Intelligent Consultation Assistant

Final Report

End of term project

*by **Antonin LENFANT***

*and **Miguel NAVARRO REINOSO***

Table of Contents

1. Goal of the project

This project adapts an existing project to help to fight against the child mortality in Burkina Faso.

The existing project is call REC (Registre Electronique de Consultation) is developed over Linux environment, and is working in Netbooks with Ubuntu OS.

Intelligent Consultation Assistant (ICA hereinafter) is a new develop to create an Android application to improve the intuitive use of this system of consultation to people less formed in medical sciences, allowing touches on the screen.

For that, this development starts with a study of previous application, to understand the functionality.

Although we had as reference the old application, the design of the new one is create since zero. This version is created to foment the use of touch screen and have touchable elements. That does the learning for users very easy.

This project also has into account the destiny of the application. For that, Android tablets are a good way to reduce costs in equipment buying. The price of a tablet is sensitively lesser than a netbook used until this moment.

REC
Registre Éléctronique De Consultation
ICA
Intelligent Consultation Assistant
Eee PC 1101HA
ubuntu
android

2. State of the art of Android development solutions

We will develop here the different possibilities for developing an Android application. For each one, we will present some of the most popular tools that can be used, and detail its advantages and disadvantages.

2.1. Native application

A native application is the “official” way to develop for a mobile platform, using tools and APIs directly provided by its creator. Since they are specialized, the application will not be able to run on other platforms.

Android SDK

Using the official Android SDK provided by Google, it’s possible to create applications using the Java programming language.

The SDK is coupled to the Android Development Tools (ADT) which is composed of a plugin for the Eclipse IDE (Integrated Development Environment) adding functionality such as syntax highlighting and code completion specific for Android (including the XML resource files used for instance to build interfaces), a graphical interface builder, and debugging on an emulator or an actual device.

If a functionality of the application needs to run as fast as possible, it is possible to develop in C or C++ and use it in the application with the Native Development Kit (NDK).

Advantages

- Best possible performance, which means the app can either do more computing or run on less powerful hardware.
- Ability to make use of all the hardware (Camera, GPS, Accelerometer, Microphone, etc.) and OS APIs.
- Can be distributed via the Play Store for free or a certain price, which gives more exposure to the application since users are used to search the markets first. Furthermore, app marketplaces provide ratings, comments, rankings...
- Interacting with the hardware provides significant app potential.

Drawbacks

- Forced to use the programming language chosen by the targeted operating system's creators.
- Android apps are submitted to the Play Store with little oversight.
- Impossible to port easily the codebase to another platform.
- Necessity to learn specific skills which may not be used elsewhere.

2.2. Cross-platform application

A cross-platform application will be developed once with one code base (except maybe for platform-specific elements like the interface) and it will be possible to port it to various platforms and execute it on them without additional work.

Mobile website

Using HTML5 and its new possibilities (local storage, media playing, graphics drawing...), it's possible to create a website acting almost like an application, and which works even without Internet access with HTML5's Offline Mode.

With Android and most other smartphone operating systems, you can then put a shortcut to the website on your desktop, allowing easy access to it almost like a "real" application.

Advantages

- HTML5 is capable of significantly reducing development time, since its premise is the flexibility and simplicity in use.
- Using HTML5 for mobile applications facilitates the maintenance and support of applications, since it's the same code for all platforms.
- HTML5 is supported by virtually all browsers present in mobile terminals and will be by other devices: TVs, cars...
- Mobile websites development has a much easier learning curve and can be done by persons already experienced in web development.

Drawbacks

- Currently the browsers of mobile terminals follow different rhythms when implementing entire HTML5 specification (for instance, http://mobilehtml5.org/ shows the various differences between platforms). They also render it differently so it may not look the same on different platforms.
- Segmentation in Android and other platforms must also be taken into account when developing mobile websites because the browser's abilities depend on the version of the operating system.
- The HTML5 standard is still a draft being defined and is expected to be fully defined for 2022.
- Because the application will not have the same user interface than native apps, it will look a bit "foreign" on the device among other apps. The browser's interface will use a large part of the screen.
- HTML5 and JavaScript performance really isn't up to par with native applications, especially if you need to have animations.
- You can't access some native APIs such as notifications, sensors (cameras, geolocation, accelerometer...), contacts, file system, etc.
- Those applications can't be submitted to Google Play (ex-Android Market) or its equivalent on other platforms.
- It's almost impossible to do automatic unit testing. There are a few software, but still in their infancy and quite expensive.

