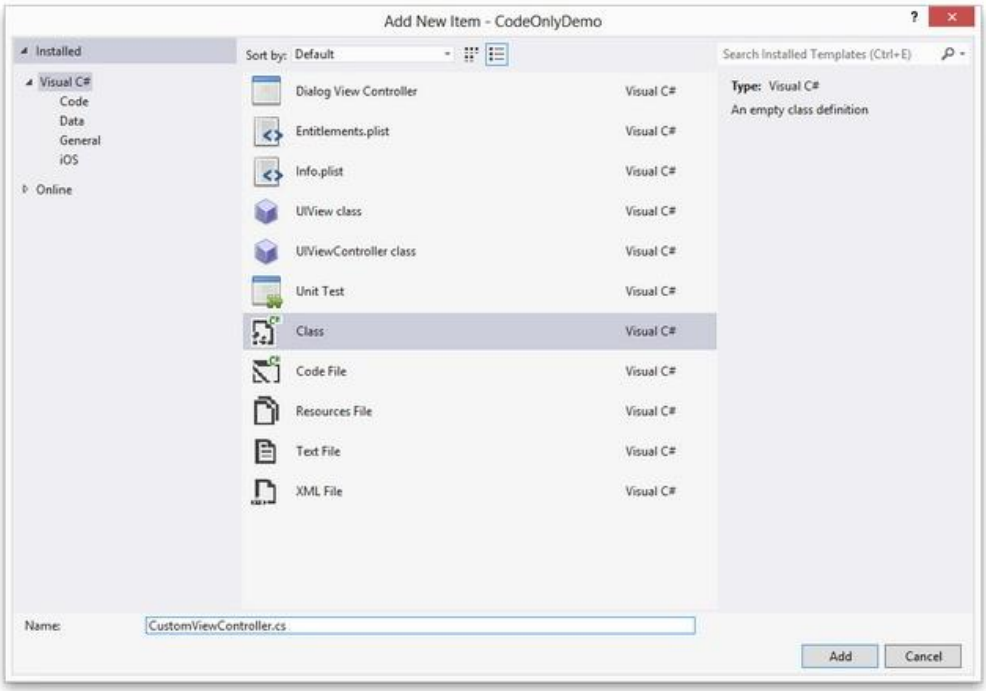
Putem seta orice subclasa “UIViewController” ca sa functioneze ca un “RootViewController”. De exemplu urmatorul cod adauga “UINavigationController” ca un “RootViewController”:

```
public override bool FinishedLaunching(UIApplication app, NSDictionary
options)
{
    // create a new window instance based on the screen size
    window = new UIWindow(UIScreen.MainScreen.Bounds);

    controller = new UIViewController();
    controller.View.BackgroundColor = UIColor.White;
    controller.Title = "My Controller";

    navController = new UINavigationController(controller);
    window.RootViewController = navController;

    // make the window visible
    window.MakeKeyAndVisible();

    return true;
     }
```

Acest lucru produce urmatorul effect:

Acum ca am vazut cum se poate adauga un Controller ca un "RootViewController" al ferestrei, haideti sa vedem cum putem crea o vizualizare oarecare a controllerului in cod:

Adaugam o noua clasa "CustomViewController"



Clasa ar trebui sa mosteneasca din UIViewController care se afla in MonoTouch.UIKit ca in codul alaturat:

```
using System;
using MonoTouch.UIKit;

namespace CodeOnlyDemo
{
    class CustomViewController : UIViewController
    {
    }
      }
```

Initializarea interfatei grafice

UIViewController are o metoda numaita ViewDidLoad ,ceea ce vom folosi in codul urmator pentru a schimba culoarea fundalului in gri:

```
using System;
using MonoTouch.UIKit;

namespace CodeOnlyDemo
{
    class CustomViewController : UIViewController
    {
        public override void ViewDidLoad()
        {
            base.ViewDidLoad();

            View.BackgroundColor = UIColor.Gray;
        }
    }
}
```

