TECHNICAL UNIVERSITY OF VALENCIA

DEPARTMENT OF COMPUTER ENGINEERING

Master Thesis

UbiqBIOPARC: A GPS and WIFI Based Context-Aware System for an Enhanced Guide Experience

José Vicente Sorribes Díaz

Advisor:

Dr. Juan Carlos Cano Escribá

September 2010

Abstract

This document discusses and evaluates the use of GPS, WIFI and iPhone SDK based technologies to develop UbiqBIOPARC, a context-aware system in a particular context: a new generation zoological park that has been created based on the zoo-immersion concept, submersing the visitor totally in the savage habitats. It offers appropriate contextual information to users, depending on their preferences and the environment in which they are positioned. UbiqBIOPARC is a context-aware application that provides information to zoo visitors. It combines the flexibility of iPhone SDK with the connectivity provided by WIFI, the location capabilities of GPS and the orientation offered by a compass integrated in the device.

In this document the overall architecture and the implementation steps followed to create this application are presented. We also demonstrate that a new approach to build context-aware applications with the aid of GPS and compass features is possible.

Finally, several experiments have been carried out in order to evaluate performance and system behavior. In particular, system reaction time and data download time are under study. Besides power consumption in different modes such as GPS, local, WIFI and compass modes is evaluated.

Keywords

GPS, WIFI, iPhone SDK, mobile devices, context-aware systems

Contents

List of Figures

List of Algorithms

Chapter 1
Introduction

In context-aware computing, applications may change or adapt their functions, the way information is presented, and the user interfaces depending on the context and the characteristics of the client using the system. This way the information available for the user may change depending on location, orientation or even on his personal profile, without users' intervention.

Nowadays context-aware applications exploit mobile wireless communication technologies to interconnect several computing devices and provide an intelligent environment.

Tourism and interactive guides is one of the business application areas for the development of context-aware applications. This kind of applications could help users in their personalized visits to places such as a zoo.

A lot of research has been carried out in this area. Examples of prototypes are the following: Cyberguide uses the current location of users to provide visitors with services concerning location and information. It uses infra-red technology to discover user location, although a GPS unit is used outdoor. Other systems such as GUIDE, PARCTab, InfoPad and Personal Shopping Assistant have been proposed [1]. CoolTown embodies web technologies, and Websign can offer tourist guide services although it provides user interactions for services associated with physical objects. There is also a solution called Rememberer that offers visitors of museums services for recording their visits and the location is discovered using infra-red technology and RFID sensors. Digital museum at Tokyo uses smart cards to detect the proximity of visitors and provide information on the museum objects. Other projects include infra-red and wireless LAN for location and connectivity. Dulenduè is used as a service searcher. It is based on the position of the user and the services existing around him, and can work as a tourist information point as well as a shopping consulting center. It uses a public network as telephony. The position of the user is obtained from the access point used to access to the Internet. Each access point is stored into the database with its localization. The user can search for services, shops or any data stored in the

database. It can also show a map with the position of the restaurant or a map with the nearest parkings.
Moreover, a context-aware system called UbiqMuseum based on Bluetooth and Java is available [1]. It provides context-aware information to museum visitors, using mobile phones, PDAs and laptops.

What is more, at the moment Apple is looking into location-temporary apps [2], that is this company is at the moment looking for patents (proposals) venturing in applications that automatically appear on iPhone OS when you come in proximity of certain locations. A user, who is near a supporting location, will see applications appearing on his device, providing him with multiple options and information. This utility uses WIFI technology or other means and deliver applications oriented with the current position. Apple gives two examples where it could be implemented: a library and a restaurant. It is a new approach to develop a context-aware system. The idea is that the operating system will make available an application only when the user is near a particular location. When the user is far from it, the application is automatically deleted. For example, a user can stand outside a restaurant and view their full menu on the device offering and making orders electronically, seeing the wait time and so on. The user can enter a public library and access their database to get information on books and other content in the library's catalog. When the user leaves the library or proximity of the restaurant, the app disappears (it will automatically be removed) so that you save memory and hundreds of local business apps will appear just when necessary. In figure 1, we can see how the context-aware system works.

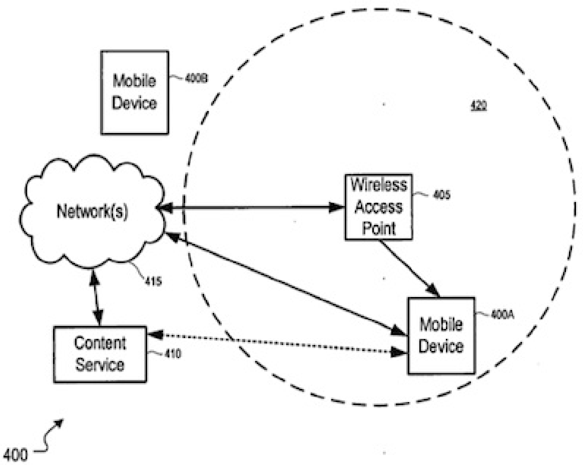


Figure 1. Apple's context-aware proposal scheme.

In this document, a new context-aware application, called UbiqBIOPARC is described, that provides context-aware information to zoo visitors using iPhone smart phones. The application provides users with updated information about which group of animals they are viewing, based on their GPS current location, the location of this group of animals and the orientation given by the compass. Information is classified in such a way that users can access to data at their level of knowledge (child, adult, student, advanced, expert) and the language they choose (Spanish, English, Valencian).

Furthermore, other options such as google maps interface are also available in order to allow users to know their current location, the position of the nearest animal and search for the animal they are particularly interested in.

In addition, a static option is also provided for the user to manually search for information. This information is structured in areas of interest (Madagascar, Equatorial Forest, Savanna, Pickleweed). If the user taps on one of these options, a list of animals, plants and landscapes is shown. The visitor can choose any of them, resulting in a query after which the information is displayed (technical records, biology, curiosities, list of videos, audio and photos). There are two main options: the local option uses a database integrated in the device, while the remote option offers updated data from a remote database server. This application is developed in order to be used in Apple iPhone mobile phones.

A remote database server has been set up, and client code has been written, providing database queries.

The proposed system uses GPS and WIFI technologies. GPS is used to discover user locations, and database GPS points around each of the areas of interest make possible to know the location of animals.

This prototype system allows us not only to confirm the correct behavior of the designed application but also to obtain experimental data to evaluate how well GPS and WIFI technologies are suited for context-aware systems.

1.1 Motivation and Objectives

The main motivation of this thesis is to develop a context-aware application that will help zoo visitors to obtain accurate information depending on their location, orientation, profile and language chosen. We will also be particularly interested in knowing if GPS is suitable for context-aware applications as well as obtaining performance evidences to discover if they can be candidate technologies for such particular context-aware applications.

This thesis overall objective is to design a context-aware application that will allow visitors to obtain accurate information depending on his preferences, location and orientation.

The main sub-objectives are:

- Database design: A database must be designed in order to store all the data concerning to the park's points of interest.
- Native application development: A context-aware application, consisting of different modes, such as local, remote and GPS modes, has to be developed.
- System deployment in the park: Prepare the system built in order to be used in the park.
- Results, functionality and tests: These will prove the system performance.

The rest of this document is organized as follows: Chapter 2 describes the system architecture. Chapter 3 presents the details of the implementation and Chapter 4 illustrates the evaluation of the proposal. Finally, in Chapter 5 some concluding remarks are given. At the end of this report, a set of appendixes to help readers understand the system, is also available.

Chapter 2
System Architecture

The overall network architecture is based on the combination of several servers using wireless technology and a communication wired network, WIFI backbone with mesh technology. The client devices (iPhone mobile phones) use WIFI and 3G technology to connect to the main server that provides information from a database.
The wired network is based on CISCO Ring single-mode Optical Fiber at 1 Gbps to connect mobile clients with the central server.
Furthermore, there is GPS coverage throughout the park. These GPS positions can be obtained by triangularization from the closest WIFI antennas, via 3G or cellular antennas.
Each iPhone device has a GPS chip and a compass, whose measures will be taken into account when developing the application.
The amount of servers laid out in the park provides users with more speed and increases performance, while all the communications are centralized in the main server.

The system considers two main software entities: client applications and the central data server. A visitor owning a GPS enabled iPhone mobile phone is the basic example of a mobile client. The GPS mode allows user to know their current position, while the database server contains GPS points surrounding each of the areas of interest. This way the device knows in which area the user is in a particular moment. The compass gives us a precise reading of the orientation (in degrees) of the device. Combining both readings, the device knows which area of animals the user is viewing in a particular moment, providing contextual information which will vary depending on the location, orientation and user profile.

Moreover, the central data server contains updated information on each of the points of interest, offering through a wireless connection all the information to the user device.

Figure 2 shows UbiqBIOPARC system architecture representation.

A mobile client, while wandering around the park, will continuously obtain his GPS location and the orientation degrees from the compass reading. Then, the system will look for information on animal GPS location in the database. An algorithm that processes these data and allows the user to know exactly which area he is viewing will be executed. Automatically, a list of animals from this area is displayed in the screen. The user can choose one of the animals and the information on it will be displayed. This information consists of technical records, biology, curiosities, videos, audios and photos. This information provided by the server depends on the user profile and language chosen in the initial configuration process which will be sent in the request.

The user can change his profile and language at any time, going backwards to the corresponding iPhone interface.

This system is designed to allow any user to connect to a wireless network.

The architecture is organized into three levels: (i) the main server, (ii) the wireless network and (iii) the clients:

- **Main server**: It contains a database and an ASP script running. All clients' requests will be done through this script. This server is connected to the network backbone.
- **Wireless network**: It consists of a set of access points which allows users to connect to the server. This wireless network is based on 802.11 N.
- **Clients**: They can connect to the server with iPhone mobile phones.

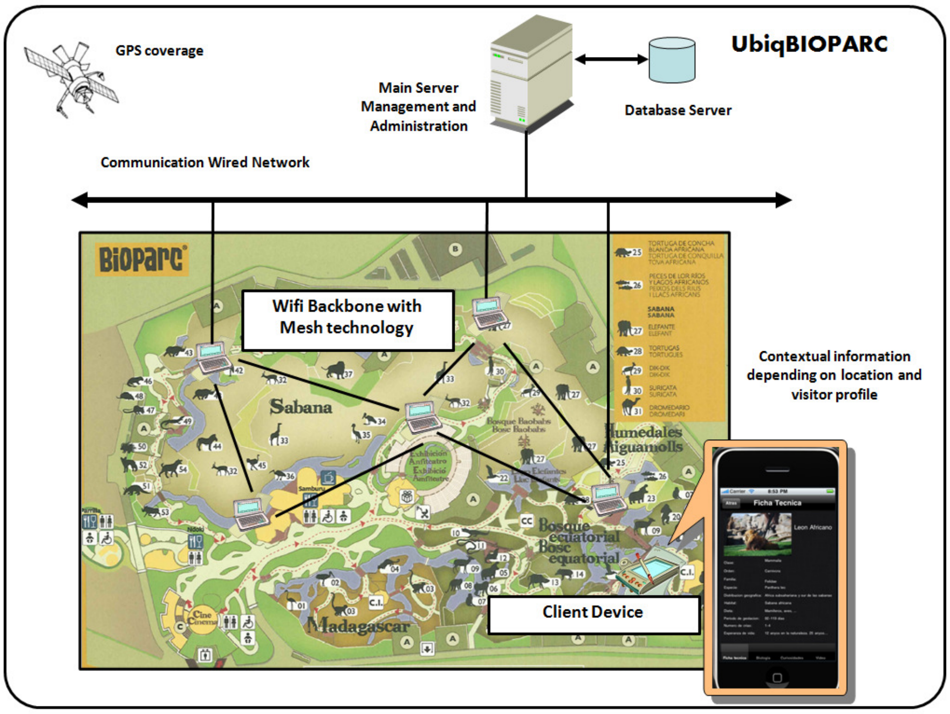

GPS coverage
Main Server Management and Administration
Database Server
UbiqBIOPARC
Communication Wired Network
Wifi Backbone with Mesh technology
Contextual information depending on location and visitor profile
Client Device

Figure 2. The UbiqBIOPARC system architecture.

Chapter 3
UbiqBIOPARC system design and implementation

3.1 Overall Overview

The system is composed of clients, integrated in the iPhone devices, and a unique server located in the park, which contains the remote database and an interface with clients composed of an ASP script. There can be multiple clients connected to the server at a particular moment. The clients use GPS utilities (location), compass readings (orientation) and a local database integrated in the device.
The databases contain information on park's points of interest. The remote database can be updated whenever the administrator desires, whereas the local database always contains the same information. However, code has been written to get information from the remote database and store it in the local database. This process is carried out automatically. This allows local database management when required.
Local database information will always be available, even if iPhone connection is out of order. Remote database information will be available at home, on a train, or wherever the user is located, but iPhone connection failures could disable this feature.
The system architecture can be seen in Figure 3.

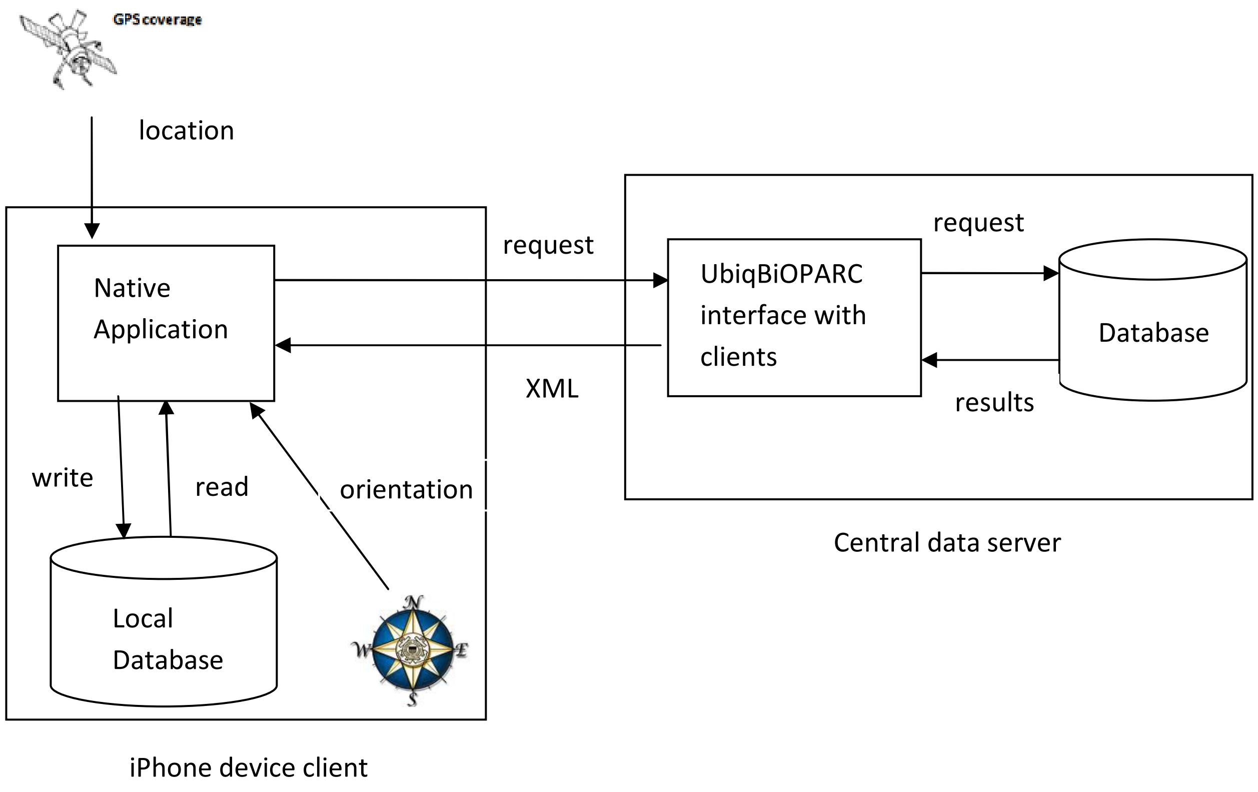


Figure 3. UbiqBIOPARC system architecture.

3.1.1 Central data server

The central data server main function is to attend requests from the clients and offer information on points of interest to them from a SQL Server 2005 database. Other database implementations could be possible, as this implementation is transparent to the client side. The only difference would be in creating the ASP script. In this case SQL Server 2005 has been chosen due to its features. The database contains all the information related to the points of interest (animals, plants and landscapes) of the park. Besides the database, the server contains an ASP script that allows the native application to make queries to the remote database. The resulting data is sent to the application. For this purpose, XML files generated by the ASP script are used. This architecture can be seen in figure 3.
This server is running all the time and it is waiting for client queries.

The native application is running in the iPhone mobile phone and, when the WIFI option is selected, it sends select queries to the database server as external information is required. Otherwise, it uses the information from the local database.

3.1.2 Database

The **database** has been designed and developed to include all the necessary information on points of interest, and GPS points taken, that the application will manage in order to be shown to the user in its different modes.

The database design is shown in figure 4.

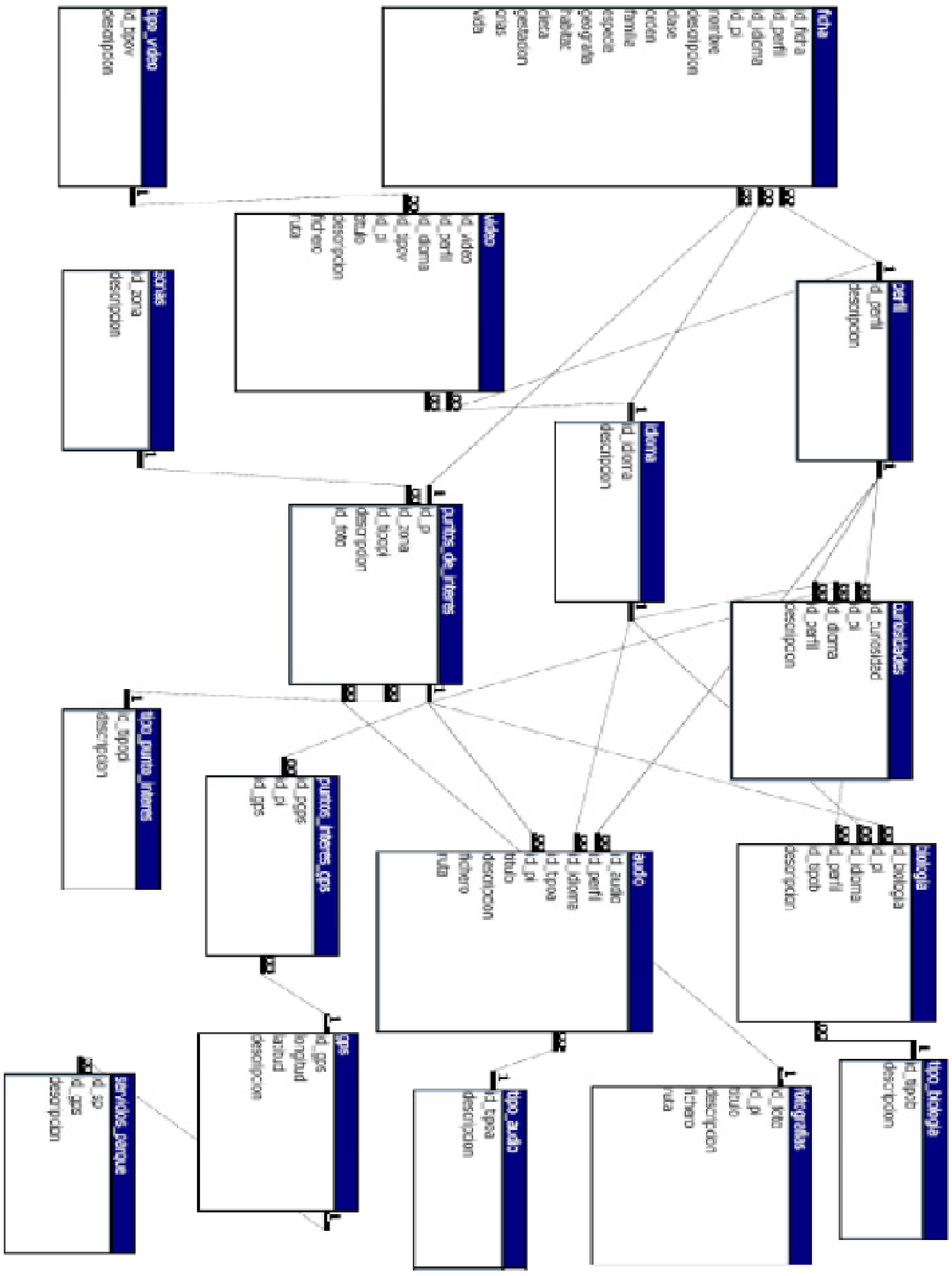


Figure 4. Database design.

The database consists of 17 tables, with different information distributed among them.

- PUNTOS_DE_INTERES contains information on the different points of interest, that is, animals, plants and landscapes, such as the area in which it is living, the type (animal, plant or landscape), description and the associated photo.
- FICHA contains the following information: id_perfil, id_idioma and id_pi to point to other tables. The essential information is structured in the rest of the fields. Nombre is the name of the point of interest id_pi for the id_idioma specified. Description is a brief description, while clase, orden, familia, especie, geografia, habitat, dieta, gestacion, crias, vida is technical information on the point of interest and language specified.
- BIOLOGIA contains information on biology for a certain point of interest for each profile and language.
- CURIOSIDADES contains information on curiosities for a certain point of interest and different profiles and languages.
- VIDEO supports information on videos for points of interest such as title, description, file name or url, classified in different languages and profiles.
- AUDIO contains the same information as the VIDEO table, but in this case for audio files.
- FOTOGRAFIAS contains title, description, file name and url for photos on different points of interest.
- GPS contains information on GPS points, as latitude, longitude and description.
- PUNTOS_INTERES_GPS contains the association of GPS points and points of interest, so a GPS point could correspond to several points of interest while a point of interest has several GPS points from which it can be seen.
- SERVICIOS_PARQUE contains the GPS position and description for various park services like restaurants, toilets or information point.
- PERFIL describes the different user's profiles.
- ZONAS describes the various areas of the park.
- IDIOMA describes the different languages supported.
- TIPO_VIDEO contains a brief description on each type of video.
- TIPO_AUDIO describes briefly each type of audio.
- TIPO_BIOLOGIA describes briefly each type of biology.
- TIPO_PUNTO_INTERES describes each type of point of interest.

The database has been designed in such a way that it is compatible with other devices' applications. In particular, Android and Mobile 7 applications related to this thesis use the same design.

The information in this database can be added, modified, deleted or viewed whenever an administrator desires, so that the information is completely up to date.

For this purpose, a **web-based application** has been designed and developed so that the administrator may connect to the server and maintain the database easily. This utility can be used by introducing the following URL in the browser.
http://merluza.grc.upv.es/Biopark/Yassine/Default.aspx
Its different interfaces can be seen below.

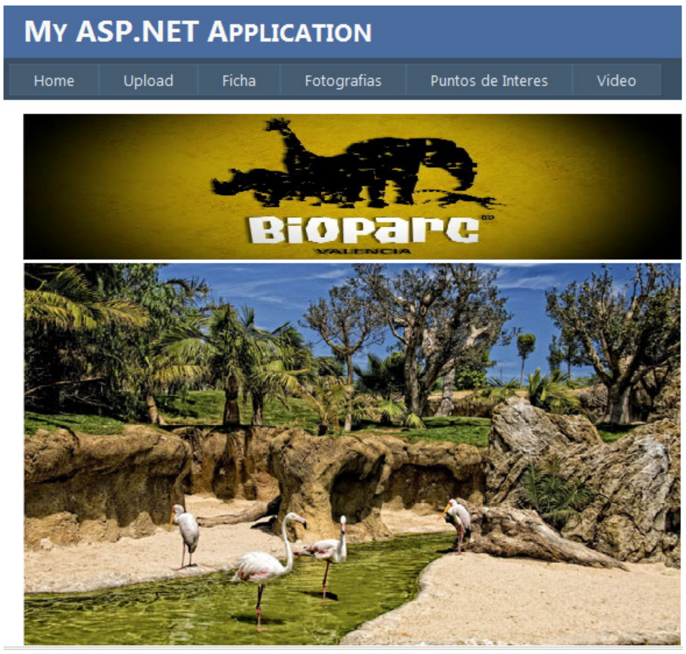


Figure 5. Application's home page.