Hybrid apps

They are web applications developed in HTML5 JavaScript and CSS, packaged as a native application and run in a customized application displaying the web pages without any interface around them (unlike a browser), which makes it look more like an application than a website.

The Frameworks used for development allow access to some native APIs, which can be used to extend an application's functionality by using the device's sensors and OS features.

Adobe PhoneGap

Leader of mobile cross-platform development and based on the open-source Apache Cordova project, PhoneGap allows to create mobile applications using web technologies (HTML5, JavaScript and CSS). It's the most used hybrid applications framework and as such it is well documented and supported.

The application can be exported to a good amount of mobile platforms: Android, iOS, BlackBerry, WebOS, Windows Phone 7, Symbian and Bada. Of course, the supported APIs depend on the target operating system.

Mosync

Just like PhoneGap, you develop a web application that will be packaged. The big difference is that if a functionality doesn't exist in the Mosync Framework, you can add it using C or C++.

It supports the most popular smartphone operating systems plus Moblin and MeeGo.

Since they basically use the same languages, the pros and cons are the same as mobile websites with a few changes.

Advantages

- The application can be submitted and distributed by Google Play and its equivalents.
- Since there isn't an additional interface on top of the application, the developer has access to more screen real estate, which allows for a better usability.
- Access to some native APIs exposed to JavaScript by the Framework that would not be available using only HTML5, which makes the application more integrated and permits more functionality.

Drawbacks

- Just like with HTML5, the exposed native APIs depend on the target operating system.
- There are still differences with different devices (available APIs and their implementations, HTML/CSS rendering engine) which may require a partial rewrite of the application in order to port it.

Interpreted apps

Unlike the previous cross-platform development Frameworks, interpreted apps are not developed similarly to a website, but instead using a script language which is then interpreted when the application is running on a device.

Appcelerator Titanium

Allow you to develop native applications using JavaScript code which will then be interpreted and executed by a virtual machine. You can use the platform's native controls instead or having them simulated or replaced using HTML and JavaScript.

It supports Android, iOS and Blackberry.

Rhomobile Rhodes

Rhomobile is a development suite under the MIT license created by Motorola. It allows developing multiplatform mobile apps using a MVC pattern inspired from Ruby on Rails, the views being in HTML and the controllers in Ruby. The application is then packaged and run on the device using the RubyVM interpreter.

Rhodes supports development for Android, iOS, BlackBerry, Windows Mobile and Windows Phone 7.

Again, they are the same as hybrid applications but with a few differences.

Advantages

- Better performance than hybrid applications.
- Better integration with the operating system since it's sometimes possible to use native controls when creating a graphical user interface.
- Unit testing is possible.

Drawbacks

- Sometimes proprietary or with licences that could be restrictive.
- Often less different OS you can export your application to (since it requires more work to develop an interpreter).
- Still some differences between the different mobile OS that sometimes requires changing parts of the code (for example with Titanium).

Cross-compilers

A step further from interpreted apps, cross-compilers compile the application's source code to native code, allowing it to run as fast as possible while still having a choice for the programming language.

Mono for Android/Monotouch

Xamarin, the creators of Mono, a software allowing cross-platform execution of programs created with the Microsoft .Net Framework, created Monotouch (for iOS) then Mono for Android in order to enable developers to create mobile applications using the C# programming language and Visual Studio.

This solution is popular with companies mainly developing for the Microsoft ecosystem since it uses the same tools and languages they already know. However, it is proprietary and quite expensive.

Adobe Flash Builder

Flash Builder is a development suite created by Adobe to address the lack of Flash support on smartphones. Using it, a developer can have a source code in Actionscript (Adobe Flash's scripting language) and compile it to native code for both iOS and Android.