Pentru a incarca acest controller il vom instantia si seta la fereastra RootViewController in AppDelegate:

```
controller = new CustomViewController();
window.RootViewController = controller;
```

Acum cand aplicatia se incarca, CustomViewController va fi incarcat iar culoarea gri va fi afisata

Adaugarea campurilor de text

```
class CustomViewController : UIViewController
{
    UITextField usernameField;

    public override void ViewDidLoad()
    {
        base.ViewDidLoad();

        View.BackgroundColor = UIColor.Gray;

        float h = 31.0f;
        float w = View.Bounds.Width;

        usernameField = new UITextField
        {
            Placeholder = "Enter your username",
            BorderStyle = UITextBorderStyle.RoundedRect,
            Frame = new RectangleF(10, 10, w - 20, h)
        };
```

```
        View.AddSubview(usernameField);
    }
}
```

Cand vom crea UITextField vom seta frame-ul astfel incat sa ii defineasca locatia si marimea. Vom folosi astfel coordonate pentru a face acest lucru. Dup ace setam frame-ul impreuna cu inca alte cateva proprietati, vom apela View.AddSubview pentru a adauga UITextField. Acest lucru va face campul pentru username o sub-imagine a UIView ca in imaginea de jos:

Putem face acelasi lucru in cazul in care dorim un camp pentru parola,dar vom folosi codul in felul urmator:

```
passwordField = new UITextField
{
    Placeholder = "Enter your pasword",
    BorderStyle = UITextBorderStyle.RoundedRect,
    Frame = new RectangleF(10, 45, w - 20, h),
    SecureTextEntry = true
};

View.AddSubview(passwordField);
```

Vom seta SecureTextEntry=true care va ascunde caracterele testate in UITextField:

Adaugarea Butonului

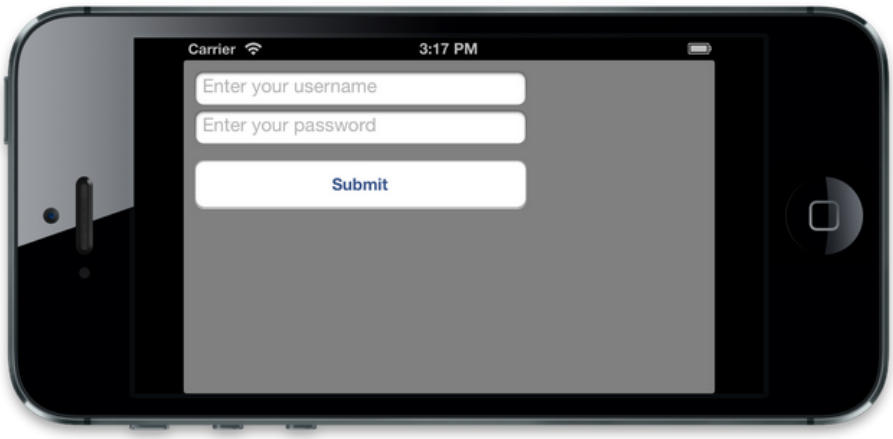
Urmatorul pas va fi adaugarea unui buton pentru ca utilizatorul sa treaca de verificarea parolei si a username-ului:

```
submitButton = UIButton.FromType(UIButtonType.RoundedRect);
submitButton.Frame = new RectangleF(10, 90, w - 20, 44);
submitButton.SetTitle("Submit", UIControlState.Normal);
submitButton.TouchUpInside += delegate
{
    Console.WriteLine("Submit button clicked");
};

View.AddSubview(submitButton);
```

Cum va fi de asteptat imaginea va arata in felul urmator:

Rotatia Afisajului

Cu toate acestea,observam ca atunci cand utilizatorul intoarce produsul in modul landscape, campurile username si password nu se vor redimensiona.