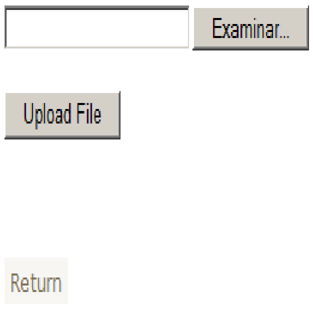
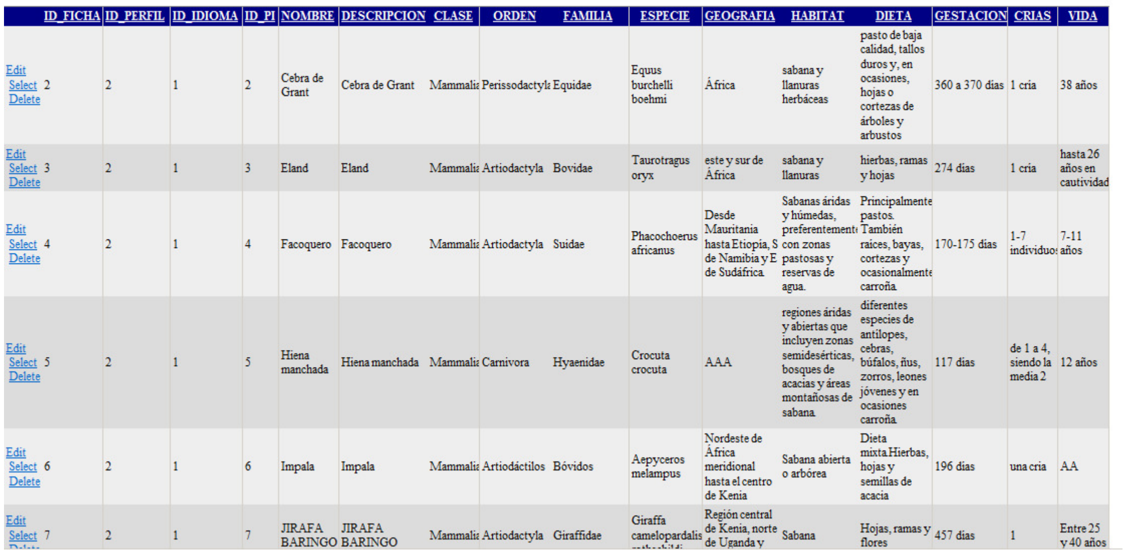
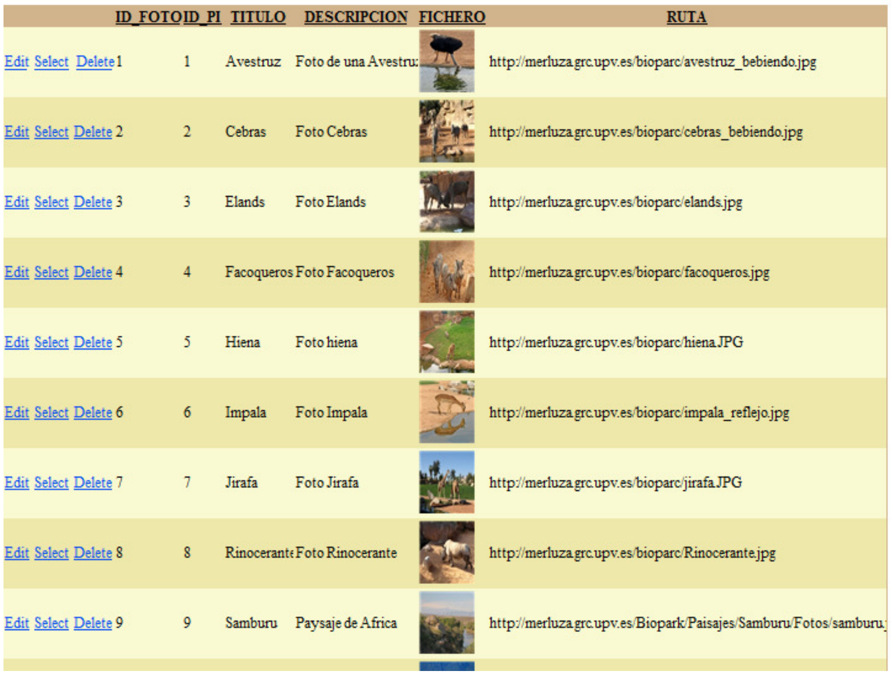


Figure 6. Upload file interface.



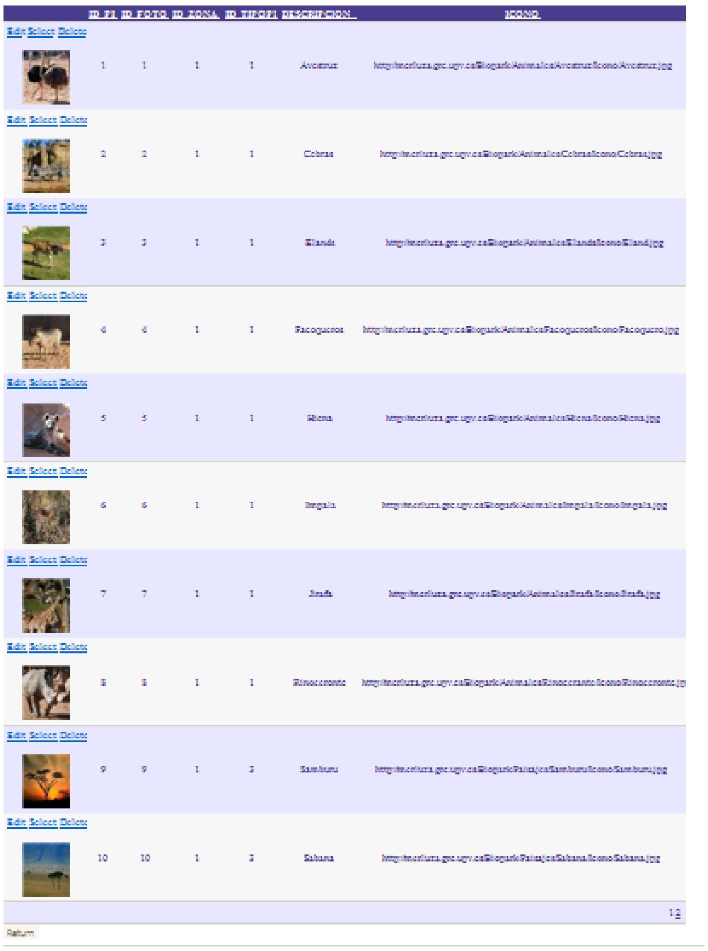
ID_FICHA	ID_PERFIL	ID_IDIOMA	ID_PI	NOMBRE	DESCRIPCION	CLASE	ORDEN	FAMILIA	ESPECIE	GEOGRAFIA	HABITAT	DIETA	GESTACION	CRIAS	VIDA
Edit Select Delete 2	2	1	2	Cebra de Grant	Cebra de Grant	Mammalia	Perissodactyla	Equidae	Equus burchelli boehmi	África	sabana y llanuras herbáceas	pasto de baja calidad, tallos duros y, en ocasiones, hojas o cortezas de árboles y arbustos	360 a 370 dias	1 cria	38 años
Edit Select Delete 3	2	1	3	Eland	Eland	Mammalia	Artiodactyla	Bovidae	Taurotragus oryx	este y sur de África	sabana y llanuras	hierbas, ramas y hojas	274 dias	1 cria	hasta 26 años en cautividad
Edit Select Delete 4	2	1	4	Facoquero	Facoquero	Mammalia	Artiodactyla	Suidae	Phacochoerus africanus	Desde Mauritania hasta Etiopia, S de Namibia y E de Sudáfrica	Sabanas áridas y húmedas, preferentement con zonas pastosas y reservas de agua.	Principalmente pastos. También raices, bayas, cortezas y ocasionalmente carroña	170-175 dias	1-7 individuos	7-11 años
Edit Select Delete 5	2	1	5	Hiena manchada	Hiena manchada	Mammalia	Carnivora	Hyaenidae	Crocuta crocuta	AAA	regiones áridas y abiertas que incluyen zonas semidesérticas, bosques de acacias y áreas montañosas de sabana	diferentes especies de antilopes, cebras, búfalos, ñus, zorros, leones jóvenes y en ocasiones carroña	117 dias	de 1 a 4, siendo la media 2	12 años
Edit Select Delete 6	2	1	6	Impala	Impala	Mammalia	Artiodáctilos	Bóvidos	Aepyceros melampus	Nordeste de África meridional hasta el centro de Kenia	Sabana abierta o arbórea	Dieta mixta Hierbas, hojas y semillas de acacia	196 dias	una cria	AA
Edit Select 7	2	1	7	JIRAFA BARINGO	JIRAFA BARINGO	Mammalia	Artiodactyla	Giraffidae	Giraffa camelopardalis	Región central de Kenia, norte de Uganda y	Sabana	Hojas, ramas y flores	457 dias	1	Entre 25 y 40 años

Figure 7. Technical Record’s management interface.



	ID_FOTO	ID_PI	TITULO	DESCRIPCION	FICHERO	RUTA
Edit Select Delete	1	1	Avestruz	Foto de una Avestruz		http://merluza.grc.upv.es/bioparc/avestruz_bebiendo.jpg
Edit Select Delete	2	2	Cebras	Foto Cebras		http://merluza.grc.upv.es/bioparc/cebras_bebiendo.jpg
Edit Select Delete	3	3	Elands	Foto Elands		http://merluza.grc.upv.es/bioparc/elands.jpg
Edit Select Delete	4	4	Facoqueros	Foto Facoqueros		http://merluza.grc.upv.es/bioparc/facoqueros.jpg
Edit Select Delete	5	5	Hiena	Foto hiena		http://merluza.grc.upv.es/bioparc/hiena.JPG
Edit Select Delete	6	6	Impala	Foto Impala		http://merluza.grc.upv.es/bioparc/impala_reflejo.jpg
Edit Select Delete	7	7	Jirafa	Foto Jirafa		http://merluza.grc.upv.es/bioparc/jirafa.JPG
Edit Select Delete	8	8	Rinocerante	Foto Rinocerante		http://merluza.grc.upv.es/bioparc/Rinocerante.jpg
Edit Select Delete	9	9	Samburu	Paysaje de Africa		http://merluza.grc.upv.es/Biopark/Paisajes/Samburu/Fotos/samburu

Figura 8. Photo’s management interface.

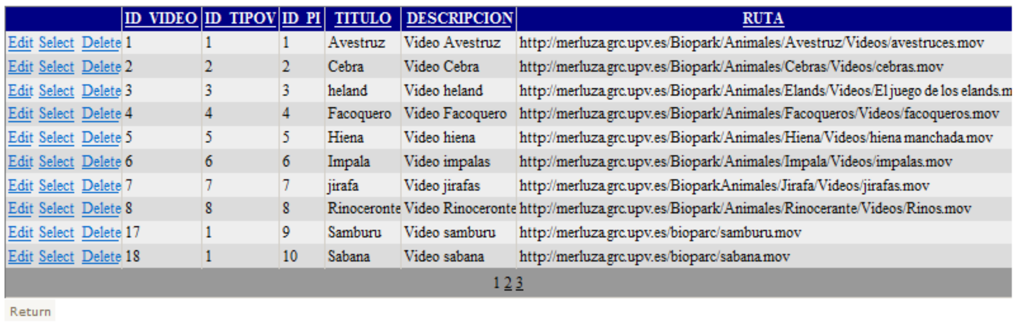


	ID_PI	ID_FOTO	ID_ZONA	ID_TIPOPI	DESCRIPCION	ICONO
Edit Select Delete	1	1	1	1	Avestruz	http://merluza.grc.upv.es/Biopark/Animales/Avestruz/Icono/Avestruz.jpg
Edit Select Delete	2	2	1	1	Cebras	http://merluza.grc.upv.es/Biopark/Animales/Cebras/Icono/Cebras.jpg
Edit Select Delete	3	3	1	1	Elands	http://merluza.grc.upv.es/Biopark/Animales/Elands/Icono/Eland.jpg
Edit Select Delete	4	4	1	1	Facoqueros	http://merluza.grc.upv.es/Biopark/Animales/Facoqueros/Icono/Facoquero.jpg
Edit Select Delete	5	5	1	1	Hiena	http://merluza.grc.upv.es/Biopark/Animales/Hiena/Icono/Hiena.jpg
Edit Select Delete	6	6	1	1	Impala	http://merluza.grc.upv.es/Biopark/Animales/Impala/Icono/Impala.jpg
Edit Select Delete	7	7	1	1	Jirafa	http://merluza.grc.upv.es/Biopark/Animales/Jirafa/Icono/Jirafa.jpg
Edit Select Delete	8	8	1	1	Rinoceronte	http://merluza.grc.upv.es/Biopark/Animales/Rinocerante/Icono/Rinoceronte.jp
Edit Select Delete	9	9	1	2	Samburu	http://merluza.grc.upv.es/Biopark/Paisajes/Samburu/Icono/Samburu.jpg
Edit Select Delete	10	10	1	2	Sabana	http://merluza.grc.upv.es/Biopark/Paisajes/Sabana/Icono/Sabana.jpg

1 2

Return

Figura 9. Points of interest management interface.



	ID_VIDEO	ID_TIPOV	ID_PI	TITULO	DESCRIPCION	RUTA
Edit Select Delete	1	1	1	Avestruz	Video Avestruz	http://merluza.grc.upv.es/Biopark/Animales/Avestruz/Videos/avestruces.mov
Edit Select Delete	2	2	2	Cebra	Video Cebra	http://merluza.grc.upv.es/Biopark/Animales/Cebras/Videos/cebras.mov
Edit Select Delete	3	3	3	heland	Video heland	http://merluza.grc.upv.es/Biopark/Animales/Elands/Videos/El juego de los elands.m
Edit Select Delete	4	4	4	Facoquero	Video Facoquero	http://merluza.grc.upv.es/Biopark/Animales/Facoqueros/Videos/facoqueros.mov
Edit Select Delete	5	5	5	Hiena	Video hiena	http://merluza.grc.upv.es/Biopark/Animales/Hiena/Videos/hiena manchada.mov
Edit Select Delete	6	6	6	Impala	Video impalas	http://merluza.grc.upv.es/Biopark/Animales/Impala/Videos/impalas.mov
Edit Select Delete	7	7	7	jirafa	Video jirafas	http://merluza.grc.upv.es/BioparkAnimales/Jirafa/Videos/jirafas.mov
Edit Select Delete	8	8	8	Rinoceronte	Video Rinoceronte	http://merluza.grc.upv.es/Biopark/Animales/Rinocerante/Videos/Rinos.mov
Edit Select Delete	17	1	9	Samburu	Video samburu	http://merluza.grc.upv.es/bioparc/samburu.mov
Edit Select Delete	18	1	10	Sabana	Video sabana	http://merluza.grc.upv.es/bioparc/sabana.mov

1 2 3

Return

Figure 10. Video management interface.

This web-based application offers remote database management. Besides, the administrator could manage the database connecting to the server by other means whenever is necessary.

3.1.3 Database server interface with clients

The database server interface with clients is composed of an **ASP script**.
The native application, when the remote mode is enabled and information is required, will send SQL sentences (requests) to the database server and the ASP script will return a XML file to the application. See Figure 3. This XML file is generated by the ASP script with the information provided by the database. Then the XML file is parsed by the application's XML parser to extract the information and view it in the interfaces.
The native application will call the ASP script, passing two parameters: the name of the table and the SQL sentence.
This way, the call to the ASP script will have the following structure:
http://serverroute/aspfile.asp?tabla=tablename&consulta=sqlsentence
In this thesis a database has been designed and stored in a server called *merluza.grc.upv.es*, and an ASP script called *BBDDtoXML.asp* has been created.
An example of calling the ASP script to query the *FICHA* table could be:
http://merluza.grc.upv.es/bioparc/BBDDtoXML.asp?tabla=FICHA&consulta=select%20*%20from%20FICHA
This means that we are connecting to the *merluza.grc.upv.es* server, using the asp script called *BBDDtoXML.asp* and making a query to the *FICHA* table with the SQL sentence *select * from FICHA*. The *%20* symbol is added by the browser to indicate that there comes a space.
The ASP script is designed in such a way that it makes a query to the database and it generates a resulting XML file with the results. The XML file contains the information structured to be used in the application.
The ASP script has two input parameters: the table name and the SQL sentence.
Basically, the main goal of the ASP script is to generate XML files containing the resulting information from the database in a structure that can be easily managed by the native application. It connects to the database, executes the SQL sentence and writes the resulting information in a XML file format.
This script's code can be seen in Algorithm 1. More information may be consulted in [19], [30] and [31].

Algorithm 1. The ASP code

```
<% @ LANGUAGE="VBSCRIPT" Codepage=65001%>   'set the language
<%
Dim xmlDoc
Dim tabla, consulta                          'two parameters
tabla = request.Querystring("tabla")
consulta = request.Querystring("consulta")
Response.ContentType = "text/xml"            'set content type
Response.CharSet= "utf-8"                    ' utf-8 format
'Open the database
Set Conn = Server.CreateObject("ADODB.Connection")
Conn.Open "Provider=sqloledb;SERVER=localhost;DATABASE=Alex;UID=ALEX;PWD=ALEX;"
'Set the SQL sentence
```

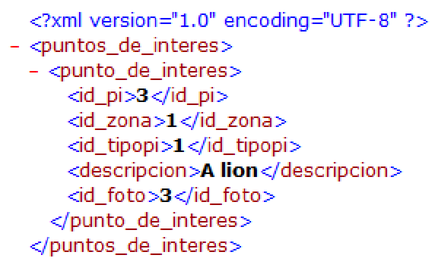
```
sSQL = consulta
‘Execute the query
set RS=createobject("ADODB.Recordset")
RS.open sSQL,Conn
%>
<%
‘ Generate the XML depending on the name of the table
comillas = Chr(34)
Response.write("<?xml version=" & comillas)
Response.write("1.0" & comillas)
Response.write(" encoding=" &comillas)
Response.write("UTF-8" &comillas)
Response.write(" ?>")

if tabla="PUNTOS_DE_INTERES" Then
‘XML for table PUNTOS_DE_INTERES
Response.write("<puntos_de_interes>")
Do While Not RS.Eof
    Response.write("<punto_de_interes>")
    Response.write("<id_pi>")
    Response.write RS("ID_PI")
    Response.write("</id_pi>")
    Response.write("<id_zona>")
    Response.write RS("ID_ZONA")
    Response.write("</id_zona>")
    Response.write("<id_foto>")
    Response.write RS("ID_FOTO")
    Response.write("</id_foto>")
    Response.write("<id_tipopi>")
    Response.write RS("ID_TIPOPI")
    Response.write("</id_tipopi>")
    Response.write("<descripcion>")
    Response.write RS("DESCRIPCION")
    Response.write("</descripcion>")
    Response.write("</punto_de_interes>")
RS.MoveNext
Loop
RS.close
Response.write("</puntos_de_interes>")
'Close the connection

‘Then, do the same for the rest of the tables
ElseIf tabla="FOTOGRAFIAS" Then
        ‘For FOTOGRAFIAS ……
ElseIf tabla="SERVICIOS_PARQUE" Then
          ‘For SERVICIOS_PARQUE ….
ElseIf tabla="GPS" Then
          ‘For GPS table ……
ElseIf tabla="TIPO_PUNTO_INTERES" Then
          ‘For TIPO_PUNTO_INTERES table  ….
ElseIf tabla="PUNTOS_INTERES_GPS" Then
          ‘For PUNTOS_INTERES_GPS table ….
ElseIf tabla="ZONAS" Then
```

```
        ‘For ZONAS table ….
ElseIf tabla="TIPO_AUDIO" Then
        ‘For TIPO_AUDIO table …..
ElseIf tabla="CURIOSIDADES" Then
        ‘For CURIOSIDADES table …
ElseIf tabla="AUDIO" Then
        ‘For AUDIO table …..
ElseIf tabla="TIPO_BIOLOGIA" Then
        ‘For TIPO_BIOLOGIA table …
ElseIf tabla="BIOLOGIA" Then
        ‘For BIOLOGIA table …
ElseIf tabla="IDIOMA" Then
        ‘For IDIOMA table ….
ElseIf tabla="FICHA" Then
        ‘For FICHA table ….
ElseIf tabla="PERFIL" Then
        ‘For PERFIL table ….
ElseIf tabla="VIDEO" Then
         ‘For VIDEO table …
ElseIf tabla="TIPO_VIDEO" Then
        ‘For TIPO_VIDEO table ….
End If
%>
```

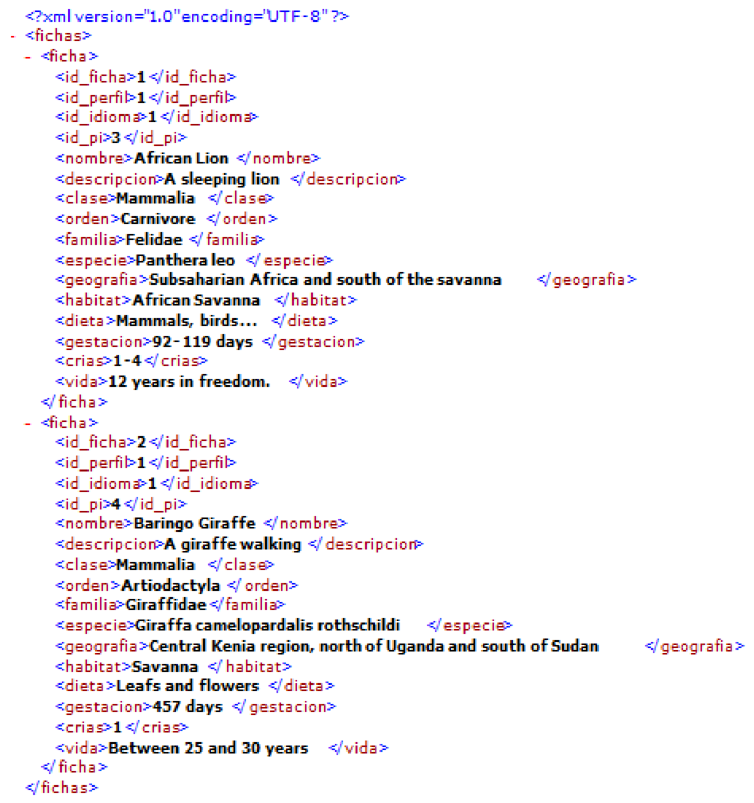
The complete code is a bit too long to be included in this document but, essentially, for each table the same process as for *PUNTOS_DE_INTERES* has to be carried out to generate the other XMLs.

Checking the structure of the corresponding XML file, helps to decide which code should be written. This XML files will be shown next.

The **XML files** must follow a well-defined structure that can be easily obtained from the database.
First of all, these XML files structure must be defined. For this application, the client is only allowed to make queries to the database. The structure must be different depending on which table we are going to make the query at, and which SQL sentence we are going to use.
The XML files must have the same structure as the following examples:



```
  <?xml version="1.0" encoding="UTF-8" ?>
- <puntos_de_interes>
  - <punto_de_interes>
      <id_pi>3</id_pi>
      <id_zona>1</id_zona>
      <id_tipopi>1</id_tipopi>
      <descripcion>A lion</descripcion>
      <id_foto>3</id_foto>
    </punto_de_interes>
  </puntos_de_interes>
```



```
  <?xml version="1.0" encoding="UTF-8" ?>
- <fichas>
  - <ficha>
      <id_ficha>1</id_ficha>
      <id_perfil>1</id_perfil>
      <id_idioma>1</id_idioma>
      <id_pi>3</id_pi>
      <nombre>African Lion</nombre>
      <descripcion>A sleeping lion</descripcion>
      <clase>Mammalia</clase>
      <orden>Carnivore</orden>
      <familia>Felidae</familia>
      <especie>Panthera leo</especie>
      <geografia>Subsaharian Africa and south of the savanna</geografia>
      <habitat>African Savanna</habitat>
      <dieta>Mammals, birds...</dieta>
      <gestacion>92-119 days</gestacion>
      <crias>1-4</crias>
      <vida>12 years in freedom.</vida>
    </ficha>
  - <ficha>
      <id_ficha>2</id_ficha>
      <id_perfil>1</id_perfil>
      <id_idioma>1</id_idioma>
      <id_pi>4</id_pi>
      <nombre>Baringo Giraffe</nombre>
      <descripcion>A giraffe walking</descripcion>
      <clase>Mammalia</clase>
      <orden>Artiodactyla</orden>
      <familia>Giraffidae</familia>
      <especie>Giraffa camelopardalis rothschildi</especie>
      <geografia>Central Kenia region, north of Uganda and south of Sudan</geografia>
      <habitat>Savanna</habitat>
      <dieta>Leafs and flowers</dieta>
      <gestacion>457 days</gestacion>
      <crias>1</crias>
      <vida>Between 25 and 30 years</vida>
    </ficha>
  </fichas>
```

As we can see, the first XML file example is the result to a query to the *puntos_de_interes* table, with just one register, which consists of the following fields: *id_pi, id_zona, id_tipopi, descripcion, id_foto*.
The second XML file example is the result to a query to the *ficha* table, with two registers, which consist of the following fields: *id_ficha, id_perfil, id_idioma, id_pi, nombre, descripcion, clase, orden, familia, especie, geografia, habitat, dieta, gestacion, crias, vida*.
The rest of the XML files' structure is built the same way, consulting the fields from the table in particular and building the XML. The complete list of the possible XML files can be seen in Appendix 2.

More information on XML may be consulted in [5], [6], [23] and [25].

Once we know how to carry out the queries to the database, and generate the XML results by developing an ASP script, we can see how the client side works.

3.1.4 iPhone device client

The iPhone device client receives location data via GPS, compass measurement, and contains the native application and a local database.
The information received is processed in the native application whenever the user location is required. The compass readings are also used in the application whenever a user orientation is necessary.
Finally, a local database contains information on points of interest and GPS points.
The information on points of interest has a similar structure to that used in the remote database, whereas the GPS information is structured to be perfectly used in the particular algorithms when required.
When the user selects remote option, the information stored in this local database is used to be managed and displayed in the interfaces. Otherwise queries are made to the remote database in order to retrieve the information needed.

The information on GPS points is structured in subareas and follows the structure that can be seen in figure 22. A subarea is composed of multiple points of interest. Multiple GPS points are taken around each subarea.