The advantages are basically the same as native apps, with some additional drawbacks.

Advantages

- Native-like performance.
- No need to learn a new programming language.
- Unit testing is supported.

Drawbacks

- Proprietary and expensive.
- Few documentation and free support.

2.3. Choice of a solution

First, it's necessary to consider the application's specifics in order to choose a development solution.

The application will be used on tablets provided by an NGO (Terre des Hommes) and used in Burkina Faso, so it is probable that the tablets will not be high-end devices and that they might use a weak processor, which we need to take into account. As we saw, the best performance is attainable by developing either native or cross-compiled applications. Since there is not budget for development tools and cross-compilers are expensive, it's better to create a native application if we want to take this into account. There is also no plan to use devices based on another platform than Android, so multi-platform development, which is the biggest drawback of developing native applications, is not the most important factor here.

Furthermore, the application will be used for medical treatments, so it is absolutely primordial to ensure that it does a good diagnostic and follows exactly the Integrated Management of Childhood Illness (IMCI) protocol because errors could be dangerous or even fatal. Therefore, testing will be one of the focuses of this project: it should be possible to check in depth with various scenarios if the given treatment is the right one. As we saw in the previous part, native development on Android makes it easy to do unit testing, in part because it uses the Java language and there are already many tools dedicated to this purpose (such as JUnit for instance).

The application is going to be used by people who most probably never used a technological device in their lives, even less a computer or smartphone. It is then necessary to take this in consideration when designing the user interface, which has to be intuitive, simple and easy to use. In order to do this, it is necessary to have flexible enough tools that allow a good amount of customization. It is also necessary to focus, at least a bit, on animations because they are believed to help novice users better understand the interface when they are used correctly.

Again, for both those objectives, native applications are one of the best choices.

We wondered if it might be possible to re-use at least part of original application, which might have been possible if we had chosen to create a mobile website or a web application by copying the JavaScript code. However, it was developed in Ruby on Rails and consequently most of its internal logic should be in Ruby, and not in JavaScript which will be used only for presentation. Therefore, it seems unlikely that we would be able to reuse the existing code.

3. Prerequisites

In order to be able to build and deploy the application, multiple prerequisites are needed.

Android ADT Bundle

The Android ADT Bundle is a pack of applications for Windows, Linux and OSX containing all the development tools needed for Android applications development:

- Eclipse : The most popular open-source Integrated Development Environment, and the only one officially supported by the SDK
- ADT plugin : The Android development plugin for Eclipse
- Android SDK Tools : The SDK tools, containing among others the SDK updater and the Android emulator
- Android Platform-tools : Tools used to communicate with the Android Operating System
- The latest Android platform : The latest version of the Android Operating System
- The latest Android system image for the emulator : An image of this latest version usable in the emulator

Github (or Git)

Install Github for Windows, OSX or Eclipse (Linux-compatible) in order to download the sources directly from the Github repository.

You can also directly install the git package on Linux or OSX and retrieve these using the command-line.

Android Device

Although you can test the application on the Android emulator, it is better to directly use a real Android device, such as a phone or tablet, for this purpose.

Indeed, it will not only be faster but it will also better reproduce the usability of the final application. The Android device must at least use the version 2.2 of the Operating System.

Android Emulator

If you don't have an Android device at your disposal, you can use the emulator but it is best to download the Intel Android image using the SDK Updater and install the Intel HAX package located in [Android SDK Root]\extras\intel\Hardware_Accelerated_Execution_Manager to enable hardware accelerated execution, which will give you greater speeds when using the emulator.

4. Application Folders

When opening the application source code, you will find multiple folders containing different elements.

Documentation

Contains the various documentation files created during the application's development, including this one.

It also contains the documentation for the previous version of the application, in the "REC data" folder.

IMClapp

The main project of the application, it is an Eclipse project.

IMClappTest

The testing project for the application, it is also an Eclipse project. Its goal is to test if the application works successfully.

REC-laptop

The previous version of the application, in Ruby on Rails.

Resources

Images files used in the application.

Screenshots

Screenshots about some activities of the application.