Pentru a repara acest lucru , setam AutoresizingMask:

```
AutoresizingMask = UIViewAutoresizing.FlexibleWidth
```

Acum la rotatia produsului observam ca tot ce am setat pana acum se potriveste perfect si in modul landscape

Stabilitatea Aplicatiilor in comparatie cu Android

Inainte de a considera acest aspect, tin sa mentionez ca aplicatiile sunt scrise de producatori independenti, deci stabilitatea acestora nu va putea fi considerata criteriu real pentru stabilirea sistemului de operare in sine.

Desi anumite rapoarte spun ca pe iOS aplicatiile sunt mai putin stabile decat pe Android, am observat ca majoritatea celor care scriu aceste rapoarte folosesc de fapt criterii destul de anapoda. in general, sunt de parere ca, daca o aplicatie se blocheaza in repetate randuri, cel mai bine este, probabil, sa se renunte la ea. O ricum pe iOS aplicatiile native nu ridica probleme de blocare (lucru care imi spune de fapt ca intr-adevar, este vorba de o problema a aplicatiei, atunci cand aceasta se blocheaza), iar pentru Android exista destule variante similare, astfel ca renuntarea la una in favoarea alteia se va face fara mari regrete.

Exista totusi cateva considerente pe care as vrea sa le aduc in discutie. Primul este faptul ca Apple a ales o metoda de management al blocarii aplicatiilor prin care practic, isi tine clientii in intuneric. De fiecare data cand o aplicatie se blocheaza, Android isi anunta clientii prin mesaje de tipul „Aplicatia nu mai raspunde. Fortezi inchiderea ei?”, pe cand Apple trimite doar un raport al blocajului si directioneaza utilizatorul catre desktop. Acest lucru este perceput de utilizatorii celor

doua sisteme de operare aproximativ in felul urmator: iOS ofera utilizatorilor un sentiment de siguranta in sens ca, psihologic vorbind, impactul pe care blocarea unei aplicatii il are asupra acestora este minim. Atitudinea abordata de Android este etichetata ca fiind matura si corecta, pe cand iOS a fost acuzat ca musamalizeaza aspecte sale negative pe considerente de marketing. De asemenea, s-a speculat faptul ca iOS fiind legat intrinsec de iPhone, iPad si iPod Touch, Apple depinde de o imagine curata a sistemului sau de operare pentru a-si putea comercializa echipamentele, pe cand Android e facut de Google nu doar pentru echipamentele proprii, ci pentru orice producator doreste sa-l adopte, motiv pentru care Android nu are nevoie de asa ceva.

Stabilitatea sistemului de operare in comparatie cu Android

Criteriile de stabilitate ale celor doua sisteme de operare sunt foarte sensibile. De multe ori acestea sunt influentate de altele, cum ar fi stabilitatea aplicatiilor sau configuratia hardware. O testare efectiva ar trebui facuta pe echipamente similare, fara aplicatii independente, dar in acest fel vom avea doua *smartphone*-uri cvasi-inutile, si neexpuse niciunui risc.

Totusi, ar trebui avut in vedere faptul ca, in primul rand, iOS este un sistem de operare mai vechi decat Android si nu esentialmente schimbat de la inceput. Mai mult au fost facute adaugiri decat modificari. Astfel ca, pe structura lui (ma refer la *kernel*), orice problema care sa creeze instabilitate a fost, cu siguranta, indepartata. Android este mai nou, iar variantele noi aduc in primul rand, actualizari de *kernel*. in al doilea rand, iOS produce variante beta de testare care ruleaza intre 5 si 7 saptamani inainte de lansarea oficiala, pe cand Android nu face asa ceva (si nici nu ar putea, nu la scara la care actioneaza in acest sens Apple, pentru ca nu avem de-a face cu acelasi sistem centralizat si controlat ca si in cazul iOS). si in al treilea rand, este vorba de numarul deosebit de mare de producatori de echipamente pentru telecomunicatii mobile care creeaza echipamente ce ruleaza cu Android, iar toti acestia, separat nu au cum sa aiba puterea de a testa si – extrem de important – de a modifica sistemul de operare, pe care o are Apple. in consecinta, cel mai probabil iOS este mai stabil decat Android.