When the "GPS searcher" mode is chosen, its algorithms use information on GPS points corresponding to animals. The information stored in the database is then used to achieve this functionality, that is to locate the nearest position of the animal the user is looking for. The same happens when the "Nearest animal" mode is used. In this particular case, the information stored in the database is also used to determine which is the nearest GPS point, that is the minimum distance point from the user to all the points stored in the database.

While combining the algorithms displayed with the compass measurement utility, the context-aware mode achieves its main goal.

What is more, a **script** which takes all the information stored in the tables from the remote database and stores it in the local database, has been written within this thesis and is available to be used. This way, an administrator can introduce, modify or delete data in the remote database, and by means of this script, it can be easily and quickly stored in the local database.
This script could also proof whether making queries remotely just when necessary is better than storing all the data in the local database when application starts or not.

In the native application, the user has to choose his preferences on language and level of knowledge.
So information can be displayed using 3 possible ways, implemented in this thesis:

1) The user selects the area, and the point of interest he is particularly keen on. Then information on it will be displayed. This option has the following drawback. The user has to search for the information manually among all the areas and points of interest. It is an easy process but the user must do a great effort on looking for information.
2) The user selects the animal within a search field, resulting on his location on a map and the information on it. The same happens with the nearest animal option. It requires less effort from the user but this process still can be improved.
3) The user just selects the context-aware option. After that, the application will display information without user's intervention. Depending on his location and orientation the information will vary. This option provides the user with accurate information and simplifies the searching process. It minimizes the user effort on searching for information.

3.1.5 Additional remarks

An experimental result has to be mentioned before concluding this chapter. First of all the multimedia information was accessed from the remote database and displayed on the device interfaces. After testing it deeply, we noticed that the photos took a long time to be downloaded, the videos could not be properly seen due to high time response showing static photos that changed after a lot of time, and audio files couldn't be listened properly. Although full access to remote multimedia files is also implemented, we finally decided to use locally stored multimedia files and remote textual information. This way the videos can be seen properly without problems and audio can be listened with high quality, whereas textual information can be easily updated as needed. The file name and url information on multimedia files is also accessed from the remote database. This way this multimedia information is linked to the user preferences on language and level of knowledge.
In chapter 4, an experiment has been carried out to evaluate both approaches, storing images in the local database and remote database, and comparing their results.

Another experimental result is that the GPS measures given by the device is not always accurate, introducing an error margin of more than 40 or 50 metres in some occasions. The first GPS measure is taken from a local cache memory introducing the corresponding GPS error. After that measure it gives a non reliable measure. As time passes, the measure gets better and better. What is more, there are different accuracy options within GPS location in iPhone: 1km, 100 metres, 10 metres, best. Although choosing the best accuracy one, experiments leads us to the conclusion that user location reading varies from the real one within a few metres.
So the algorithms that find out the nearest animal will not always be completely accurate, although they will find the nearest animal position with a little margin of error (introduced by the GPS chip reading). Neither will be the context-aware utility for the same reason. In this sense two approaches have been presented before. One of

them gives completely accurate results, and never fails because the algorithm does not depend on user location, and the other one introduces errors depending on the GPS location accuracy. These two algorithms will be further evaluated and compared in chapter 4.
Furthermore experiments prove that compass readings are accurate for this thesis purposes, allowing the user to use the context-aware functionality with 100% accuracy.

Another point that has to be taken into account while designing such a system is power consumption. In that sense, GPS management must be designed so that power consumption is minimized. Taking GPS measures continuously will result in a great accuracy, while consuming a lot of power. Otherwise, we could take measurements when the user has moved lots of metres. This would give us poor accuracy but it would save a lot of power. The same happens with orientation measurements. In this thesis a timer is raised when the user moves 5 metres giving good positioning results. The rest of the time the device is asleep, so that the power consumption is minimized. In chapter 4 the power consumption will be further analyzed. What is more, the system has to read the database, run the algorithms and give positioning and orientation results. This needs a gap of time to make calculations. A margin of time of about 5 metres and about 45 degrees of orientation change is appropriate to offer the results properly. Otherwise a new GPS position reading would be generated while the algorithms are still calculating using the previous one. As the algorithms need a time margin, the timer can’t be raised until a brief period of time has passed. A margin of 5 metres and 45 degrees is suitable for this thesis.

3.2 UbiqBIOPARC system functionality

According to its functionality the application can be divided into several parts, such as location in google maps, local and remote modes, which will be further explained later.
So, additionally to the context-aware functionality, the users can search for a particular animal location in a google map, or automatically locate the nearest animal in a google map. Besides, local or updated remote information can be obtained from a database in the device or a remote database server.

The application was developed using iPhone SDK 3.1. In particular, the context-aware module uses the *CoreLocation* framework into the SDK, *CLLocationManager* class to obtain the GPS location and the compass orientation.

3.2.1 Local mode:

It is the most basic one.
Users can manually search for multimedia information about animals, plants and landscapes.
The module consists of several interfaces that allow users to navigate through the information.
First of all, a Welcome view is displayed.
The user can set the language the information will be translated to, his level of knowledge and choose a WIFI option. At the moment, 3 languages (Spanish, English and Valencian) and 5 profiles (Child, Adult, Student, Advanced and Expert) are implemented. These interfaces can be seen in figure 11.
The WIFI switch allows users to choose between local data option and up to date remote data option. A remote database, whose information is available in the application by choosing the WIFI option, is continuously been managed and offers up to date information. WIFI OFF option must be selected to get local information. This provides information to the user even when the phone connectivity is out of order.
The local information is stored in a sqlite3 ([3], [4] and [26]) database that can be easily maintained, modified, deleted or expanded. There are two ways that can be used in order to manage the sqlite3 database.
The first one uses the terminal to execute SQL sentences, such as select, insert, update or drop.
To open or create the database called *bioparc.db* the administrator should write the following sentence:

```
$sqlite3 bioparc.db
```

To create new tables the administrator has to write the appropriate sentences.
An example of creating a table is shown.

```
$create table biologia (id INTEGER PRIMARY KEY, idAnimal INTEGER,
idIdioma INTEGER, idPerfil INTEGER, descripcion TEXT);.
```

The main sentences used to add, view, modify or delete data are shown in the following example.

```
$insert into biologia (id,idAnimal,idIdioma,idPerfil,descripcion) values
(1,1,1,1,'The African Lion is the king');
$select (descripcion) from biologia where id=1;
$update biologia set descripcion='The African Lion is not the king' where
id=1;
$drop table biologia.
```

Once the information is introduced into the local database, the application must be built and run in the device.

Another option within sqlite3 management, consists in the usage of Sqlite Manager software [26], which offers a graphical interface to the user, making it easy to manage and update the database.

Sqlite3 is the most suitable option in order to store huge amounts of data in a device, and it is used in this thesis due to its flexibility and ability to store a lot of information. However other technologies could be used to store the information. Property lists and NSMutableDictionary are two of the most important ones.

Next users can choose the area they are particularly keen on. This can be seen in figure 11.

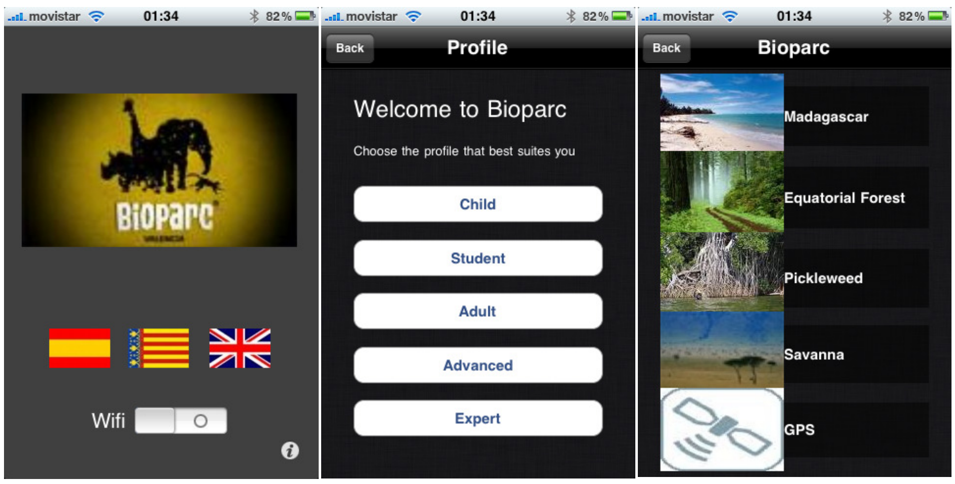


Figure 11. Welcome view, language, WIFI, profile and area options.

Depending on the profile and the language the user has chosen, the information shown will vary. These two parameters are taken into account along the rest of the application. Users are allowed to change them in any particular moment if their preferences have changed.

After choosing the area, a new interface appears showing the list of animals, plants and landscapes corresponding to this area. This can be seen in figure 12. The information is filtered so that the list of points of interest shown corresponds to the language and the area. A photo, point of interest's name and brief description is shown.

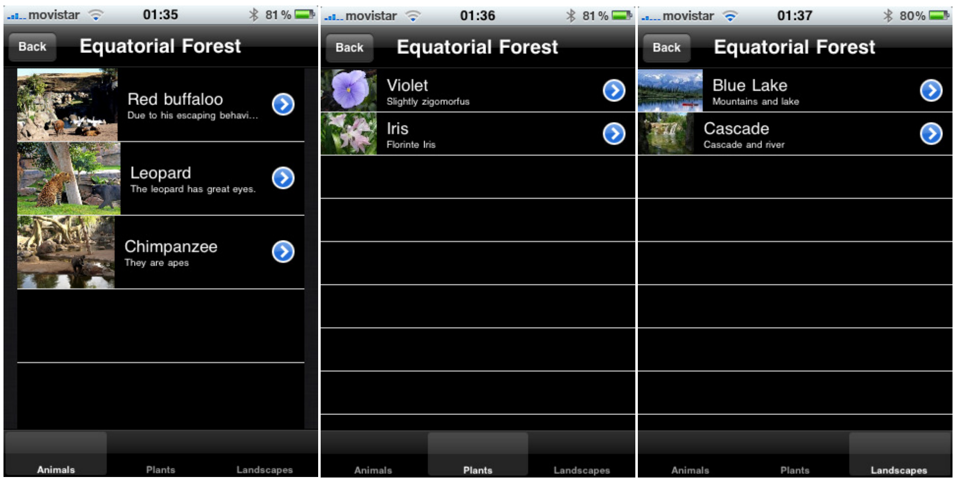
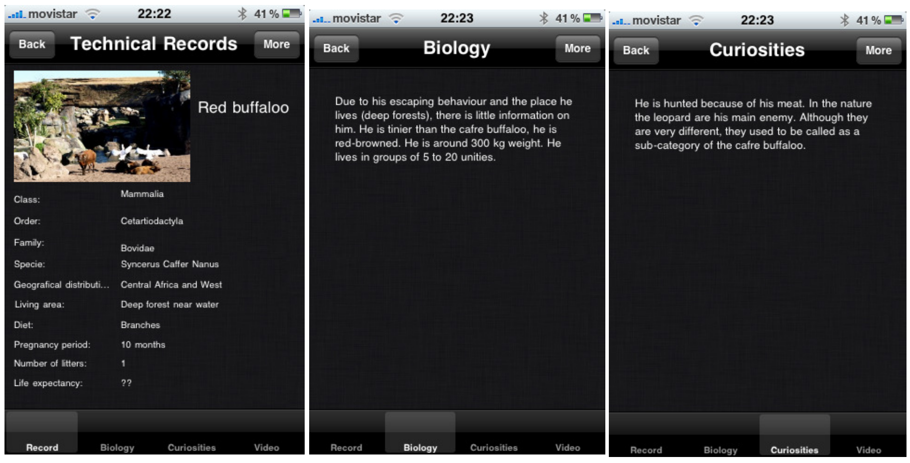


Figure 12. List of animals, plants and landscapes, for Equatorial Forest and English language.

When the user taps on the animal, the technical record, biology, or curiosities interfaces are shown. This can be seen in figure 13.



Red buffaloo

Class:	Mammalia
Order:	Cetartiodactyla
Family:	Bovidae
Specie:	Syncerus Caffer Nanus
Geografical distributi…	Central Africa and West
Living area:	Deep forest near water
Diet:	Branches
Pregnancy period:	10 months
Number of litters:	1
Life expectancy:	??

Due to his escaping behaviour and the place he lives (deep forests), there is little information on him. He is tinier than the cafre buffaloo, he is red-browned. He is around 300 kg weight. He lives in groups of 5 to 20 unities.

He is hunted because of his meat. In the nature the leopard are his main enemy. Although they are very different, they used to be called as a sub-category of the cafre buffaloo.

Figure 13: Technical record, biology and curiosities views, for Red Buffaloo, English and child level of knowledge.

The information shown in these three views vary depending on the language and the level of knowledge selected, and it comes from the sqlite3 database.

When the user selects the Video option, a list of videos, their titles and brief descriptions is displayed. This can be seen in figure 14. The videos displayed also depend on the language and level of knowledge selected. This way, an English child can watch videos suited to his profile.
The user can choose one of them to start watching it. The result can be seen in figure 15.

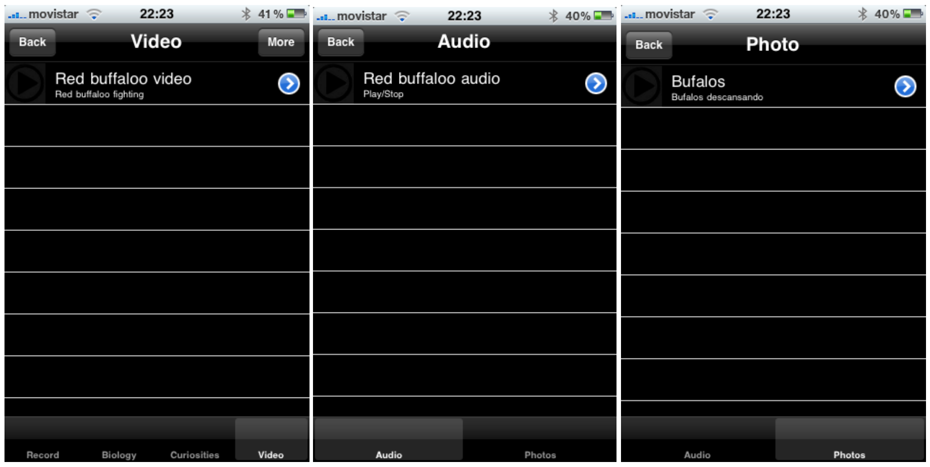


Figure 14. List of videos, audio and photos, for English language and Child level of knowledge.

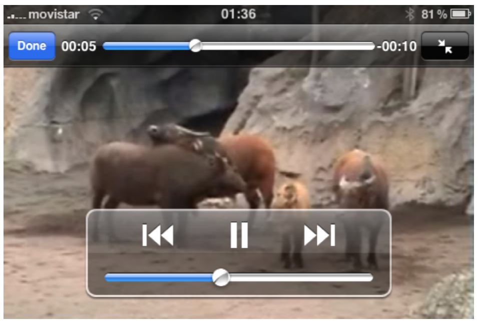


Figure 15. Quicktime interface appearance for Buffaloo video.

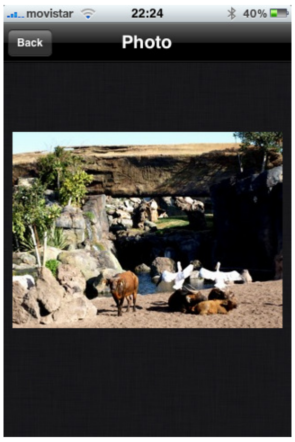


Figure 16. Photo show interface.

When the user selects the Audio option, a list of audios, their titles and a Play/Stop message is displayed. This can be seen in figure 14. The audios

displayed also depend on the language and level of knowledge selected. This way, an English child can listen to audios suited to his profile.
The user can start listening to it by tapping on the audio item. If he taps on it again, the audio stops. These two actions can be repeated the number of times the user desires.
When the user chooses the Photos option, a list of photos, their titles and a brief description is displayed. This can be seen in figure 14. If the user taps on a photo, it is shown. See figure 16.
As it was pointed out above, all this information is stored locally in a sqlite3 database.

3.2.2 Remote mode

Users have the option to enjoy consulting completely up to date information, by connecting to a remote database server.
In order to select this option, the user must choose the WIFI ON option in the first menu.
All the information and interfaces have the same layout as the ones shown in the local module. The difference is that this information is obtained from an SQL Server 2005 database server.

3.2.3 GPS location and google maps

This alternative is available for users by selecting the GPS option in the interface shown in figure 17.

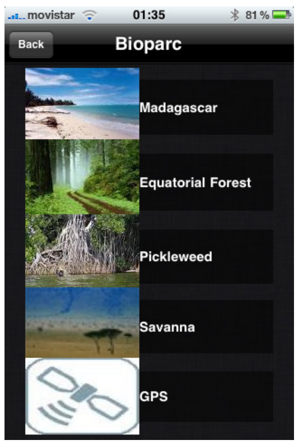


Figure 17. GPS option.

a) First of all, a **GPS searcher** can be used to locate animals in a map and discover the distance (in metres) from the user location to the animal. The interface shown to the user can be seen in figure 18.

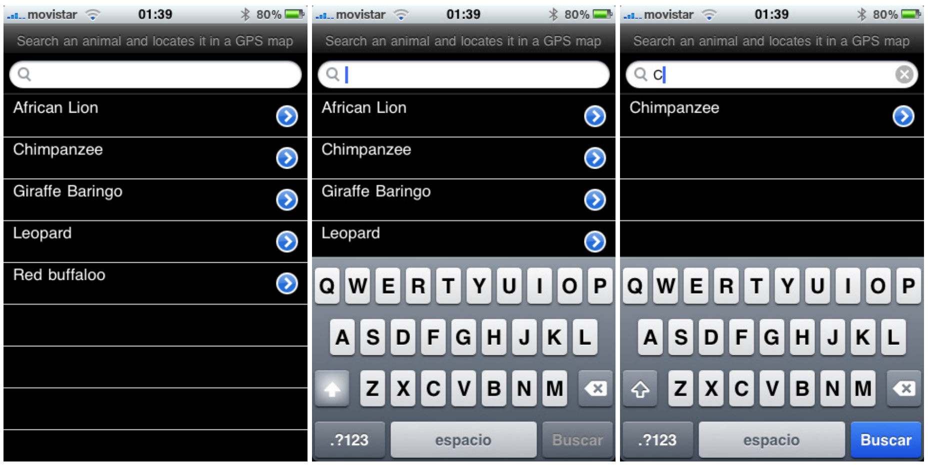
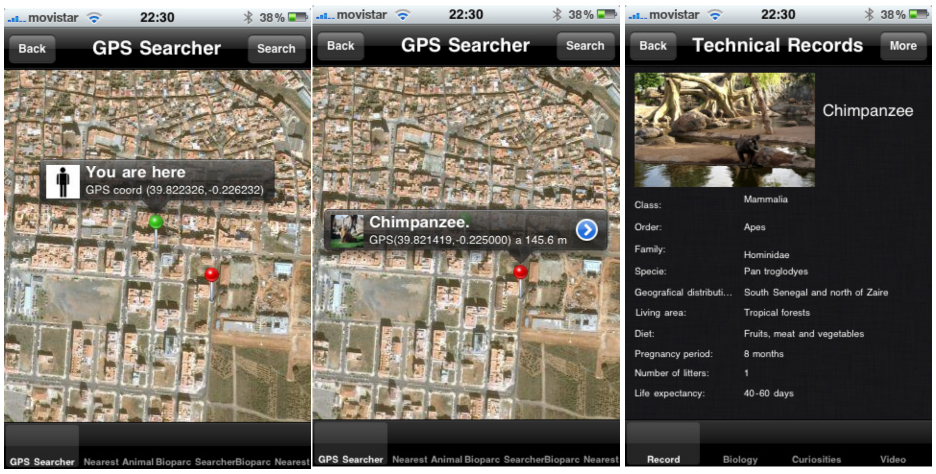


Figure 18. GPS Searcher interface.

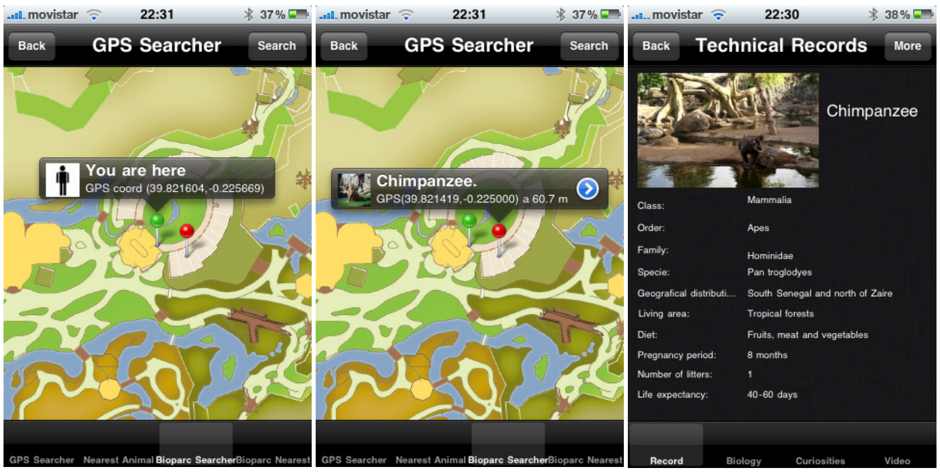
A list of animals, a contextual keyboard and a search field appear in the view. The user can directly tap on the animal to select it. Additionally he can tap on the search field and introduce the animal he wants to look for using the keyboard. As he introduces characters, the list is modified (filtered) appearing just those animals that begin with the letters he has introduced.
The animals shown are stored in the database, so that the user can access them in this utility. The database structure will further be explained later.
Once the animal is selected, a new map view appears. It can be seen in figure 19.



Chimpanzee

Class:	Mammalia
Order:	Apes
Family:	Hominidae
Specie:	Pan troglodyes
Geografical distributi...	South Senegal and north of Zaire
Living area:	Tropical forests
Diet:	Fruits, meat and vegetables
Pregnancy period:	8 months
Number of litters:	1
Life expectancy:	40-60 days

Figure 19. Search map interface and animal view.



Class:	Mammalia
Order:	Apes
Family:	Hominidae
Specie:	Pan troglodyes
Geografical distributi...	South Senegal and north of Zaire
Living area:	Tropical forests
Diet:	Fruits, meat and vegetables
Pregnancy period:	8 months
Number of litters:	1
Life expectancy:	40-60 days

Figure 20. Search custom map interface and animal view.

In this interface a satellite map centered in the area where the user is positioned and two pins appear. The map allows scroll and zoom utilities. The green pin shows the user location, while the red one shows the animal's location.
The user can tap on the green pin, then it appears a bubble showing a user icon and a message indicating that this is the user's position.
Besides the user can tap on the red pin, then it appears another bubble indicating the name of the animal and the distance in metres between the user and the animal.
When the user taps on the arrow button, the technical records view about this animal appears, allowing the user to view this technical record, the biology, curiosities, videos, audio, photos, and so on. The search button selection, leads the user to the animal selection interface seen in Figure 18.
A custom map interface can be seen within this utility. Instead of displaying a google map, it uses a park custom map. Once more, two pins appear. The green one indicates the user's location and the red one points out the animal location. See figure 20. The functionality is exactly the same as that in the google map option. By tapping the search button, interfaces in figure 18 appear, allowing animal search.
The behavior of the utility explained is achieved by including in the local database several tables that will store GPS points (basically their latitude and longitude coordinates). These GPS points are taken around each of the areas where the animals are. Each area consists of several animals, and it is composed of several GPS points taken around it. So each animal can be viewed from different GPS points and a GPS point corresponds to several animals. A GPS point belongs to an area and an area consists of several GPS points.

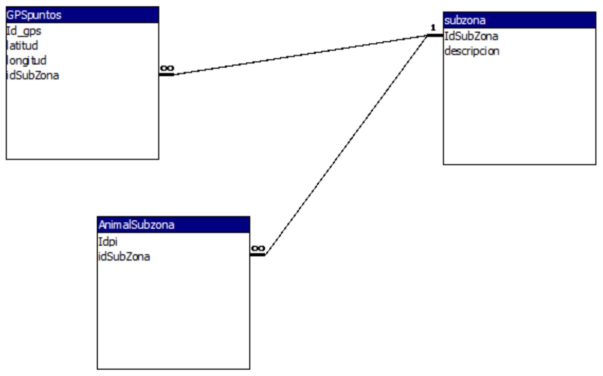


Figure 21 Tables corresponding to the GPS utility.

The application reads the GPS points from the database that corresponds to the animal the user wants to look for and calculates the minimum distance from the user's device to each of these points. When the algorithm has finished calculating the minimum distance and the position of this GPS point, tells the map to locate a pin in this position, showing the name of the animal and the distance in metres.
The following schema shows how GPS points are taken around each of the areas.