5. Building

After have installed Eclipse, and Android ADT, to build the project in order to obtain the application's Android apk binary file, do as described below:

1. Open Eclipse.
2. When asked set the project folder as the Eclipse workspace.
3. Click "File" menu, and choose "Import" option.
4. In "Android" folder, click in "Existing Android Code Into Workspace".
5. Browse the root of the project folder on Github directory, choose the project called "IMCIAppActivity", and click "Finish".
6. Now, the project is shown in Project Tree. Click in "Project" menu and do a clean of the project (sometimes is needed to avoid errors).
7. Click with mouse secondary button, go to "Android Tools" menu, and choose "Export Unsigned Application Package…".
8. Set the directory to store the apk file.
9. You will find the build file IMCIapp.apk in selected directory.

6. Running the project

Emulator

To run the project using the Android emulator, do the first 3 steps of the Building process, and then follow the steps below:

1. Click the ▶ button in Eclipse's toolbar (or Run menu then click Run).
2. The first time, create a new Android device using the AVD manager. Allow it at least 512mb of ram and use Android with a version superior or equal to 2.2. In order to have a faster emulator, use an Intel Android image.
3. Select the virtual machine you want to run the application on.
4. A window opens. Wait for Android to boot, the application will then be started.

Android Device

It is better to run the project on an actual Android device. To do so, do the first 3 steps of the Building process, and then follow the steps below:

1. Go to your device's Settings.
2. If you can't see the developer options, enable them (sometimes it is necessary to go to About Phone and then click rapidly six times on the build number, although it might be something else depending on your device).
3. Go to the developer options, and enable USB debugging.
4. Plug your Android device on your computer with an USB cable, and wait for the drivers to install (download them on the manufacturer's website if necessary).
5. Click the ▶ button in Eclipse's toolbar (or Run menu then click Run).
6. Select your device in the list of running Android devices.

7. Deploy the project

First, you need the packaged application in its apk format that you obtain after following the “Building” part of the documentation.

You can then send the application’s file to the target device(s) using various ways, for instance:

- Copying the file to the target device using an USB cable, and using a File Manager to open it.
- Sending the file by e-mail or using the cloud (Dropbox for instance) and downloading the file using the target device’s internet browser.

When running the file, you might have to enable the installation of applications from other sources if it wasn’t enabled on your device. A window will warn you of the problem and take you to the location where you can enable this setting, do so and open the file again.

The application should now run on your device.

8. Testing the project

In order to ensure the smooth running of the project, we have developed tests that check various important parts of it and will report if everything works according to plan or if there is a problem.

To test the application yourself, you have to use the IMCIappTest project using the following steps:

1. Open Eclipse.
2. When asked set the project folder as the Eclipse workspace.
3. Click "File" menu, and choose "Import" option.
4. In "Android" folder, click in "Existing Android Code Into Workspace".
5. Browse the root of the project folder on Github directory, choose the project "IMCIAppTest", and click "Finish".
6. Now, the project is shown in Project Tree. Click in "Project" menu and do a clean of the project (sometimes is needed to avoid errors).
7. Click the ▶ button in Eclipse's toolbar (or Run menu then click Run).
8. In the window that appears, choose to run the project as an Android JUnit Test.
9. Wait for the tests to run.
10. A new window called JUnit will open on the left, detailing every test that was run and if it was successful or not.
11. In case of a failed test, clicking on it will show where the problem occurred, and the Failure Trace window below the JUnit results will display the exact stack trace.

9. References

http://fr.slideshare.net/peterfriese/cross-platform-mobile-development-11239246

http://es.slideshare.net/itsas_ehu/f4hc-2011-moviles

http://mashable.com/2010/08/11/cross-platform-mobile-development-tools/

http://fr.slideshare.net/kcresus/phonegap-vs-appcelerator

http://www.phonegap.com/

http://www.mosync.com/

http://www.appcelerator.com/

http://www.motorola.com/Business/USEN/Business+Product+and+Services/Software+and+Applications/RhoMobile+Suite

http://xamarin.com/monotouch

http://www.adobe.com/fr/products/flash-builder.html