CONCLUZII

iOS ar putea fi apreciat mai mult de utilizatorii care prefera aplicatii si utilitati preinstalate care isi fac realmente treaba de baza. Desi exista o varietate la fel de bogata de aplicatii software pentru iOS ca si pentru Android pentru inlocuirea sau imbunatatirea celor de baza, adevarul este ca aplicatiile preinstalate pe Android sunt mai putine decat cele de pe iOS. iOS in sine este un sistem de operare extrem de minimalist care, desi ii aduce garantia ca publicul tinta pentru care a fost creat va fi cu siguranta atins, il face totusi sa fie ceva mai putin capabil de productivitate. Toate aplicatiile integrate si majoritatea celor create pentru iOS au randament maxim, intuitivitate incredibila – pana si un copil mic poate sa le utilizeze – dar sunt create pentru utilizatorul mediu, fara o directionare catre un scop precis de functionalitate. Acesta este motivul pentru care utilizarea lui in mediul profesional este doar un capriciu, fiind, de fapt, lipsit de utilitate. Evident, exista cateva industrii (ingust orizont, totusi), unde un iPhone ar putea face fata. Printre ele, evident, industria muzicala (iTune a fost, pana una-alta, prima aplicatie cu adevarat profesionala pentru muzica, incluzand chiar de la inceput caracteristici cum ar fi catalogarea, descarcarea sau distribuirea de melodii) sau cele in care imagistica are o pondere mare.

Totusi, daca nu faceti parte din aceste industrii, atunci cand achizitionati un iPhone, ar fi bine sa o faceti pentru design, calitate hardware, grad de popularitate al echipamentului, si nu pentru ideea de productivitate in cadrul afacerii in care activati.

Android ar putea fi preferat de utilizatorii pentru care experienta unor aplicatii diferite, care insa fac acelasi lucru – diferite aplicatii de mail, mai multe aplicatii pentru ascultat muzica sau pentru editat fotografii – este mai incitanta. in general aplicatiile pentru Android necesita, in vederea configurarii pentru o experienta optima, cunostinte ceva mai avansate decat cele pentru iOS, primele fiind construite dintr-o perspectiva oarecum mai tehnica, pe cand ultimele, au o abordare ceva mai intuitiva.

Android nu are, in privinta elegantei atatea beneficii precum iOS. Fiind minimal la instalare, in general toata artileria de aplicatii vine de la producatorii independenti, iar asta duce in mod direct la o incapacitate de mentinere a unui stil de baza, a unei teme predilecte, precum iOS. in general un telefon care ruleaza cu Android si care este incarcat cu aplicatii, va arata mai mult ca un tablou de Picasso, pe cand un iPhone va aduce mai mult cu un tablou de Monet (sau Dali, cel putin asa il percep eu).

In ceea ce priveste stabilitatea, cele doup sisteme sunt oarecum departajate, o pondere mai mare avand-o iOS, dar asta si pentru ca echipamentele Android sunt utilizate mai intens, avand instalate pe ele mai multe aplicatii *third-party*.

iOS este mai putin vulnerabil decat Android, dar pentru o problema de securitate care ar putea aparea, din pacate trebuie asteptata interventia Apple, care trebuie sa scoata o actualizare de sistem, pe cand in cazul Android interventia se poate face oricand si de catre oricine, avand in vedere Licenta Apache sub care este distribuit sistemul.

BIBLIOGRAFIE

1. *OReilly Programming iOS 5 2nd (2012)*

2. *Wrox Press Mac OS X and iOS Internals, To the Apple's Core (2013)*

3. *OReilly Programming iOS 5, Fundamentals of iPhone iPad iPod touch Development 2nd (2012)*

4. *OReilly Learning iOS Programming, From Xcode to App Store 2nd (2012)*

5. *OReilly iOS 5 Programming Cookbook (2012)*

6. *OReilly Developing Enterprise iOS Applications, iPhone and iPad Apps for Companies and Organizations (2012)*

7. *www.wikipedia.com*