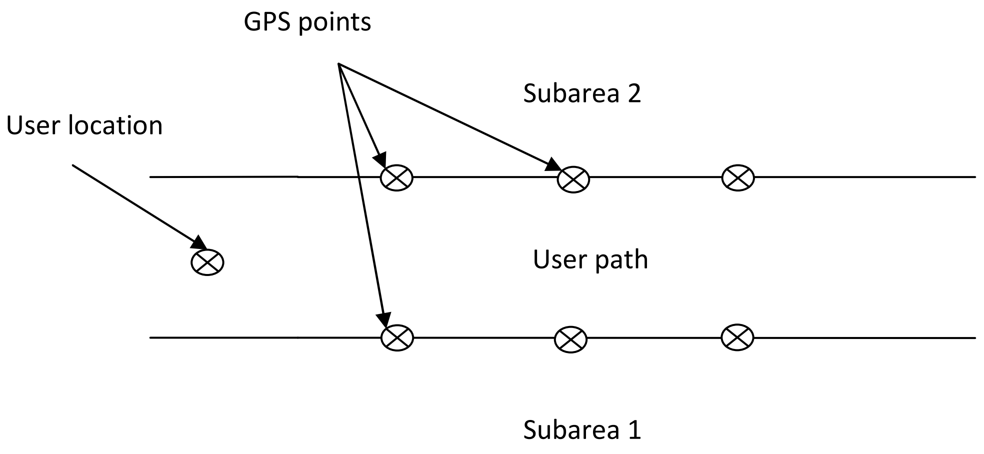


Figure 22. GPS points around each subarea.

b) A **nearest animal** utility has been developed to tell the user which is the closest animal and its distance from it. The interfaces can be seen in 23 and figure 24.
Figure 23 shows google map interfaces, whereas figure 24 shows the park's custom map interfaces. Their functionality is exactly the same. Instead of displaying a google map, it displays a custom map obtained from the park.

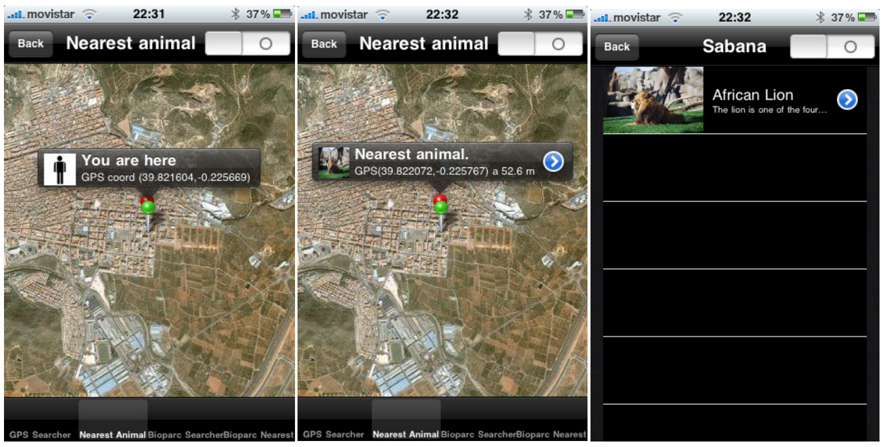

movistar 22:31 37%
Back Nearest animal
You are here
GPS coord (39.821604,-0.225669)
movistar 22:32 37%
Back Nearest animal
Nearest animal.
GPS(39.822072,-0.225767) a 52.6 m
GPS Searcher Nearest Animal Bioparc Searcher Bioparc Nearest
movistar 22:32 37%
Back Sabana
African Lion
The lion is one of the four...

Figure 23. Nearest animal utility interfaces.

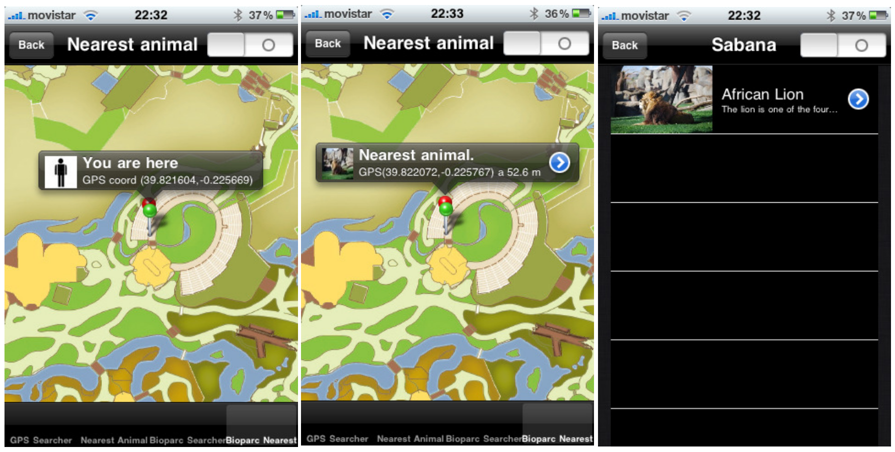

movistar 22:32 37%
Back Nearest animal
You are here
GPS coord (39.821604,-0.225669)
movistar 22:33 36%
Back Nearest animal
Nearest animal.
GPS(39.822072,-0.225767) a 52.6 m
GPS Searcher Nearest Animal Bioparc Searcher Bioparc Nearest
movistar 22:32 37%
Back Sabana
African Lion
The lion is one of the four...

Figure 24. Nearest animal custom map utility interfaces.

This module automatically displays a map centered in the user location and two pins. The green one shows the user position, while the red one shows the nearest animal position.
When the user taps on the green pin a bubble with the user icon and a message indicating that this is the user's position is shown. If the user taps on the red pin a "Nearest animal" message and the distance (in metres) from the user to the closest animal is displayed. When tapping on the arrow button, a list including the set of animals that belong to the nearest area is displayed.
All this is achieved by reading the GPS points from the database and calculating the minimum distance from the user to each of the points. When the minimum distance is calculated, the corresponding pin is shown, the distance is displayed, and the list of animals that belong to the area defined by this minimum-distance GPS point is shown when tapping on the arrow button.

The algorithms developed for these two utilities are also applied in the context-aware module.

A complete list of interfaces can be seen in Annex 2.

3.2.4 Context-aware mode

This module's goal is to implement a context-aware utility for the application. The user should be able to wander around the park and the application, when he is near enough to an area, should display the list of animals he is viewing at this particular moment without user intervention.
This means that the user will see in his device exactly the same animals he is physically looking at. When he moves or changes orientation, the list of animals is updated, that is, if he is looking at another area, the application will automatically change the list of animals to the new one.

Context-aware utility within the Nearest animal map. It includes a switch above. When the switch is enabled the context-aware option is available, whereas if the switch is disabled the nearest animal utility is available.
Figure 25 shows the nearest animal map, with orientation switch enabled. The interface in the left corresponds to the google's map, whereas the interface in the right corresponds to the park custom map.

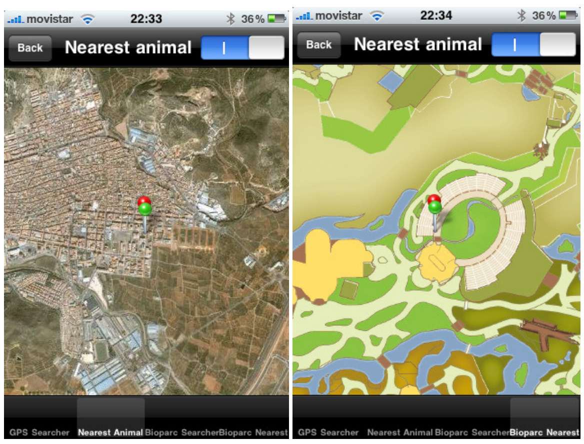
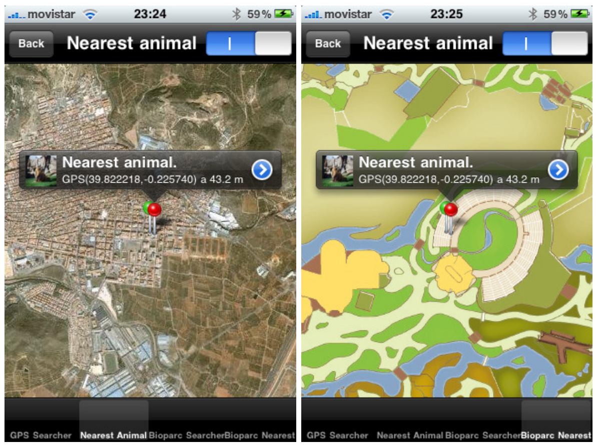


Figure 25. Nearest animal map, orientation enabled.



Nearest animal.
GPS(39.822218,-0.225740) a 43.2 m

Nearest animal.
GPS(39.822218,-0.225740) a 43.2 m

Figure 26. Nearest animal message.

When the user taps on the red button, a bubble appears indicating the nearest animals and the distance to them. This can be seen in Figure 26. If the user then taps on the arrow button, a list containing the animals corresponding to the user's orientation is displayed. In figure 27, context-aware results are shown, that is, a list containing the African Lion when the user is physically looking at this animal is displayed, and a list containing nothing is displayed when the user isn't looking at any animal. Once more, we could disable the orientation switch. If we do so, the context-aware mode is disabled and the nearest animal list appears.

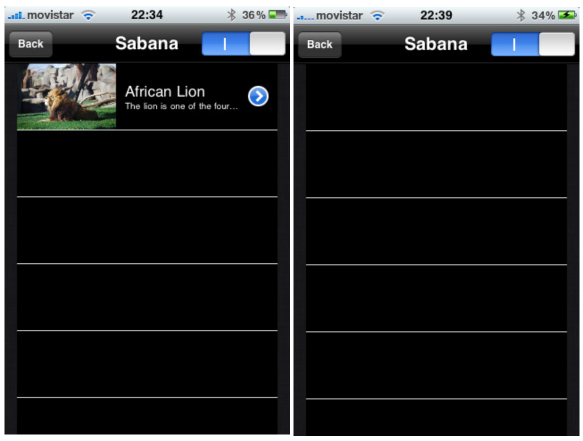


Figure 27. Context-aware results (viewing an area containing an African Lion and another area containing nothing).

This is the most complex module in the sense that two measures have to be combined and taken into account, that is, GPS locations and compass readings, and several database searches have to be carried out. One of these measures is the user's location and the other one is the compass reading.
The application has to cope with a serious problem, as the devices that support compass readings are only iPhone 3GS and iPhone 4 at the moment.
In order to solve this problem, the compass measurement part could be disabled by the user that owns a device previous than iPhone 3GS using the switch that is displayed in the corresponding list of animals interface.

Two different approaches have been proposed to implement the context-aware functionality, each one providing its advantages and drawbacks.

The **first approach** differentiates the scenario where the user is immersed in two possible types of zones: "two-area" zones and "one-area" zones. Each area consists of different animals. "Two-area" zones have animals at both sides of the path, that is, it is composed of two areas, whereas "one-area" zones have animals at just one of the sides of the path, that is, it is composed of just one area. They can be seen in figures 28 and 29.
This approach deals in a different way those zones where there are two areas and those with just one area.

For the “two-area” zones we make the assumption that the GPS points are taken from the border of both sides of the path. They are placed at both sides in such a way that if we draw a line from one point marking subarea 1 to another marking subarea 2 the lines form 90 degrees from the axis of the path. See figure 28.
For the “one-area” zones we assume that the GPS points are taken from just one of the sides of the path. See figure 29.
The first thing we have to discover is if the zone has two areas or just one.
Then, the angles of vision have to be calculated.
Once the angles of vision are known, the compass reading (an angle) must be compared to the angles of vision to discover which area the user is viewing.
The calculation of the angles of vision is carried out in a different way for “two-area” zones and “one-area” zones.
Two algorithms are proposed within this first approach.
The first algorithm discovers whether the user is near enough to an area of animals or not, then it finds out if there are two areas at both sides of the path or just one.
This algorithm can be seen in Algorithm 2, and visual clarification in figures 28 and 29.

Algorithm 2. Find out if there are two areas at both sides of the path or just one

```
For every GPS point in the database:
        Calculate the distance from the user to it
        If its distance is the minimum distance → new minimum distance
                                                  new minimum point
d1 is the minimum distance
 p1=lat1,lon1 are this point’s coordinates
sub1 is the area corresponding to lat1,lon1.
For every GPS point in the database except those from sub1:
        Calculate the distance from the user to it.
        If this distance is the minimum → new minimum distance
                                          New minimum point
d2 is the minimum distance to another area
p2=lat2,lon2 are this point’s coordinates
sub2 is the area corresponding to lat2,lon2.
If (d1<35)   #near enough to an area, 35 in metres
Then
        d12 = distancia p1 a p2
        If (d12<15)   #there are two areas, 15 in metres
                Two areas (sub1,sub2)
        Else
                One area (sub1)
                For every GPS point in the database (except lat1,lon1)
                Calculate the distance from the user to it.
                        If its distance is the minimum distance
                                → new minimum distance
                            new minimum point
                        d3 is the minimum distance to other point in the same area
                        p3=lat3,lon3 are the coordinates of this point
                        sub1 is the area corresponding to lat3,lon3.

        End
End
```

The algorithm searches for the minimum distance (d1) point from the user position (p0) to each of the database GPS points and gets this area's number (sub1) and its location (p1). See figure 28. Then it looks for the minimum distance (d2) point from the user position p0 to each of the database GPS points without including points from sub1, and gets this area's number (sub2) and its location (p2). If the closest area (d1 or d2) is near enough, say 35 metres, we are near an area with animals, then it calculates the distance from p1 to p2 (d12); this is the distance between p1 and p2 (at both sides of the path). If this distance is less than 15 metres (the width of the path is always less than 15 metres), we can assure that there are two areas at both sides of the path. Otherwise, there is just one area. In this case, we calculate the second minimum distance point from the user (p0) to this area (p3). See figure 29.
After finishing this algorithm, we know if there are two areas or just one, the position of the minimum distance points at both sides (p1 and p2) or the position of the minimum distnance point in a single side (p1 and p3), and the areas' numbers (sub1 and sub2).
Next, we must calculate the angles of vision to both areas (in "two-area" zones) and to the single area (in "one-area" zones). See Figure 30 and 31.

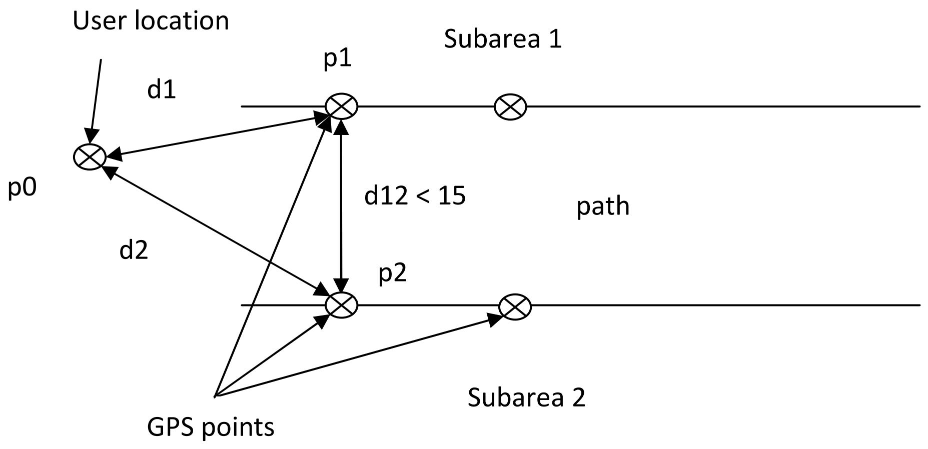


Figure 28. Two areas near the user's location.

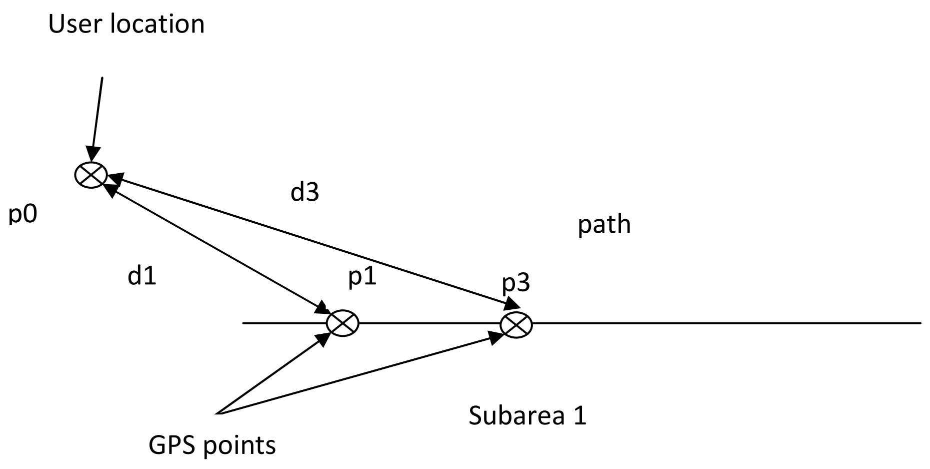


Figure 29. One area near the user's location.

The next step is to discover the area the user is viewing, which is done by a second algorithm that finds out the angle of vision to the areas in both cases. Then, knowing the user's orientation degrees (from the compass), it discovers if the user is viewing one area or another.
This second algorithm can be seen in Algorithm 3, and visual clarification in figures 30 and 31.
It simply calculates the angles of vision *alpha1* and *alpha2*, which point to both areas. Then, it compares the user orientation (provided by the compass reading) with these two angles, and discovers the area he is viewing. If there is just one area, it calculates the angle of vision *alpha1*, which points to the single area. Then, it compares the user orientation (once more provided by the compass reading) with this angle, and discovers whether he is viewing this area or not.

.

Algorithm 3. Find out the angles of vision and the area the user is viewing

```
If there are two areas      #two areas known by algorithm 2
Then
        #Calculate the angle formed by drawing a line from p2 to p1
        dx = lat1-lat2    #lat1, lat2, lon1,lon2 know from Algorithm 2
        dy = lon1-lon2
        If (dx==0)
        Then
                If (dy>0)
                Then  angle12=90
                Else  angle12=270
        Else
                angle12 =atan(dy/dx)*180/pi
        End
        If (dx<0)   Then             #for other two quadrants
                angle12 = angle12 +180
        End
        If  (angle12<0)  Then        # for negative angles
                angle12 = angle12 +360
        End
        alpha1 = angle12             #angle of vision of sub1
        alpha2 = angle12 +180        #angle of vision of sub2
Else    #just one area
        #calculate the angle formed by drawing a line from p3 to p1
        dx = lat1-lat3      #lat1,lon1,lat3,lon3 known from Algorithm 2
        dy = lon1-lon3
        If (dx==0)
        Then
                If (dy>0)
                Then  angle13=90
                Else  angle13=270
        Else
                angle13 =atan(dy/dx)*180/pi
        End
        If (dx<0)   Then             #for other two quadrants
                angle13 = angle13 +180
        End
        If  (angle13<0) Then         # for negative angles
                angle13 = angle13 +360
        End
        alpha1 = angle13 – 90   #angle of vision of sub1
End
#alphausuario is the reading from the compass in degrees
# +/- 60 degrees of margin.
If ((alphausuario >=(alpha1-60)) and (alphausuario<=(alpha1+60)))   Then
        #between -60 and +60 degrees
        Subareaviewed = sub1
Else If ((alphausuario>=(alpha2-60)) and (alphausuario<=(alpha2+60))) Then
        #between -60 and +60 degrees
        Subareaviewed = sub2
Else
        Subareaviewed = 0 #the user is not viewing any area
End
```

Algorithm 4 finds out if the user is viewing one area or another, even if he is not viewing any area; in this last case the algorithm's response will be that the user is viewing area 0. The result is given in a variable called *subareaviewed*.
First of all, if there are two areas, the algorithm calculates the angle of vision of one area and another (*alpha1*, *alpha2*). See Figure 30. The angle between p1 and p2 (*angle12*) is calculated. Then *alpha1* is this *angle12*, while *alpha2* is this *angle12* plus 180 degrees.
Afterwards, a comparison is made to determine if the user is viewing subarea 1 or subarea 2. This comparison takes into account the angle read by the compass (*alphausuario)* and includes an angle deviation between -60 and +60 from the angles of both areas (*alpha1* and *alpha2*). The result is the area the user is

viewing. Figure 30 may be consulted to clarify the comprehension of this algorithm.
If there is just one area, the algorithm calculates the angle between p3 and p1 (*angle13*). See Figure 31. The angle of vision (*alpha1*) is this angle minus 90 degrees. Then we must compare the angle read by the compass (*alphausuario*) and *alpha1*, and include a deviation angle of -60 and +60 from the angle obtained *alpha1*. Figure 31 could clarify this explanation.
The -60 and +60 deviation means that if the user orientation angle (*alphausuario*) is between *alpha1*-60 and *alpha1*+60 the algorithm gives us the result of viewing area *sub1*.

Furthermore, code must be written to execute these two algorithms each time the user moves or changes its orientation. *iPhone SDK*, and more in particular, the *CoreLocation* framework has methods that allow us to call these algorithms each time the user moves a certain distance, say 5 metres, or the compass reading has changed an angle, say 45 degrees. Adjusting the distance and the angle, we can save power or get an accurate response of the system. Due to the features of this application and the great accuracy that the system has to offer, it is preferred to adjust little distance movements (5 metres) and orientation changes (45 degrees) to obtain high accuracy and fast response, although incurring in a higher power consumption. This issue will be studied deeply in chapter 4, where power consumption and time response are evaluated by experimentation.
The main advantage of this approach is that it just needs GPS points taken from a single side of the path for “one-area” case. This means that the number of GPS points will be lower in comparison to other approaches, saving memory in the database, lightening the effort in taking the GPS points throughout the park, and the effort in introducing the GPS points in the database.
The most important drawback of this approach is that it is not 100% reliable, that is, in some occasions “one-area” cases could give a *subarea* erroneous result, as the angle of vision calculation depends on the user’s position in “one-area” cases, and the user GPS position is sometimes not 100% accurate as the iPhone GPS chip gives the GPS location with a margin of error. What is more, it is more difficult to be implemented than other approaches. So a second approach will be proposed, which is 100% accurate.

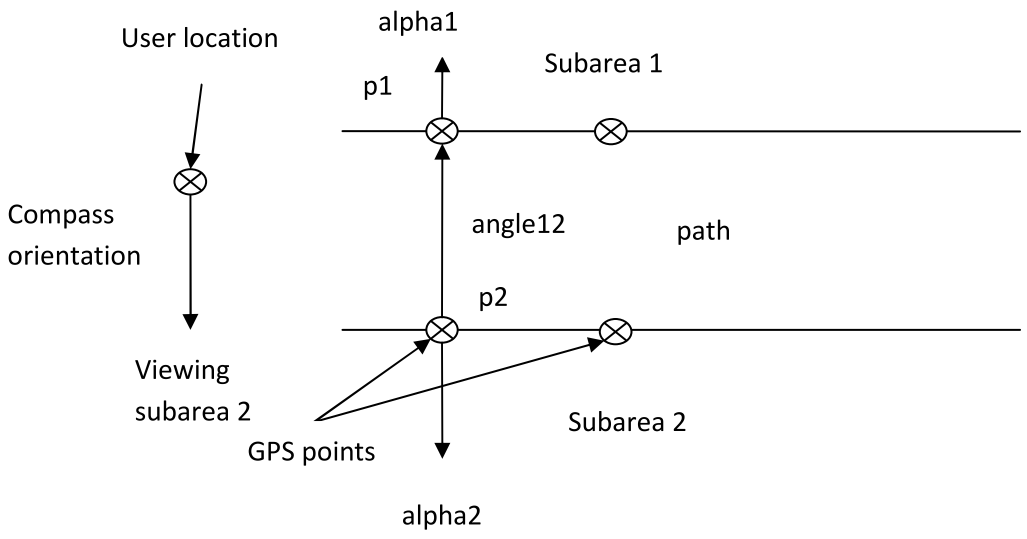


Figure 30. Angles of vision (two areas).

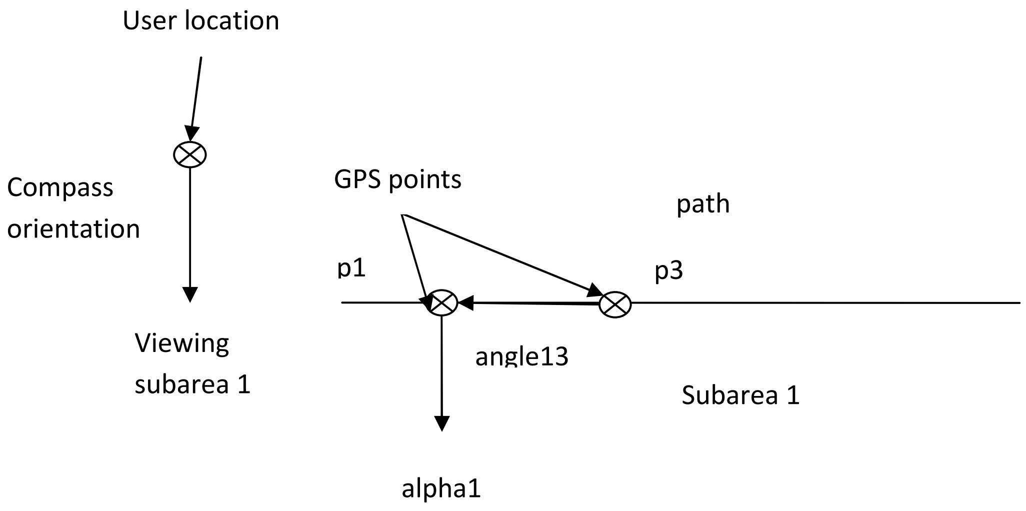


Figure 31. Angles of vision (one area).

Before introducing the second approach, another way to calculate the angles of vision for "two-area" zones can be proposed, although it is slower and more difficult to be implemented. It offers exactly the same results as the one that has been explained before.
Figure 30 shows the way the angles of vision are calculated for "two-area" by this method.

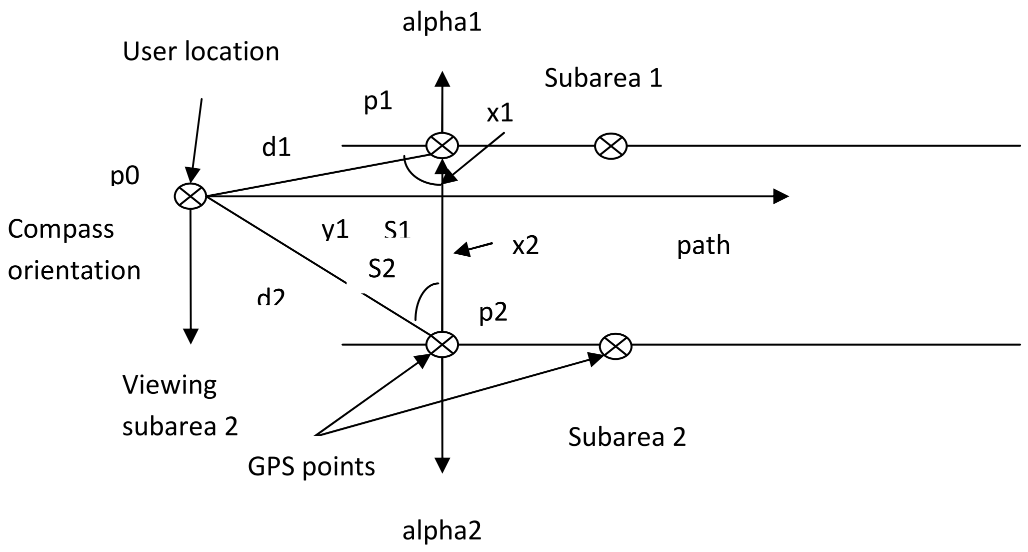


Figure 32. Another way to calculate angles of vision.

From figure 32, we can easily obtain the following equations:

$$\cos(s2) = \frac{x2}{d2} \quad [1]$$
$$\cos(s1) = \frac{x1}{d1} \quad [2]$$
$$\sin(s1) = \frac{y1}{d1} \quad [3]$$
$$\sin(s2) = \frac{y1}{d2} \quad [4]$$

From [3] and [4], we can deduce this one:

$$\frac{d2}{d1} = \frac{\sin(s1)}{\sin(s2)} \quad [5]$$

What is more, from figure 32, [2] and [1], we can easily obtain this equation.

$$x = \mathrm{x1} + \mathrm{x2} = \mathrm{d1}\cos(s1) + \mathrm{d2}\cos(s2) \quad [6]$$

From figure 32, we can obtain the following equations,

$$x1^2 + y1^2 = d1^2$$
$$x2^2 + y1^2 = d2^2$$

$$x1 = d1\sqrt{1 - (\sin(\mathrm{s1}))^2} \quad [7]$$
$$x2 = d2\sqrt{1 - (\sin(\mathrm{s2}))^2} \quad [8]$$

Using [5], [6], [7] and [8]

$$x = \mathrm{d1}\cos(s1) + d2\sqrt{1 - \frac{d1^2}{d2^2}(\sin(s1))^2} \quad [9]$$

From [5] and [9], we obtain the following equations, that allows us to obtain the angles s1 and s2

$$\frac{x}{d1} = \cos(s1) + \sqrt{(\frac{d2}{d1})^2 - (\sin(s1))^2} \quad [10]$$

$$s2 = \sin^{-1}(\frac{d1}{d2}\sin(s1)) \quad [11]$$

$$x = distance\ (p1, p2) \quad [12]$$

So our algorithm should calculate x, the distance from p1 and p2. Then, using a numerical algorithm, calculate the angle s1 using [10]. Finally, the angle s2 can be calculated using [11], the angles of vision alpha1 and alpha2 can be calculated by the following equations.

$$alpha1 = angle(p0, p1) + s1$$
$$alpha2 = angle(p0, p2) + s2$$

Due to its high time response and the complexity of its calculation, this algorithm has been deprecated. Algorithm 3 gives us exactly the same angles as [10] and [11] but more easily and faster.

A **second approach** can be used to calculate the angles of vision for both “two-area” and “one-area” cases and the area the user is viewing.
This approach considers the same model for “two-area” zones as in the first approach. See Figure 30. For “one-area” zones, the model proposed is different. See Figure 33. A single area can be seen as having “something” at one side and “nothing” at the other. This “nothing” is marked using a 0 area at the other side.

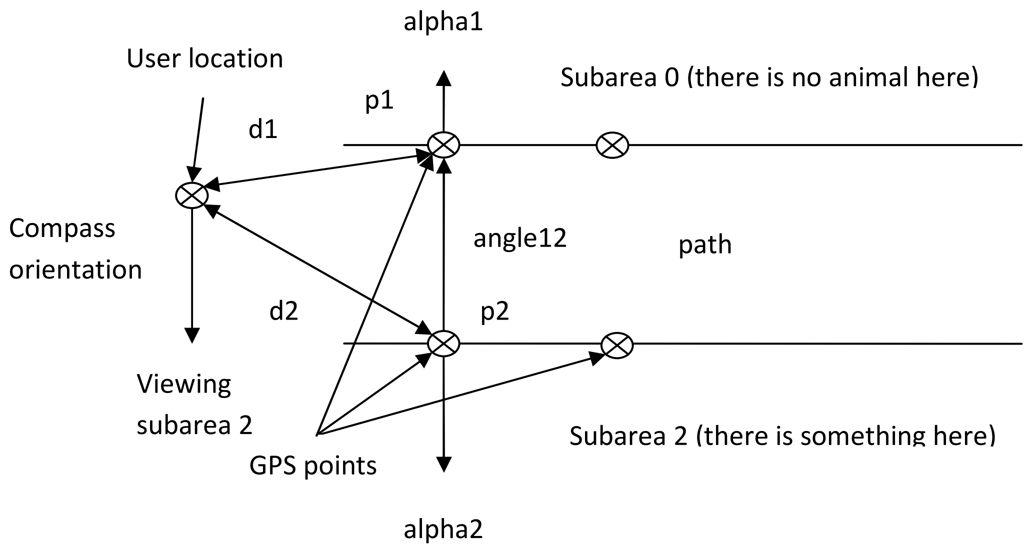


Figure 33. Another proposal to “one area” algorithm.

So our new algorithms use always a “two area” algorithm, particularizing those “one area” places with a subarea 0 indicating that there are no animals at that side.

First, we must find out the areas at both sides, that is, calculating the closest GPS points p1 and p2 (at both sides) from the user to all the GPS points in the database, and the area’s number at both sides. See Figure 30. Then we must find out the angles of vision alpha1 and alpha2 to both sides. Finally, a comparison has to be carried out between the user orientation (reading from the compass) and those angles of vision alpha1 and alpha2, resulting in the area the user is viewing.

For this purpose, two algorithms are presented.

The first one finds out the areas at both sides. See Algorithm 4.

Algorithm 4. Find out the areas at both sides.

```
For every GPS point in the database:
        Calculate the distance from the user to it
        If its distance is the minimum distance → new minimum distance
                                                  new minimum point
d1 is the minimum distance
 p1=lat1,lon1 are this point’s coordinates
sub1 is the area corresponding to lat1,lon1.
For every GPS point in the database except those from sub1:
        Calculate the distance from the user to it.
        If this distance is the minimum  → new minimum distance
                                           New minimum point
d2 is the minimum distance to another area
p2=lat2,lon2 are this point’s coordinates
sub2 is the area corresponding to lat2,lon2.
If (d1<35)   #near enough to an area, 35 in metres
Then
```

```
        Two areas (sub1,sub2)    #sub1 and sub2 will tell us which is the area and the 0 area
End                              #as this was introduced in the database
```

The minimum distance point *p1* is found out by calculating the distance between the user and all the GPS points in the database, and the area number *sub1* corresponding to this point *p1*. Then, the same process is carried out without including those points from *sub1*. The result is *p2* and *sub2*. If the minimum distance *d1* is lower than 35 metres, this means that there is at least one area near the user.

The second step is to find out the angles of vision to both areas, that is, alpha1 and alpha2, and comparing the user orientation (reading from the compass) to these angles of vision we know the area the user is viewing. See Algorithm 5.

Algorithm 5. Find out the angles of vision and the area the user is viewing

```
#Calculate the angle formed by drawing a line from p2 to p1
dx = lat1-lat2     #lat1, lat2, lon1,lon2 know from Algorithm 2
dy = lon1-lon2
If (dx==0)
Then
        If (dy>0)
        Then  angle12=90
        Else  angle12=270
Else
        angle12 =atan(dy/dx)*180/pi
End
If (dx<0)   Then              #for other two quadrants
        angle12 = angle12 +180
End
If  (angle12<0)  Then         # for negative angles
        angle12 = angle12 +360
End
alpha1 = angle12              #angle of vision of sub1
alpha2 = angle12 +180         #angle of vision of sub2

#alphausuario is the reading from the compass in degrees
# +/- 60 degrees of margin.
If ((alphausuario >=(alpha1-60)) and (alphausuario<=(alpha1+60)))   Then
        #between -60 and +60 degrees
        Subareaviewed = sub1   #sub1 contains area's number or 0
Else If ((alphausuario>=(alpha2-60)) and (alphausuario<=(alpha2+60))) Then
        #between -60 and +60 degrees
        Subareaviewed = sub2  #sub2 contains area's number or 0
Else
        Subareaviewed = 0 #the user is not viewing any area
End
```

The algorithm calculates the angle between *p1* and *p2*, that is, *angle12.* Then the angle of vision *alpha1* is *angle12*, whereas *alpha2* is this angle plus 180 degrees. Finally, a comparison between the user's orientation (*alphausuario*) and those angles of vision is been carried out to find out the area the user is viewing. Once more, a deviation angle between -60 and 60 is introduced in this comparison. This means that if the user orientation (*alphausuario*) is between *alpha1*-60 and *alpha1*+60 then he is viewing area *sub1*.

Furthermore, similar code as in the first approach must be written to execute these two algorithms each time the user moves or changes its orientation. *iPhone SDK*, and more in particular, the *CoreLocation* framework has methods that allow us to call these algorithms each time the user moves a certain distance, say 5 metres, or the compass reading has changed an angle, say 45 degrees. Adjusting the distance and the angle, we can save power or get an accurate response of the system. Due to the features of this application and the great accuracy that the system has to offer, it is preferred to adjust little distance movements (5 metres) and orientation changes (45 degrees) to obtain high accuracy and fast response, although incurring in a higher power consumption. This issue will be studied deeply in chapter 4, where power consumption and time response are evaluated by experimentation.

Due to first approach's lack of accuracy, this second approach has been introduced.
This approach's main advantage is that it is 100 % accurate while the other approach explained (corresponding to algorithms 2 and 3) depends on the GPS user location.
The user position is sometimes incorrectly obtained due to chip GPS lack of precision. So this second approach is more accurate and never fails. This second approach's main drawback is that more GPS points have to be considered making the process of obtaining and introducing them into the database more tedious. It requires more memory to store the GPS points.
These two proposals will be deeply evaluated and compared in chapter 4.

Note that we have assumed that there are linear paths. However, in the particular areas where the path draws a curve, these approaches and algorithms are also valid, that is, the approaches introduced above can also be used when the path is curved. The only thing we have to take into account in this case is that we must take more points and more closely. Then the stretch can be approximated to a linear path.

Finally, to show the animals belonging to the subarea that the user is viewing, a similar interface than that used in the "Nearest animal" module is displayed if the nearest animal distance is less than 35 metres. Otherwise, the interface changes automatically to the list of animals corresponding to the area calculated by the algorithms explained, allowing the context-aware mode usage.

Chapter 4

Performance and results

In this chapter, several experiments are going to be described, in order to evaluate the developed application's performance. In particular, correct functionality, time response and power consumption evaluation is the main goal of the experiments.

All the experiments have been carried out using about 60 GPS points stored in the database. As concerns to image size, 2.54x1.68 cm images with a resolution of 314 pixels / inch are considered in the first part of the experiments.

A Pentium IV at 2GHz has been used as a server.

First, the **system functionality in Bioparc** will be under study.

The **GPS Google Map utilities** time response is interesting to be taken into account. In particular, we are going to evaluate Nearest animal and Animal searcher modes' performance.

Nearest Animal mode time response is evaluated first.

The nearest animal mode periodically calls a method that calculates the minimum distance from the user to any of the GPS points taken. This method is called when the user moves about 5 metres, updating the user position and the nearest animal position, and the minimum distance too. Besides, it finds out the user orientation and the area he is looking at, depending on the orientation.

We are interested in evaluating the time the user has to wait for the application to calculate the nearest animals, the distance to them and their positions in the map, which also include the time invested in updating the position of the pins on the map. For that purpose, we measure the time the application takes to update the position in the map, carrying out several tests. The results obtained are shown in figure 34. 20 tests have been carried out to obtain the chart.

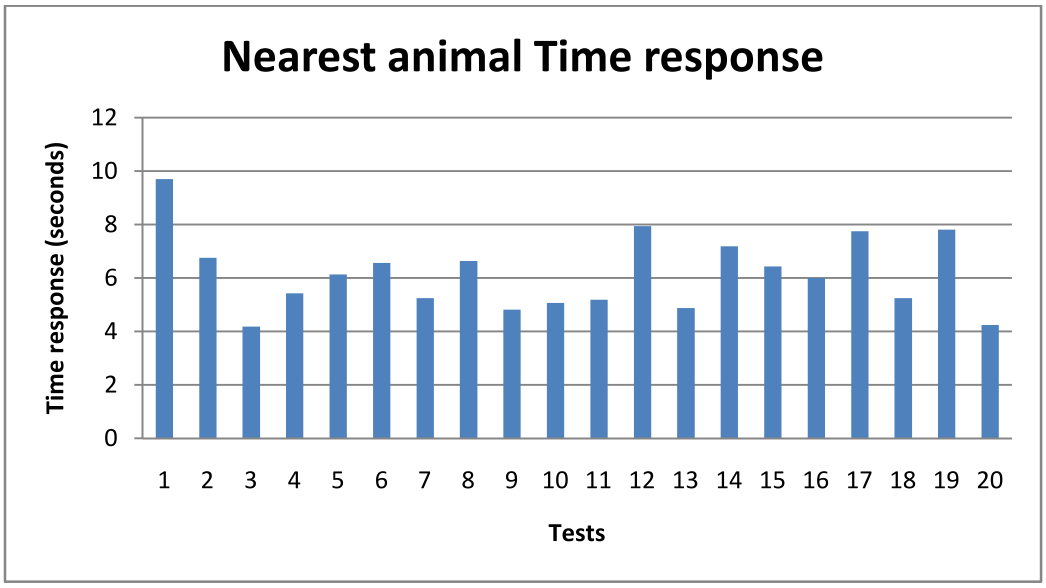


Figure 34. Nearest Animal Time response value for 20 tests. Update positions info each 5 metres, and using 3G connectivity.

The mean calculated for these data is 6.158 seconds and the standard deviation is 1.419.

From these results we observe that the user will have to wait for response about 6 seconds at an average, and in any case the user would have to wait for response no more than 10 seconds. This means that the user can be pleasantly walking throughout the park and the information will be updated in less than 10 seconds.

Next **Animal searcher** mode time response is evaluated.

The animal searcher mode periodically calls a method that calculates the minimum distance from the user to the GPS points taken corresponding to the animal. This method is called when the user moves about 5 metres, updating the user position, and the minimum distance between the user and the animal too.

We want to find out the time the user will have to wait to see the animal's position and his location update in the map. For this purpose, we measure the time between two updates. The results on this time response, which also include the time invested in updating the position of the pin on the map, are shown in figure 35. 20 tests have been carried out to obtain the chart.

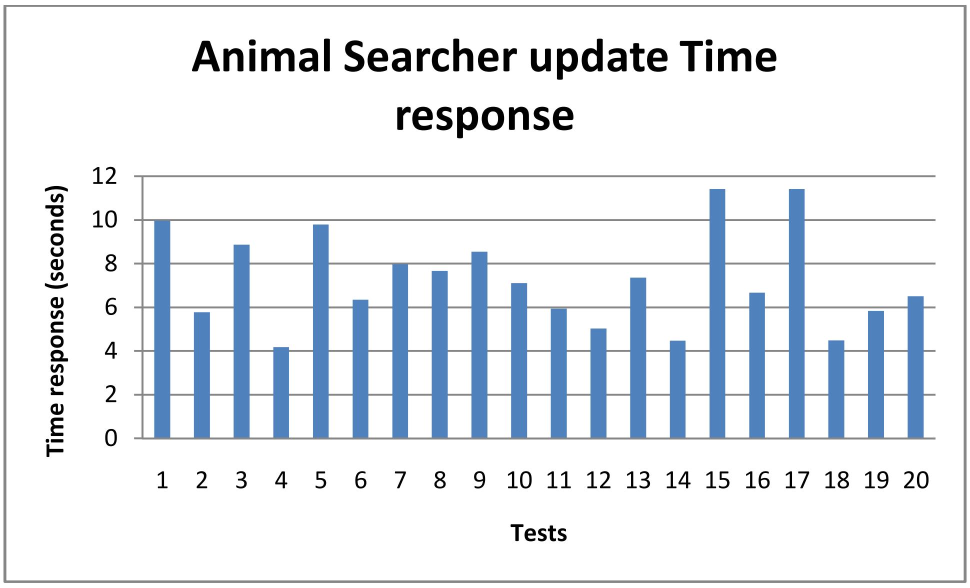


Figure 35. Animal Searcher updating time response value for 20 tests. Update position info each 5 metres, and using 3G connectivity.

The mean calculated for these data is 7.26 seconds and the standard deviation is 2.18. From these results we observe that the user will have to wait for response about 7 seconds at an average, and in any case the user would have to wait for response more than 11.5 seconds. This means that the user can be pleasantly walking throughout the park and the information will be updated in less than 11.5 seconds.

Next, we will evaluate the time invested after selecting the animal the user wants to look for and displaying the pin in the map with the distance to it. For this purpose, 10 tests have been carried out, selecting different animals and measuring the time the application takes to show the animal and user's pin in the map. The results can be seen in Figure 36.

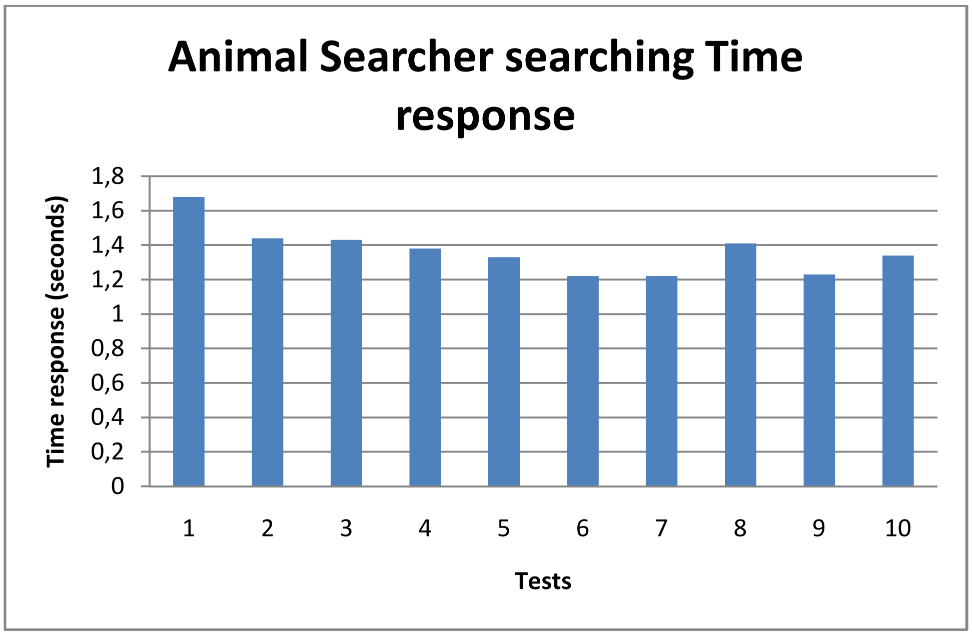


Figure 36. Animal searcher searching time response, invested in looking for the animal and displaying its location. 10 tests. Using 3G connectivity.

The mean calculated for these data is 1.36 seconds and the standard deviation is 0.14.

From these results we observe that the user will have to wait for the application to look for the animal and display its location about 1.36 seconds at an average, and in any case the user would have to wait for response more than 1.68 seconds. This means that the user can introduce the animal and wait for less than 2 seconds to the animal's location to appear.

Next **local vs. remote (in Bioparc) modes'** performance will be under study.

First of all, the **local data** mode time response is going to be evaluated.

We want to evaluate the time the user has to wait for the different views to appear after selecting the corresponding utility in local mode (without WIFI connection). For this purpose, we measure the time the view takes to appear in the different utilities. The results are shown in Figure 37.

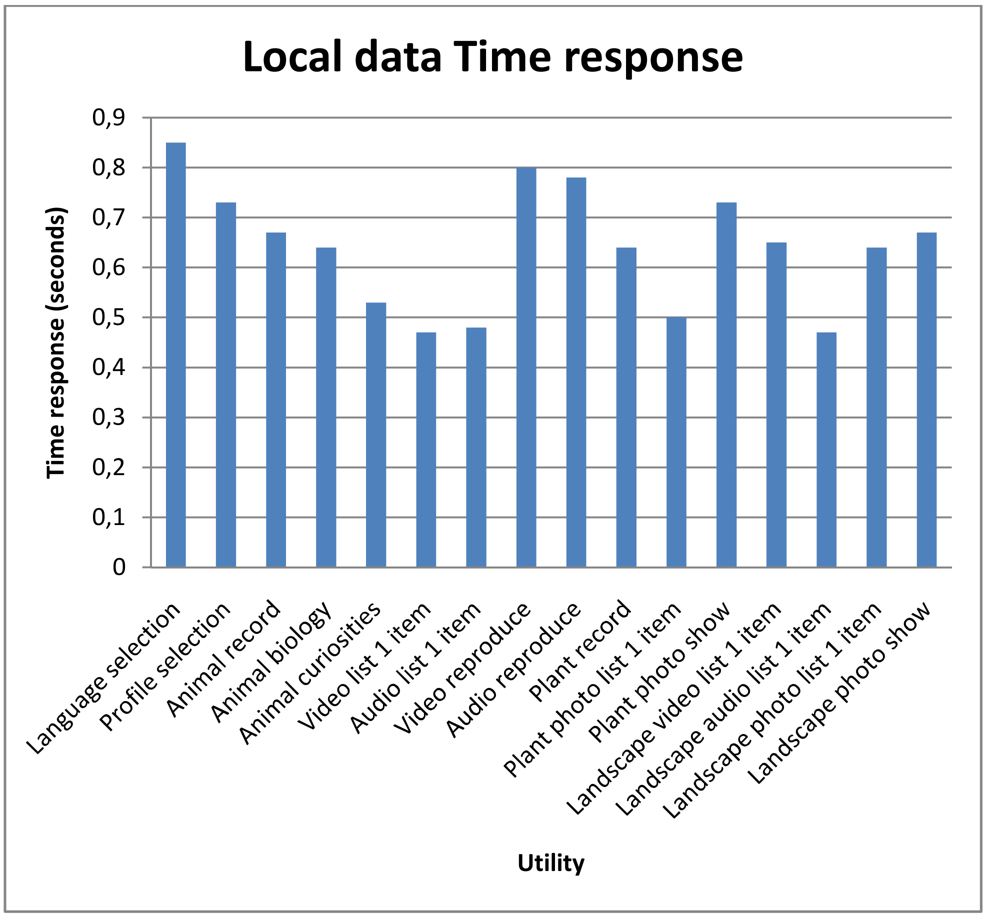


Figure 37. Time response for the different views in local mode.

From the chart, we can conclude that the user must not wait more than 1 second to view any of the information he is looking up. This information includes animal technical record, biology, curiosities, video, audio, plant technical record, photo view, and landscape information. What is more, the user has to wait less than 1 second after setting his preferences.

Next, we want to evaluate the application's time response in displaying lists of points of interest. For this purpose, 2-item animal, plant and landscape lists time response has been measured. The results can be seen in Figure 38.

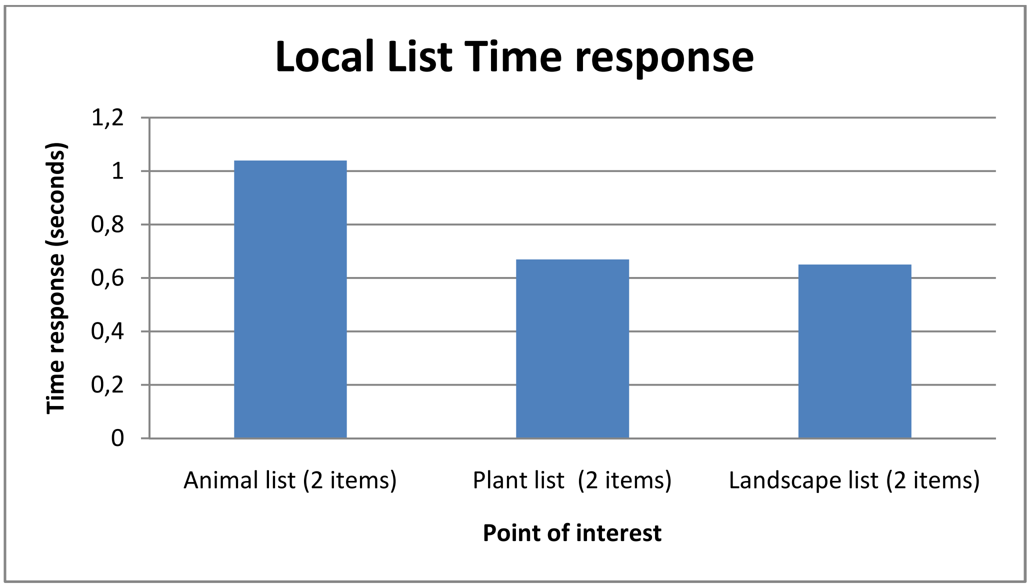


Figure 38. Time response for different lists of items (animal, plant and landscape) in local mode.

From the chart we can conclude that the plant and landscape lists take less to appear than the animal list, as there are more animals in the database than plants and landscapes, and the code has to deal with all the points of interest available in the database. Besides, the 2-item lists will take about 1 second or less to appear.

We are particularly interested in finding out the time response in displaying lists with different number of items. For this purpose, time response corresponding to different number of points of interest is measured, and the results are shown in the following figure.

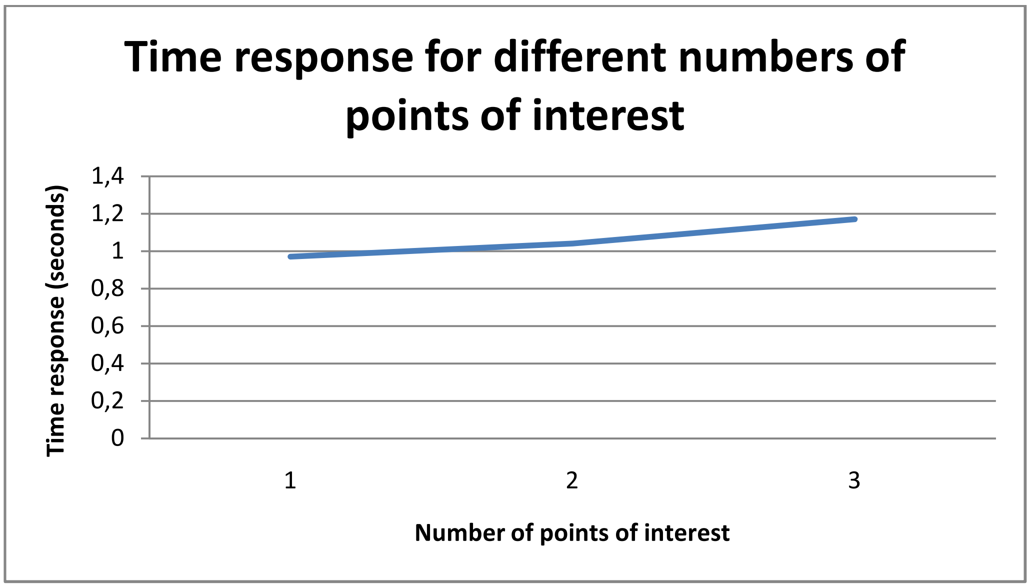


Figure 39. Time response for 1, 2 and 3 animals list in local mode.

A list containing just 1 animal, takes 0.97 seconds to appear. A list consisting of 3 animals, will take 1.17 seconds to appear. As a conclusion, if the number of animals increase, so does the time response.
The linear response encourages us to preview the time response for any number of animals in the list.
According to the data, if we model a straight line as:
Time(x) = 1.17 + (x-3) * 0.13
Time(30) = 4,68 seconds.
Time(12) = 2.34 seconds.
If we assume that the number of animals in an area is not higher than 30, we can conclude that the user will have to wait for the list to appear in any case less than 5 seconds, which is a reasonably good time response.
Moreover, if we assume that there is a maximum of 12 animals in an area of the park, the user will have to wait for the list to appear less than 2.5 seconds.

The **remote data** mode time response is going to be evaluated next. These experiments are carried out in Bioparc, at a low distance from the server.
The database server is Pentium IV at 2GHz, and it uses SQL Server 2005 technology.
In this mode's experiments the textual data is always stored in the remote database and accessed remotely, whereas audio, photo and video files are stored locally in the device. The textual data consists of textual information such as name, description, technical record, biology, curiosities, and so on, all the table identifiers such as id_foto, id_pi, and so on, and the audio, photo and video file names and url names. For all these experiments, audio and video files are stored locally in the device.

First, we would like to know the performance of the different utilities in remote mode. For this purpose, time response for the different utilities is evaluated calculating the time the application takes to show the information. The results can be seen in Figure 40.

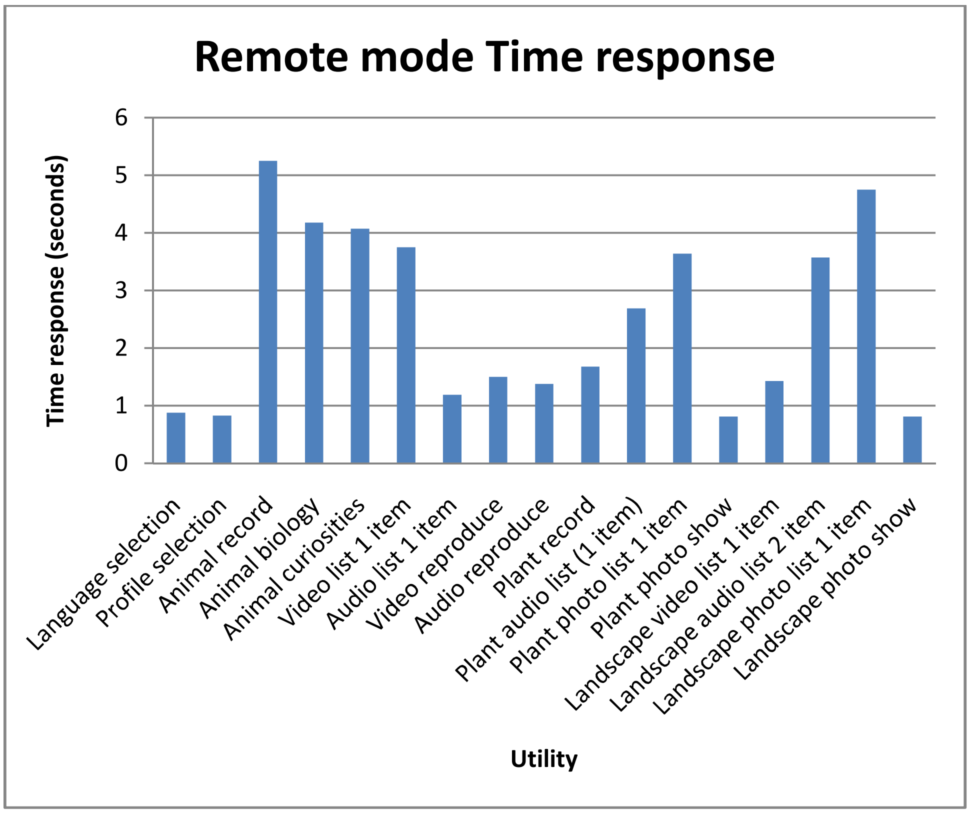


Figure 40. Time response for the different views in remote mode.

From this chart, we can conclude that the user will have to wait less than 5.5 seconds to view any of the information he is looking up. The information will take 2.5 seconds at an average and 1.55 of standard deviation to appear. The technical record has the highest time response with 5.25 seconds.

The following chart shows a comparison between local and remote modes, in their different utilities time response.

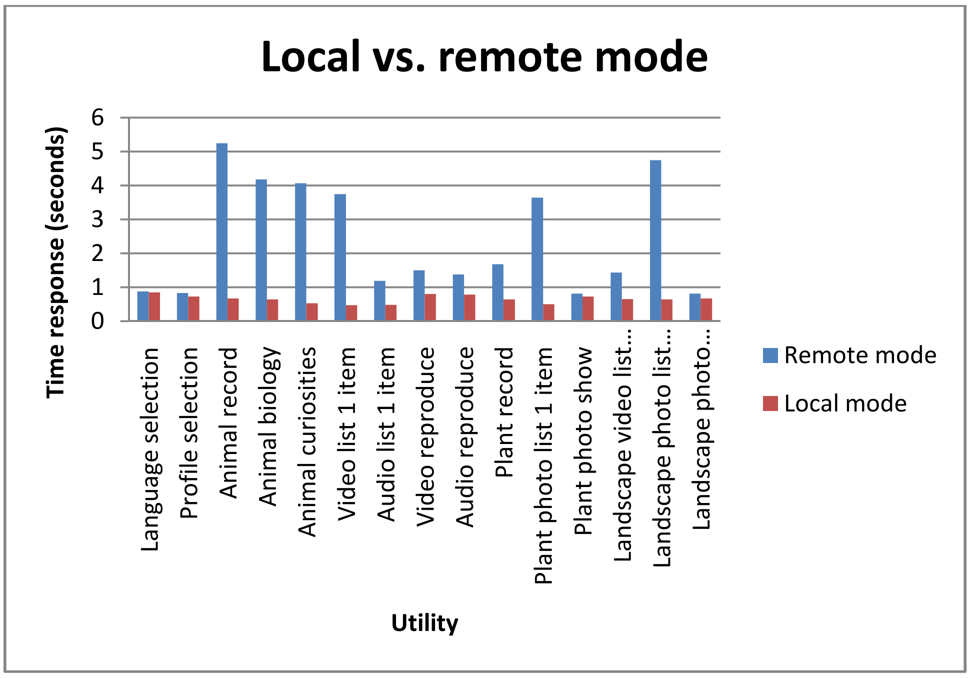


Figure 41. Time response comparison local vs. remote mode, for the different views.

We can conclude that remote mode time response is always higher than local mode's in all the possible utilities.

Next, we may be keen on finding out the application's performance in displaying lists of different types. For this purpose, we measure the time the application takes in displaying lists of different types of points of interest. The results are shown in figure 42.

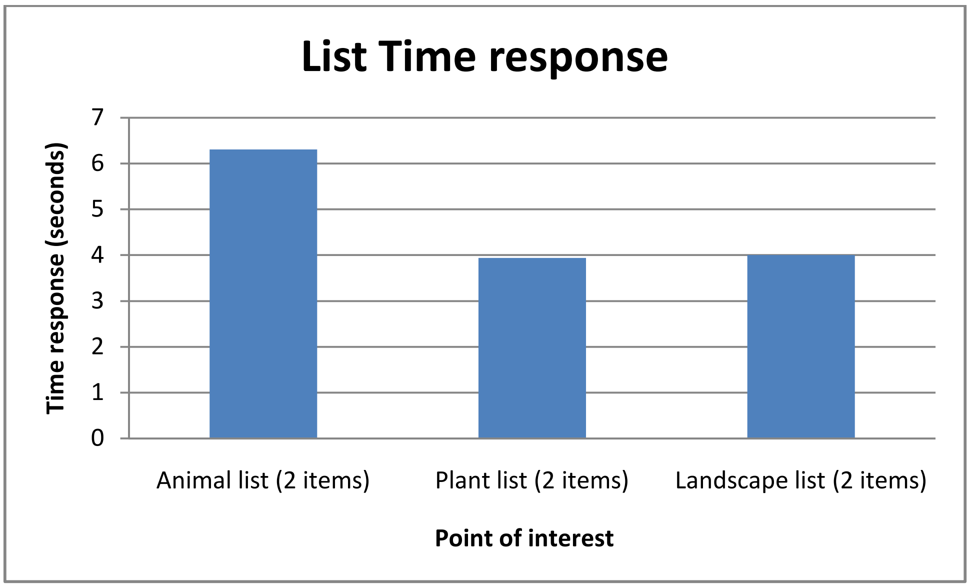


Figure 42. Time response for animal, plant and landscape 2-item lists.

From this chart we can make the same conclusion as in local mode: plant and landscape lists take less to appear than the animal list. This is due to the higher number of animals than plants and lists, and produced by the code design itself. In this occasion the difference between animal list and plant/landscape lists is even bigger, due to the communication's delay.

Next, we may be interested in discovering the time response for lists consisting of different number of items. For this purpose, the time response corresponding to different number of animals is under study. Results are shown in the following figure.

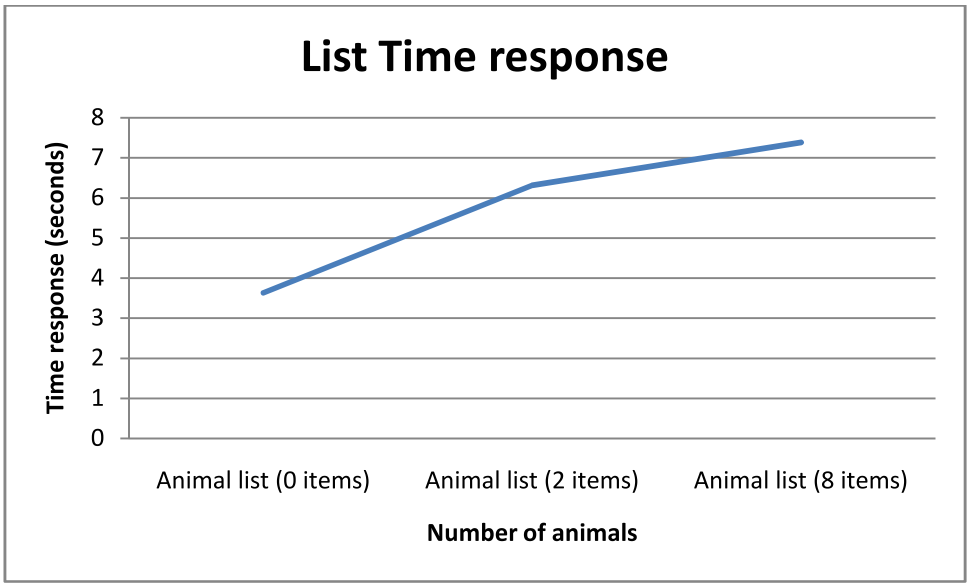


Figure 43. Time response for 0, 2 and 8 animals list.

A list containing 0 animals, takes 3.63 seconds to appear, instead of 0 seconds as expected. This 3.63 seconds includes communication delay between the device and the database server, query, generation of the XML file and XML parsing on the device. This way the figure is different from 0 seconds. For a list consisting of 2 animals, it will take 6.31 seconds to appear. If the list contains 8 animals, it will take 7.38 seconds to appear. So, if the number of animals increases, so does the time response.
In this case, a remarkable conclusion is that the remote mode introduces a considerable delay when a list has to appear, whereas the rest of the information has a reasonable time response.

The following experiments' main objective is to evaluate the system performance **varying the image size.**
First, images are stored in the iPhone device, that is, they are locally stored. We focus on evaluating the time it takes the application to display a list of 8 animals, depending on the images' size. The results can be seen in figure 44.

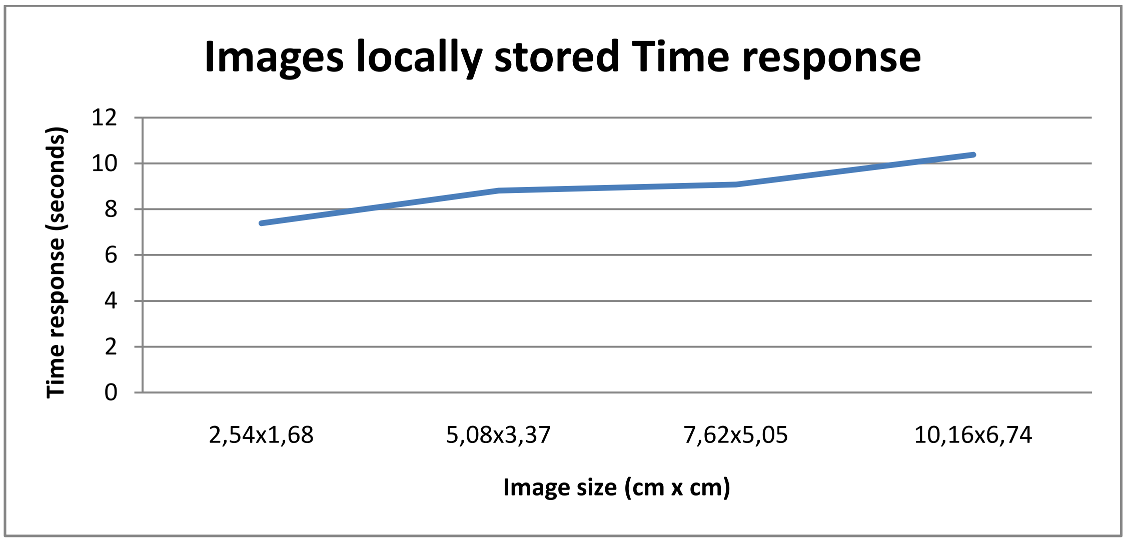


Figure 44. Time response for a list of 8 animals at different image sizes. Images in local. 314 pixels/inch.

This chart shows that the time response increases as image size grows, taking values from 7.38 to 10.37 seconds.
Below 2.54x1.68 size, the image is too small for the application, and the application does not support sizes above 20.32x13.48.
In conclusion, the user will not have to wait more than 10.4 seconds for the list of 8 animals to appear.

Now, different experiments **varying the image size** are carried out in order to discover how the application behaves displaying lists of 8 animals, as image size changes. In this occasion, images are **stored in the server**, that is, they are not stored locally in the device.

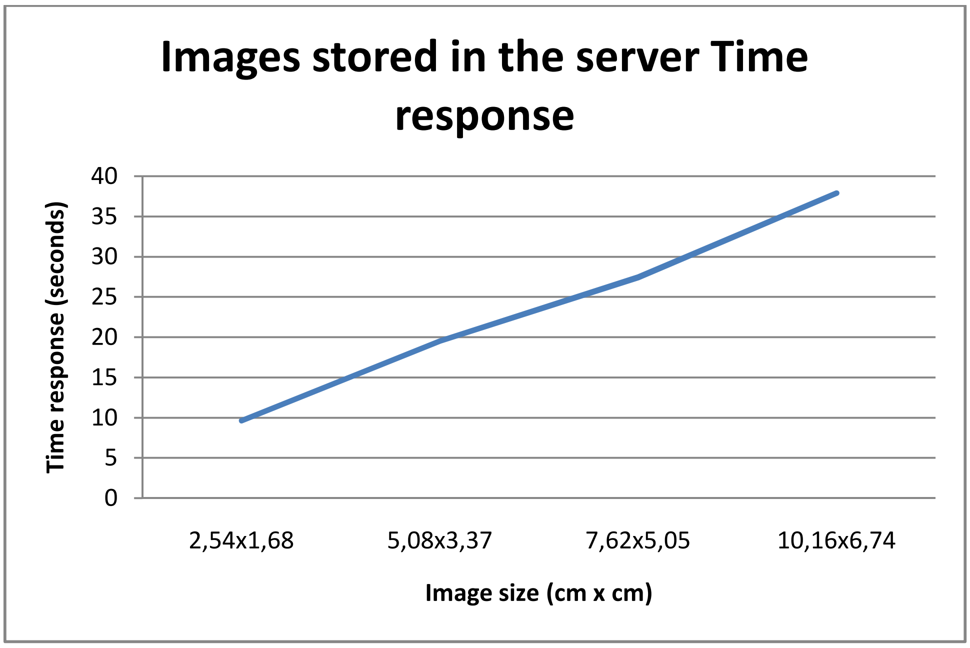


Figure 45. Time response for a list of 8 animals at different image sizes. Images in the server. 314 pixels/inch.

This chart shows that the time response increases as image size grows, taking values from 9.62 to 37.87 seconds.

Below 2.54x1.68, the image is too small for the application, and the application does not support sizes above 20.32x13.48.

In conclusion, storing the images in the server can save memory in the device. However it takes a long time to download images and appear in our iPhone's view.

The following chart shows the comparison of both modes.

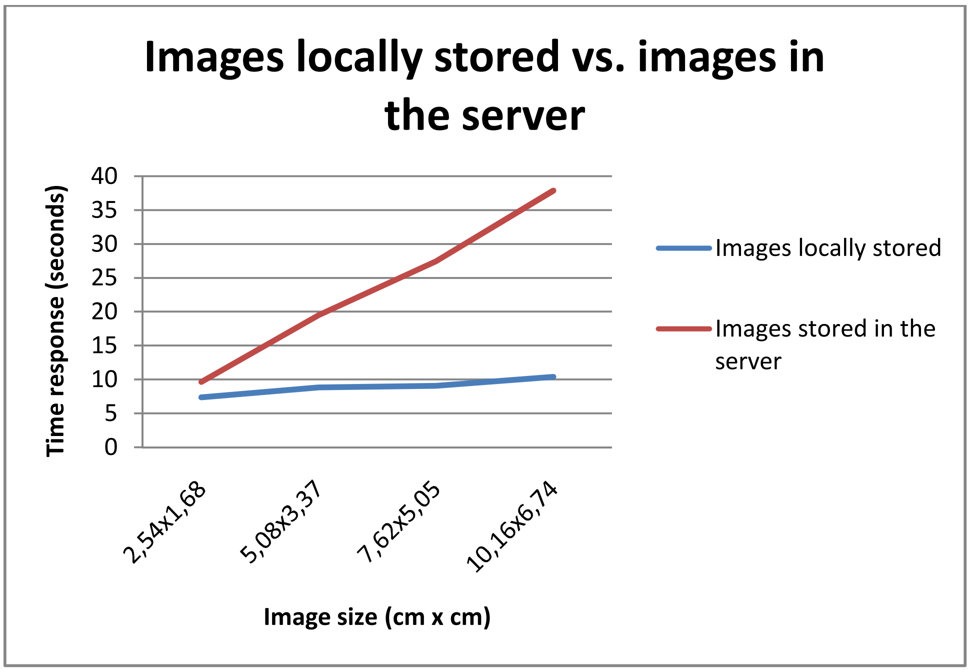


Figure 46. Time response (seconds) for two options: images locally stored and images stored in the server (8 animals list). 314 pixels/inch.

As we can see, performance is always better for images locally stored mode. Although its performance is almost the same for 2.54x1.68, it differs more and more as the image size increases.
A remarkable conclusion from this behavior is that we should decide between memory and time response.
Images stored in the server can save memory, but introduces high delays. In particular, delays higher than 15 seconds could be considered prohibitive if the user requires a low time response.
What is more, for 10.16x6.74 images, the size of each file is 180.5 KB. If we consider an 8 animals list, the amount of memory is 1444 KB, that is 1.41MB.
So at the maximum size, the memory required for this list is about 1.41 MB. In this case, the photos can be seen properly but the user has to wait 37.87 seconds, which is too much.
If more than 8 animals' lists are considered, this behavior could be even worse for the in-server mode, and the memory required would increase.
In conclusion, depending on the number of animals considered, the memory and time response requirements, we should choose between images stored locally or images stored in the server mode.

Other experiments have been carried out to evaluate remote mode time response, with 2.54x1.68 **images in the server**. In this case, we want to evaluate the application's performance in displaying lists consisting of different number of animals. The results are shown below.

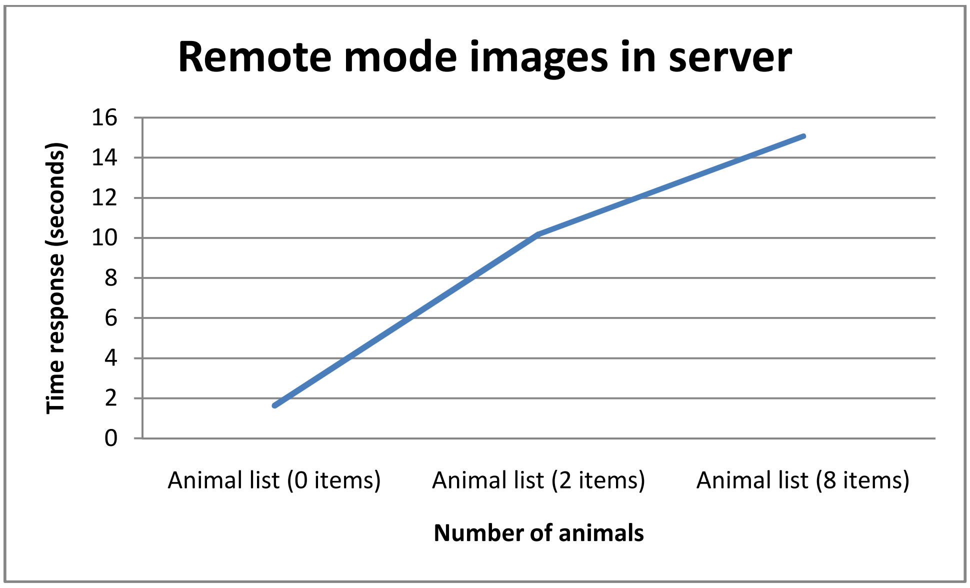


Figure 47. Animal list 0, 2 and 8 items for remote mode. Images in the server.

As we can see in Figure 47, as the number of animals of the list increases, so does the time response.

What is more, the time response for an Animal Record to appear has been measured, considering this conditions, and the result is that it takes to appear 3.43 seconds.
Next, we may be keen on comparing the time the application takes to show the animal's technical record. For this purpose, a comparison chart between locally stored images and images in server modes is shown.

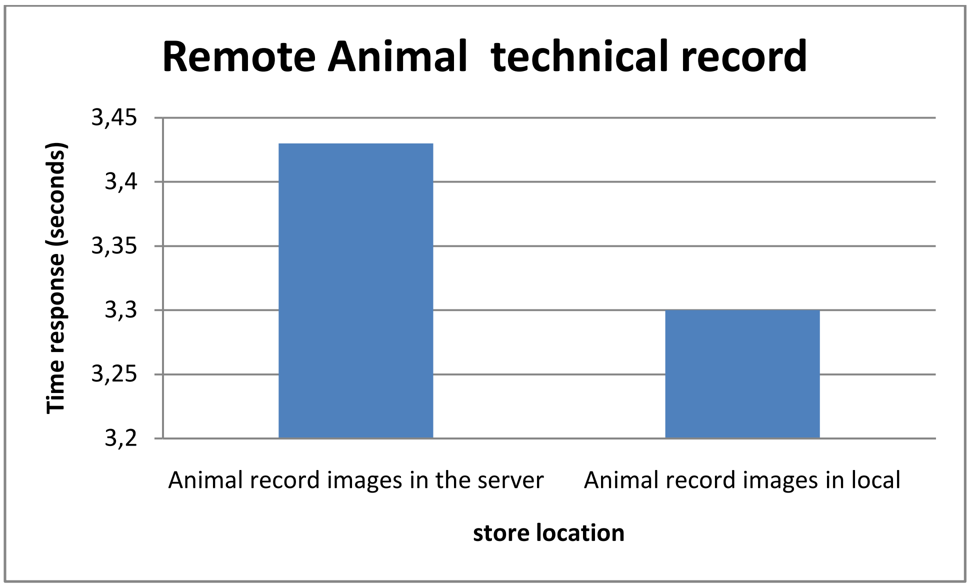


Figure 48. Animal technical record in locally stored images and images in the server.

Animal record takes less to appear if the images are locally stored, about 3.3 seconds, although the difference in time response between both modes is too small to be perceived.

The following chart's aim is to compare locally stored images and in-server stored images modes, for increasing number of animals' lists.

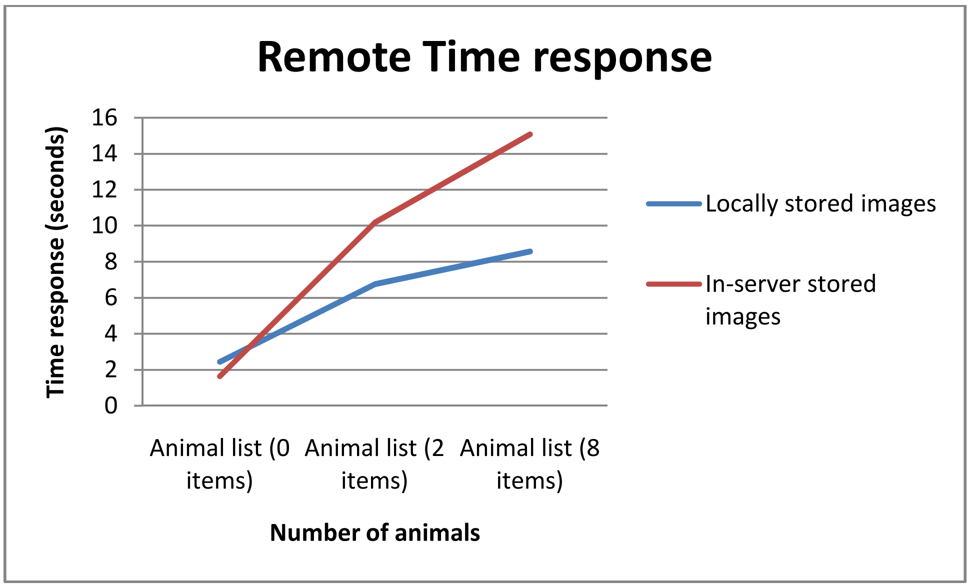


Figure 49. 0-item, 2-item and 8-item Animal list, for locally stored images and images in the server, remote modes.

Locally stored images remote mode time response is better than in-server stored images remote mode, for lists of any number of items.
We must remember that varying the image size, locally stored images was also better than in-server stored images.

Furthermore, we can be particularly keen on studying the system performance using the application far from the database server's location, that is, studying how the system behaves when the user connects offline to the server at high distances from it. For example, the user may choose the remote option when he is at home. So a couple of tests have been carried out at **47 km of distance (offline)** from the user to the database server. 3G connectivity is used to connect to the remote database server.

First, time response is evaluated within the different utility views in the remote mode. For this purpose, time response is measured using the application's different utilities.

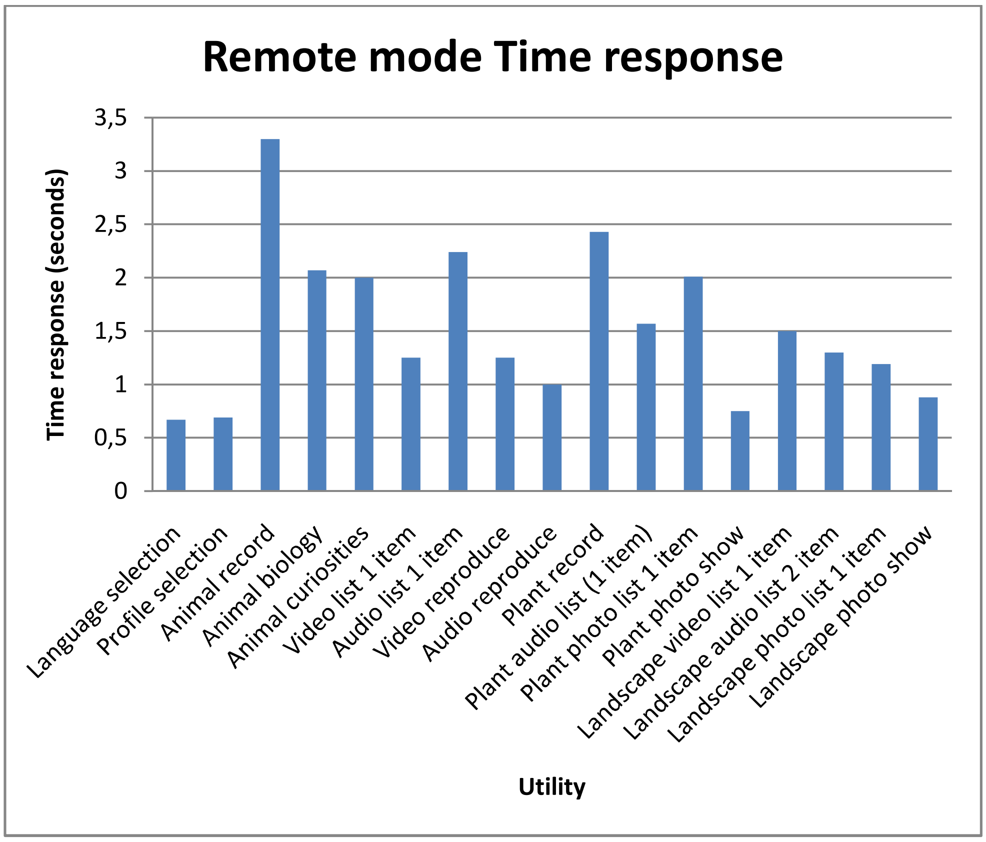


Figure 50. Time response for the different views in remote mode.

From this chart, we can conclude that the user will have to wait less than 3.5 seconds to view any of the information he is looking up. The information will take 1.53 seconds at an average and 0.72 of standard deviation to appear. What is more, 93.3% of the information will take less than 2.5 seconds to appear. The animal technical record has the highest time response with 3.3 seconds, which is really pretty fast.

A study about the time response in displaying lists of different types will also be presented. Time response is measured for different sort of lists. Results can be consulted in Figure 51.

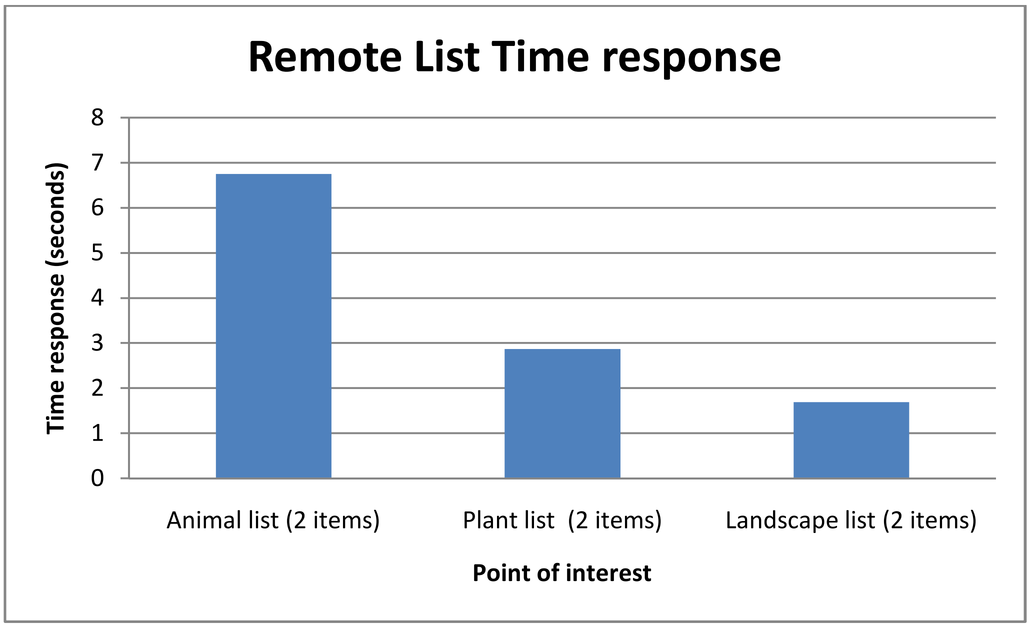


Figure 51. Time response for animal, plant and landscape 2-item lists in remote mode.

From this chart we can make the same conclusion as in local mode: plant and landscape lists take less to appear than the animal list, due to implementation features. In this occasion, the difference between animal list and plant/landscape lists is even bigger, and in all three cases time response is higher than in local mode, which was less than 1 second.

Finally the time response corresponding to different number of points of interest is under study. The time the application takes to display the list of animals is measured. The results can be seen in the following figure.

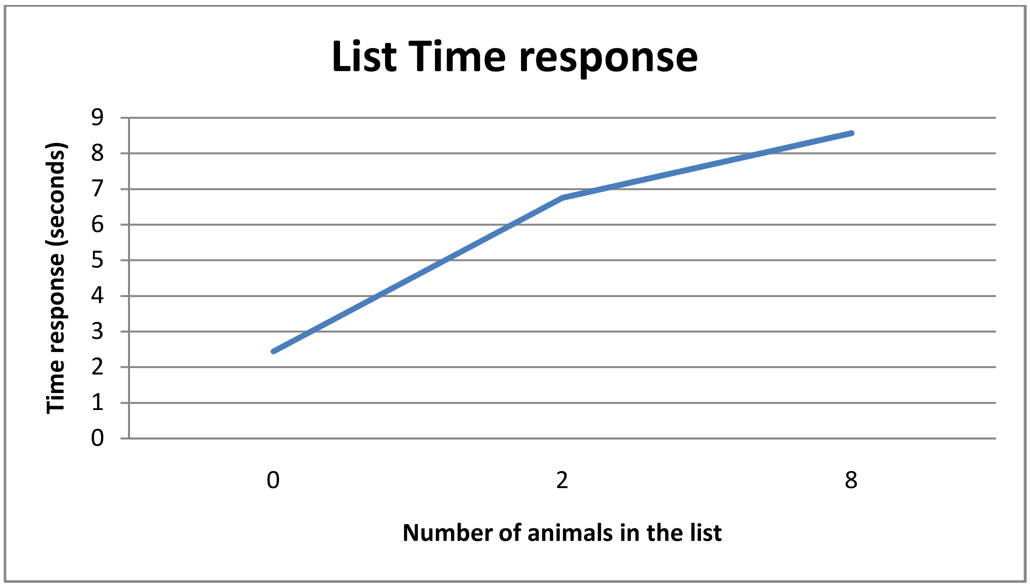


Figure 52. Time response for 0, 2 and 8 animals list, in remote mode.

A list containing 0 animals, takes 2.44 seconds to appear, instead of 0 seconds as expected. This 2.44 seconds includes communication delay between the device and the database server, query, generation of the XML file and XML parsing on the device. This way the figure is different from 0 seconds. For a list consisting of 2 animals, it will take 6.75 seconds to appear. If the list contains 8 animals, it will take 8.57 seconds to appear. So, if the number of animals increases, so does the time response.
In this case, a remarkable conclusion is that the remote mode introduces a considerable delay when a list has to be shown, whereas the rest of the information views have a reasonable time response.

Moreover**, context-aware** time response will be evaluated by carrying out several experiments. Remember that this mode shows a list with some animals or others depending on the user's location and orientation.
In particular, the areas considered for these experiments consist of 1 or 2 animals. So the user will see lists of 1 or 2 animals, depending on the area he is viewing.

First, this mode will be evaluated when the user changes orientation from an area with animals to another without animals. So time response must be measured by changing orientation from an area containing animals to a place containing no animals at both sides of the user's path. Five measures are taken to study its behavior. The results are shown below.

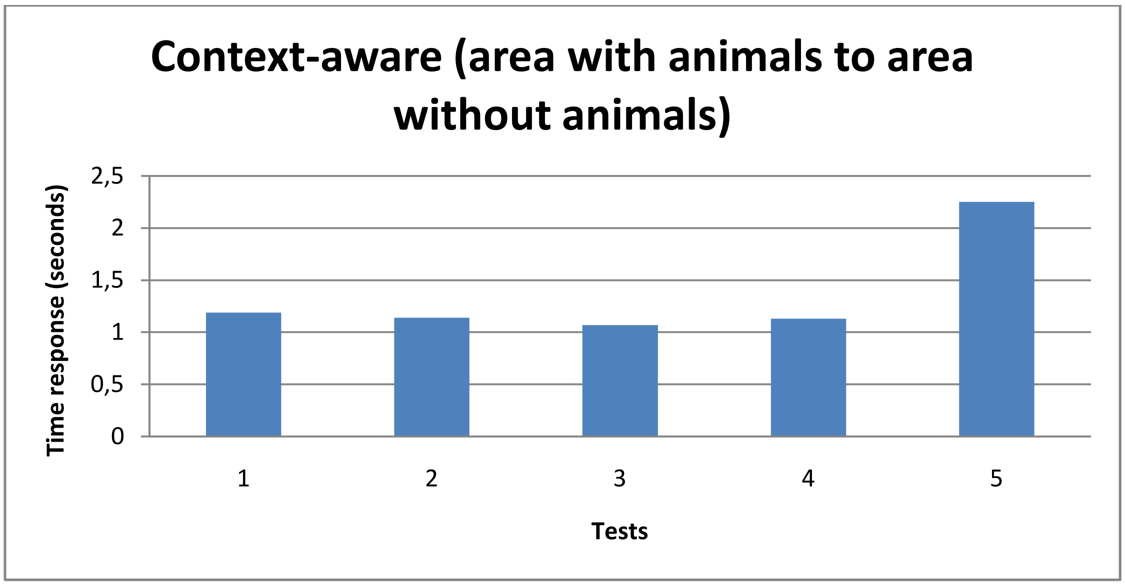


Figure 53. Context-aware time response (changing from an area with animals to another place without animals). 5 tests.

The mean value is 1.356 seconds, and the standard deviation is 0.45 seconds.

The maximum value taken is 2.25. This means that the user will have to wait less than 2.3 seconds to view the empty list. In 80% of the times, the user will have to wait around 1 second.

The following experiment's aim is to discover the mode's performance when the user changes orientation from an area without animals to another with animals. The experiment is carried out, changing orientation from a place without animals to an area containing animals, at both sides of the user's path. 5 measures have been taken. The results can be seen below.

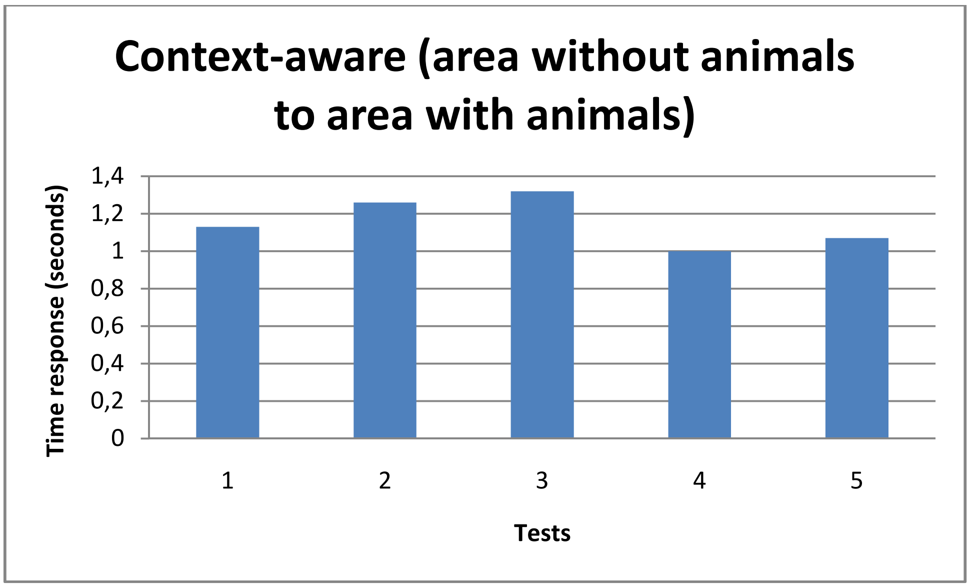


Figure 54. Context-aware time response (changing orientation from a place without animals to an area with animals). 5 tests.

The mean value, this time, is 1.156 seconds, whereas the standard deviation is 0.13 seconds. The user will have to wait 1.32 seconds or less to view the list of animals.

The following experiment considers measures taken changing orientation from an area with animals to another area with animals, at both sides of the user's path. This experiment has been carried out by changing orientation from an area to another and calculating the time response the user will have to wait including the time invested by the user in turning round to see the other area. 10 measures have been taken, obtaining the following results.

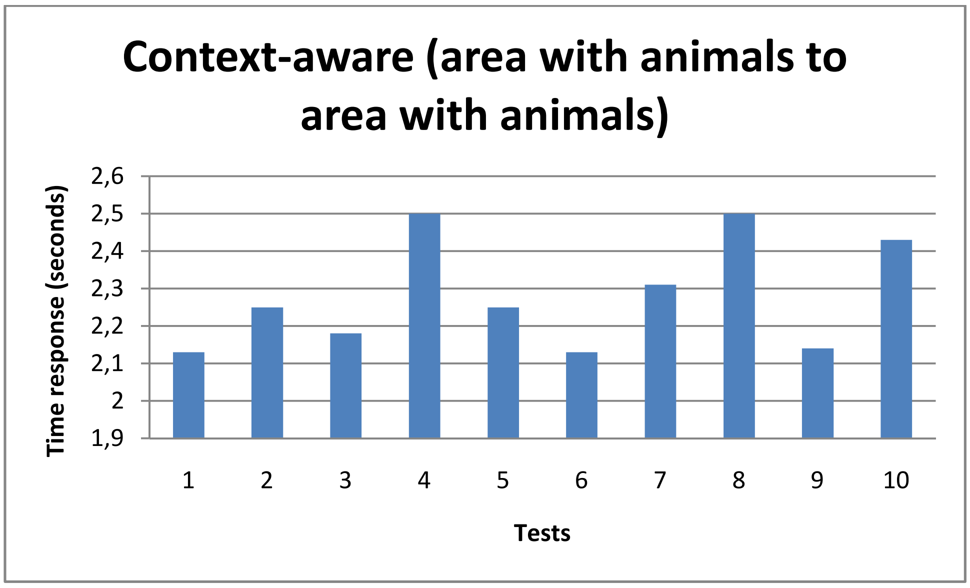


Figure 55. Context-aware time response (changing orientation from an area with animals to another area with animals). 10 tests.

The mean value is 2.282 seconds. The standard deviation is 0.1477 seconds. In 100% of the cases, the user will have to wait 2.5 seconds or less for the list of animals to change to the new area he is viewing.

We may consider of particular interest other time responses within context-aware mode, such as the time it takes the application to display the list of animals after tapping on the arrow button in the animal's pin located in the map. For this purpose, we take 5 measures by clicking the arrow button in the map and calculating the time the list of animals will take to appear.

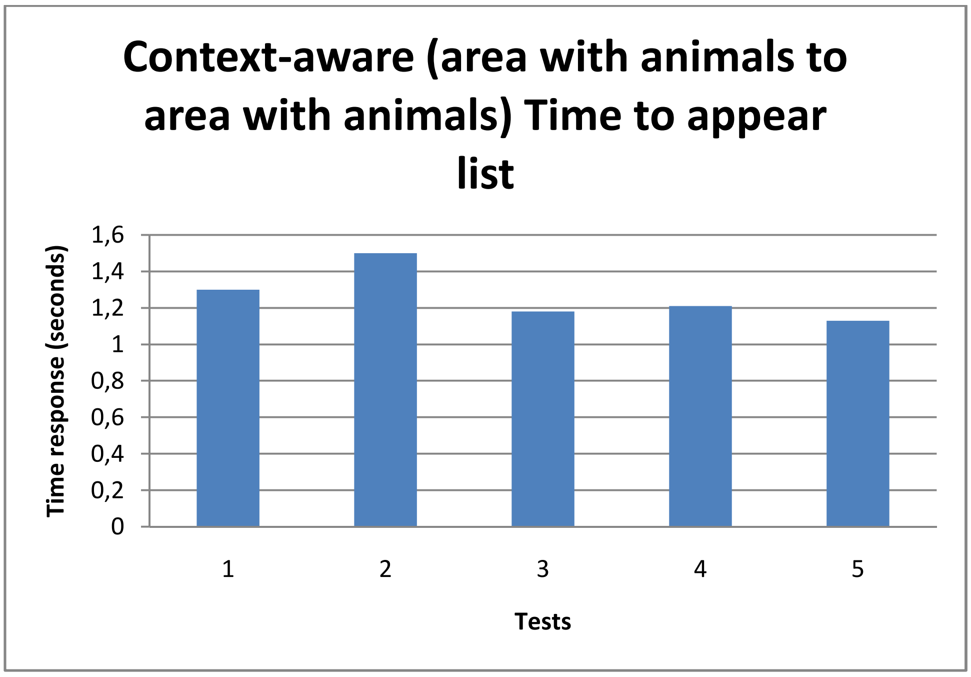


Figure 56. Context-aware time response (changing orientation from an area with animals to another area with animals). Time to appear the list of animals after tapping on the arrow button.

The mean value is 1.264 seconds, whereas the standard deviation is 0.145 seconds. The user will have to wait 1.5 seconds or less for the list of animals to appear after tapping on the arrow button.

If the user taps on a particular animal in the list, a time will take to show the technical record. If we measure the time that the user will have to wait for the technical record to appear, the following results are obtained.

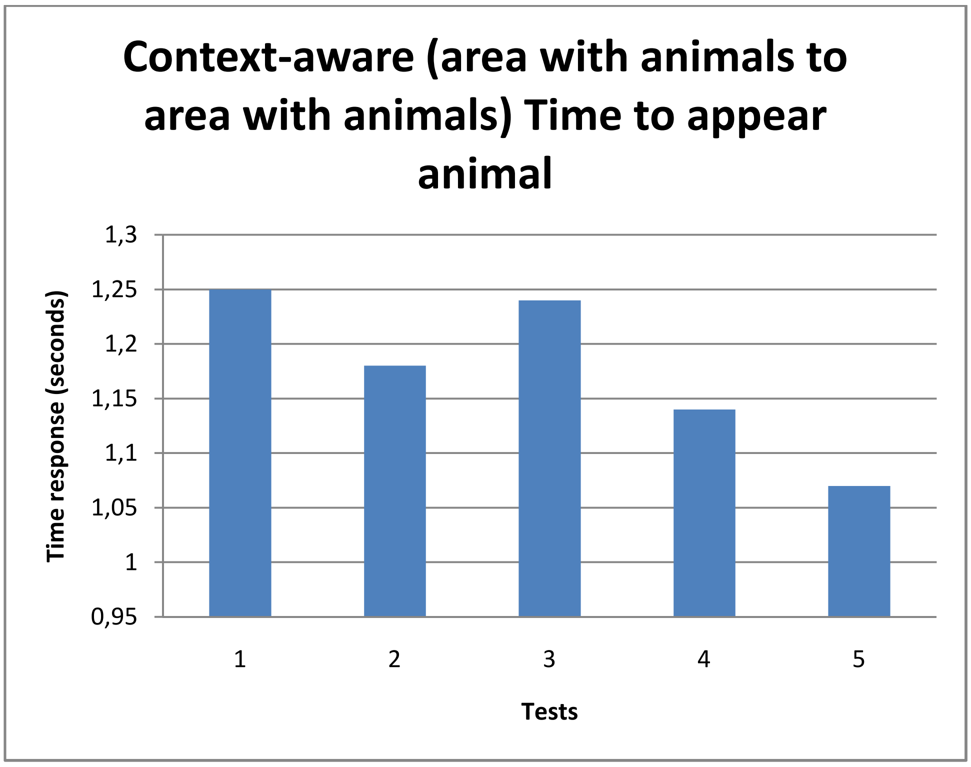


Figure 57. Context-aware time response. (changing orientation from an area with animals to another area with animals). Time to wait for the technical record to appear after tapping on the animal.

The mean calculated is 1.17 seconds, whereas the standard deviation is 0.074 seconds. So the user will have to wait less than 1.25 seconds for the technical record to appear after tapping on the animal.

To summarize, the user will see the list change on the device in a few seconds, less than 2.5 seconds in most of the cases to be precise, which is a reasonable figure.

In chapter 3, we proposed two approaches within the context-aware mode. The first one dealt "two-area" and "one-area" zones in a different way, whereas the second one considered "one-area" zones as "two-areas" zones with a 0 area at one of the sides. Four algorithms were presented. Algorithms 2 and 3 correspond to the first approach, while Algorithms 4 and 5 correspond to the second approach. Each approach had its advantages and drawbacks. The second approach was 100% accurate, whereas the first approach introduced a margin of error due to GPS chip inaccuracy.
Algorithms 2-3 and 4-5 will be under study to discover this inaccuracy.
For this purpose, 60 tests have been carried out to evaluate Algorithms 2-3 and Algorithms 4-5 accuracy. The results are shown in the following two figures.

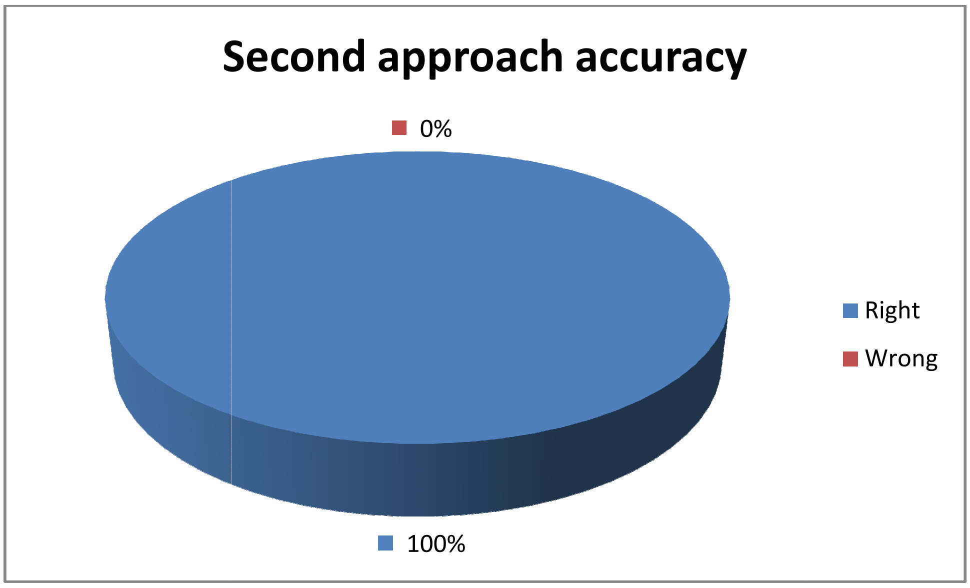


Figure 58. Algorithm 4 and 5 accuracy.

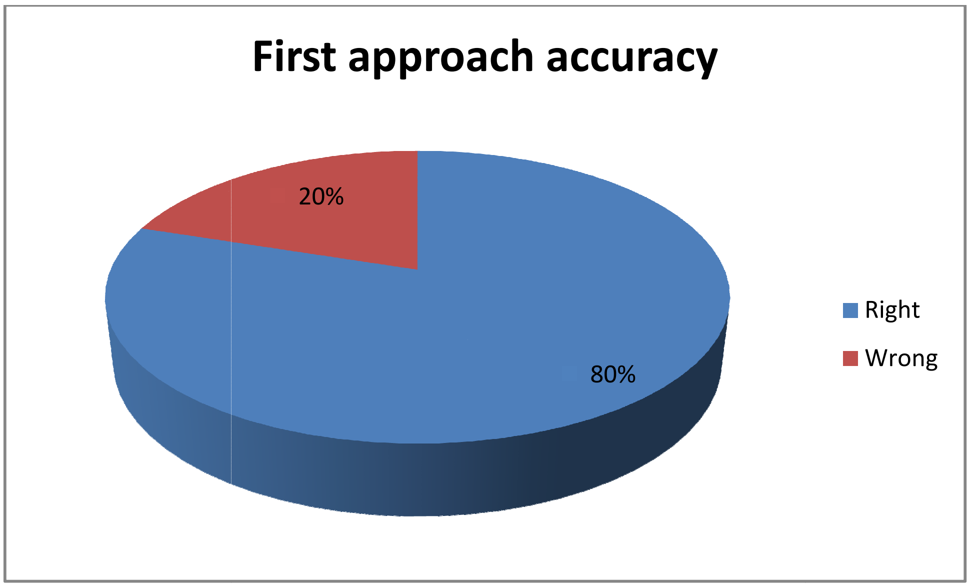


Figure 59. Algorithms 2 and 3 accuracy.

We can conclude that Algorithms 4 and 5 are 100% accurate, and never fail.
As GPS chip introduces location errors, Algorithms 2 and 3 won't be 100% accurate. This approach has an accuracy of 80%. This means that in 80% of the occasions the approach will offer valid results, whereas there is an error rate of 20%.
GPS inaccuracy is the main drawback that all iPhone applications using GPS location inherit. A GPS location given by the GPS chip may differ around 40 or 50 metres from its real location. This happens when our application uses the GPS location for the first time. As we continue using the application, the accuracy improves, producing errors of about 10 metres. Finally, accuracy reaches to a couple of metres errors.

This is the main problem our application has to cope with. For this reason, the second approach is proposed to be used in this application. Although the amount of memory needed by this second approach is higher and we must take more GPS points throughout the park, increasing the effort in doing this process, the accuracy is higher. So for high accuracy requirements, we should use the second approach.

Power consumption is also an important parameter that we should evaluate. This parameter will show us how long the battery will last.
In this case, the amount of time that the visitor can use the application is under study. The study is carried out dividing the application into the main system utilities, that is, animal searcher, nearest animal, context-aware, local textual, local video, local audio, remote textual, remote video and remote audio modes.
In this experiment, the percentage battery indicator has been useful to measure the battery life. If we measure the time that the device has decreased this percentage by one, and we multiply this time by 100, we will know the time the battery will discharge completely from 100% to 0%.
So we can measure the time the device has decreased this percentage by one, using one utility exclusively. Then we do the same thing using another utility exclusively. The same process has to be carried out using the different utilities. The results can be seen in Figure 60.

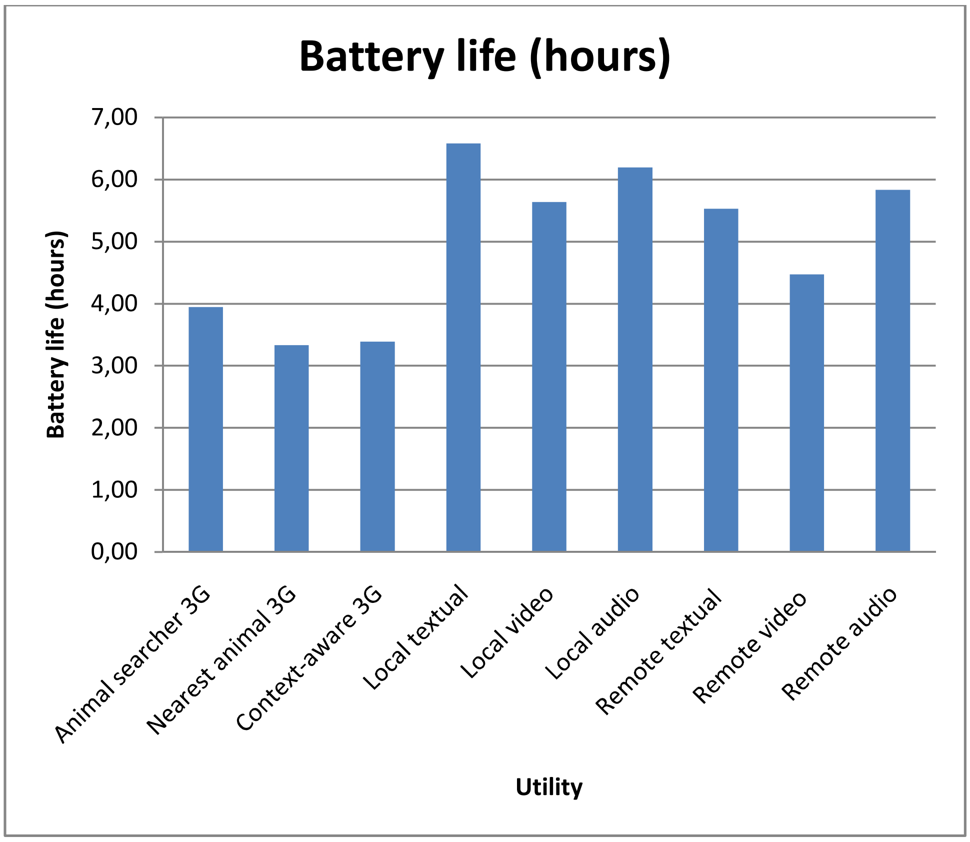


Figure 60: Battery life under different conditions.

The mean calculated for these data is 4.99 hours and the standard deviation is 1.23 hours.
From these results we observe that the user will be able to use the application at least 3.33 hours in any case before having to recharge the battery. This limit will happen when he is continuously using the nearest animal utility and only this utility.
At an average the user will be able to use this application about 5 hours. When the user looks up textual information and photos, he will be able to stay a maximum of 6.58 hours before having to recharge the battery.
GPS and compass utilities will suppose higher consumption than others, and remote option will cause more power consumption than local option.
What is more, the video option will suppose higher consumption than textual and audio options.
To summarize, if the visitor uses all the application utilities for about the same amount of time, he will be able to use it for about 5 hours without having to recharge the battery, which is a reasonable figure.

Chapter 5 Conclusions

In this document, we demonstrate that GPS and WIFI might be a candidate wireless networking technology to support context-aware applications.

UbiqBIOPARC was presented, as an experimental GPS and WIFI-based context-aware application developed in iPhone SDK. UbiqBIOPARC combines the convenience and flexibility of iPhone SDK with the universal connectivity of WIFI technology, GPS location, and compass readings.

The system was designed with the goal of providing visitors information about what they are viewing at their level of knowledge and in the language they prefer. A graphical interface developed for iPhone mobile devices, improve their experience.

A central data server has been designed and set up, containing a SQL Server 2005 database. Clients and ASP scripts have been written, to offer routines to make client requests to a remote database server. GPS location and compass readings have been managed to provide map-based utilities on animal searching and nearest animal options. Finally a context-aware option is available, which shows the list of animals the user is viewing, and changes it as the user's location or orientation change without user's intervention. Several schemes and algorithms have been proposed and evaluated within this thesis, that provide users with the context-aware utility.

The practicality of building a context-aware system that integrates a combination of GPS, WIFI, compass and Ethernet technologies.

Performance evaluation of the application has been carried out focusing on time response and power consumption.

The impact on time response was evaluated in local, remote, nearest animal, animal searcher and context-aware modes. Time response varying image size, storing images in the server and locally on the device, and varying the distance from the user to the server, were under study. Some other experiments have been carried out to evaluate time response varying the number of devices which are running the application at the same time.

We observed that the application offered a local mode time response of around 1 second in any case. As for remote mode, reasonable figures are obtained except for animal list displaying that reach up to 8.5 second response.

Other experiments compare locally stored images vs. images stored in the server modes when the image sizes vary. Locally stored images mode is faster, and as the image size increases the difference between these modes is even bigger. For high image size, images stored remotely mode takes up to 38 seconds, which could be considered prohibitive. Number of animals, memory saving and time response should be considered when selecting one mode or another.

Furthermore, we concluded that locally stored images remote mode time response is better than in-server stored images remote mode, for lists of any number of items.

In Nearest Animal mode, the user must wait less than 10 seconds to see the updated pin in the map, whereas in Animal Searcher he will have to wait less than 11.5 seconds. In Animal Searcher mode, the user will have to wait for the animal he is looking for to appear in the map less than 2 seconds.

Context-aware mode results prove that the user will have to wait less than 3.5 seconds for the information to appear or change in the device, which is a fast time response.

Two approaches have been proposed in order to implement context-aware. One of them has 80% accuracy depending on the GPS chip error introduced, whereas the other is 100% accurate.

Power consumption within all the modes has also been evaluated. These experiments prove that the user will be able to use the application for at least 3.33 hours, worst case figure that corresponds to using the nearest animal mode all the time. At an average, that is using all the utilities the same amount of time, the battery will last about 5 hours. GPS utilities suppose higher power consumption than the other modes, whereas local mode saves more power than the other modes. The application will be able to be used a maximum of 6.58 hours before having to charge the battery. Anyway, these figures show us that this application's power consumption is not too high for the system requirements and the battery will last enough time to visit the park.

Further research work should be done to improve the remote mode time response, and increasing even more the battery life in modes such as nearest animal, animal searching and context-aware, which are available for the user at the moment for less than 4 hours.

References

1. J.C Cano, P. Manzoni, C.K. Toh, "UbiqMuseum: A Bluetooth and Java Based Context-Aware System for Ubiquitous Computing", Polytechnic University of Valencia Spain, University of Hong Kong, 2006
2. "Apple looking into location-temporary apps" http://macapper.com/2010/05/18/apple-looking-into-location-temporary-apps/ May 2010
3. Command Line Shell for SQLite. http://www.sqlite.org/sqlite.html
4. SQLite Tutorial. http://souptonuts.sourceforge.net/readme_sqlite_tutorial.html
5. Objective-C: Use NSXMLParser. http://www.radupoenaru.com/objective-c-use-nsxmlparser/
6. Tree-Based XML Programming Guide: Introduction to Tree-Based XML Programming Guide for Cocoa. http://developer.apple.com/mac/library/documentation/Cocoa/Conceptual/NSXML_Concepts/NSXML.html
7. YouTube-Tutorial 9 iPhone SDK: MKMapView avanzado. http://developer.apple.com/mac/library/documentation/Cocoa/Conceptual/NSXML_Concepts/NSXML.html
8. iOS Application Programming Guide: About iOS Application Design. http://developer.apple.com/iphone/library/documentation/iPhone/Conceptual/iPhoneOSProgrammingGuide/Introduction/Introduction.html
9. Drawing polylines or routes on a MKMapView (using Map Kit on the iPhone). http://spitzkoff.com/craig/?p=65
10. Drawing polylines or routes on a MKMapView (as an MKAnnotationView) Part 2. http://spitzkoff.com/craig/?p=108
11. UITabBar Class Reference. http://developer.apple.com/iphone/library/documentation/uikit/reference/UITabBar_Class/Reference/Reference.html#//apple_ref/occ/instp/UITabBar/items
12. iPhone Tutorial: Adding a search bar in TableView. http://blog.webscale.co.in/?p=228
13. YouTube. Tutorial 13 iPhone SDK: UITableView basico. http://www.youtube.com/watch?v=yUtV6eysuLo&NR=1
14. YouTube. Xcode iPhone SDK Tutorial. Assigning Image to a UIImageView. http://www.youtube.com/watch?v=66jxild-gH4
15. YouTube. Xcode iPhone SDK Tutorial. Playing a Video File With Controls. http://www.youtube.com/watch?v=9hL5jREBwu4
16. Custom MKPinAnnotation callout bubble similar to default callout bubble. http://loopingrecursion.com/index.php?t=mkpinannotationview
17. El lenguaje Objective-C para programadores C++ y Java.

18. NSString class reference. http://developer.apple.com/mac/library/documentation/cocoa/reference/Foundation/Classes/NSString_Class/Reference/NSString.html
19. Taller de ASP. http://www.desarrolloweb.com/manuales/11/
20. Tutorial Request: Core Location. http://www.iphonedevsdk.com/forum/tutorial-requests/2343-tutorial-request-core-location.html
21. View Controller Programming Guide for iOS: Tab Bar Controllers. http://developer.apple.com/iphone/library/featuredarticles/ViewControllerPGforiPhoneOS/TabBarControllers/TabBarControllers.html
22. Global variables in Cocoa. http://www.cocoabuilder.com/archive/cocoa/73592-where-to-put-global-variables-in-cocoa.html
23. XML Tutorial. http://www.w3schools.com/XML/default.asp
24. YouTube: iPhone SDK Tutorial – Implementing Multiple Views Part 1: going down the rabbit hole. http://www.youtube.com/watch?v=m9Ii0tGvPUM
25. How to write an XML parser app with the SDK. http://www.iphonexe.com/blog/?p=74
26. SQLite Manager, gestiona bases de datos SQLite en Firefox. http://www.visualbeta.es/1183/navegadores/sqlite-manager-gestiona-bases-de-datos-sqlite-en-firefox/
27. CLLocation – getDistanceFrom – iPhone Dev SDK Forum. http://www.iphonedevsdk.com/forum/iphone-sdk-development/19587-cllocation-getdistancefrom.html
28. CLLocation class reference. http://developer.apple.com/iphone/library/documentation/CoreLocation/Reference/CLLocation_Class/CLLocation/CLLocation.html
29. How to use sqlite with a UITableView in iPhone SDK. http://blog.objectgraph.com/index.php/2010/04/08/how-to-use-sqlite-with-uitableview-in-iphone-sdk/
30. ASP tutorial. http://www.w3schools.com/asp/default.asp
31. Cómo crear un XML desde ASP. http://www.webtaller.com/construccion/lenguajes/asp/lecciones/crear_xml.php
32. Cocoa with Love: Singletons, AppDelegates and top-level data. http://cocoawithlove.com/2008/11/singletons-appdelegates-and-top-level.html
33. MKAnnotationView class reference. http://developer.apple.com/iphone/library/documentation/MapKit/Reference/MKAnnotationView_Class/Reference/Reference.html#//apple_ref/occ/instp/MKAnnotationView/image
34. How do I use NSTimer. http://stackoverflow.com/questions/1449035/how-do-i-use-nstimer
35. Compass Programming Guide – iPhone Dev SDK Forum. http://www.iphonedevsdk.com/forum/iphone-sdk-development/21553-compass-programming-guide.html

36. UISwich class reference. http://developer.apple.com/iphone/library/documentation/uikit/reference/UISwitch_Class/Reference/Reference.html

Appendix 1
Application UML diagram

The application developed consists of several view controllers and their corresponding views. Each view controller is associated to a view.
These view controllers are represented in the UML diagram in the following figure.

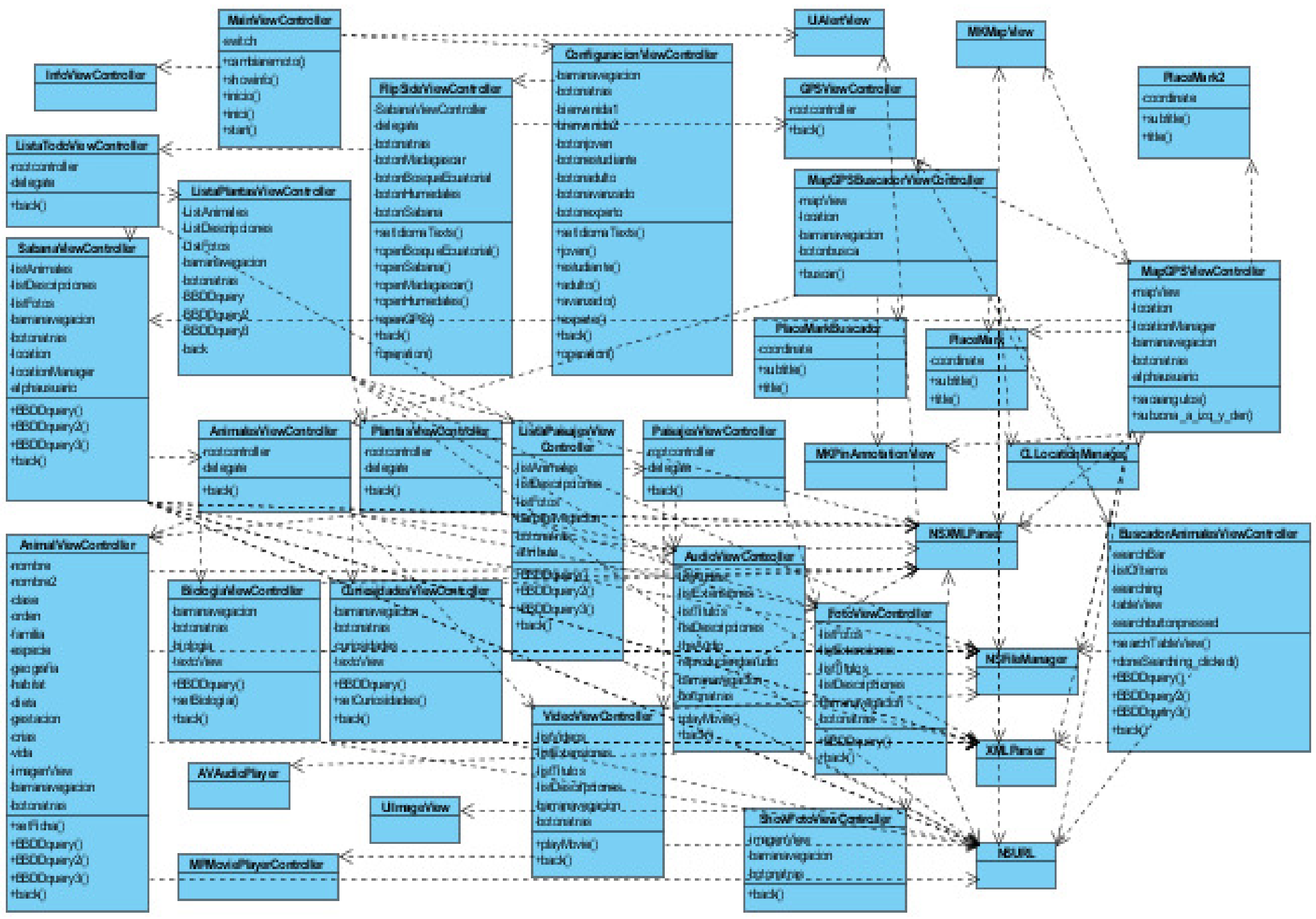


Figure 61: Application UML Diagram.

The main view controllers, and the functions they implement, are described below.

1- *MainViewController*: It is called when the application starts. It consists of a switch, an info button and three language buttons. When the switch is selected the *cambiaremoto* method is called. If the user chooses the info button the *showinfo* method is called. *Inicio, inici, start* methods run when selecting each one of the language buttons.
2- *InfoViewController*: contains information on the application and the system designer.
3- *FlipSideViewController* implements the areas menu, including buttons to Madagascar, Equatorial Forest, Pickleweed, Savanna, and the GPS option button.
4- *ConfiguracionViewController*: Controller associated to the set up view. It implements the Welcome message and the profile buttons.
5- *ListaTodoViewController*: This controller implements the view containing three buttons to change among list of animals, plants and landscapes.
6- *SabanaViewController*: Shows the list of animals, including the name, a brief description and a photo.
7- *ListaPlantasViewController*: Shows the list of plants, including the name, a brief description and a photo.
8- *ListaPaisajesViewController*: Shows the list of landscapes, including the name, a brief description and a photo.
9- *AnimalesViewController*: This controller implements the animal view containing 5 subviews to change among technical record, biology, curiosities, video and audio.
10- *PlantasViewController*: This controller implements the plant view containing 3 subviews to change among technical record, audio and photos.
11- *PaisajesViewController*: This controller implements the landscape view containing 3 subviews to change among videos, audio and photos.
12- *AnimalViewController*: It shows the technical record including all the information classified in fields, such as class, order, family and so on.
13- *BiologiaViewController*: A text about the animal biology features is shown.
14- *CuriosidadesViewController:* A text about the animal curiosities is shown.
15- *VideoViewController:* A list of video files, including a title and a brief description, is shown.
16- *AudioViewController:* A list of audio files, including a title and a brief description, is shown.
17- *FotoViewController:* A list of photos, including a title and a brief description, is shown.
18- *ShowFotoViewController:* A view showing the photo is displayed.
19- *GPSViewController:* It shows 2 buttons used to change between the "Search animal" option and the "Nearest animal" option.
20- *MapGPSBuscadorViewController:* It implements the "Search animal" option, including a map view, the user's location, the animal's location and a search button. When the user taps on the search button, it flips to the *BuscadorAnimalesViewController*.

21- *BuscadorAnimalesViewController:* It shows the list of animals, a search field, and a contextual keyboard, allowing users to select an animal and search for it.
22- *PlaceMarkBuscador:* It handles all the information on the animal pin that appears in the map.
23- *MapGPSViewController:* It implements the "Nearest animal" option, including a map view, the user's location, and the nearest animal location. This controller also implements the required methods on positioning and orientation to find out which area of animals the user is viewing.
24- *PlaceMark:* It handles all the information on the user location pin.
25- *PlaceMark2:* It handles all the information on the nearest animal pin.

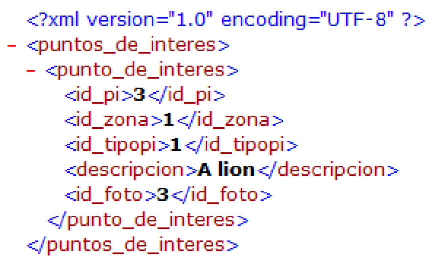
The rest of the classes in Figure 61, used by the application, are included in the following Frameworks:

1. *CoreLocation.framework*: *CLLocationManager* class is used. It offers location capabilities.
2. *MapKit.framework*: *MKPinAnnotationView*, *MKMapView* classes are used. It handles map utilities.
3. *Libsqlite3.dylib*: *sqlite3.h* class is used. Sqlite3 database management is offered.
4. *MediaPlayer.framework*: *MPMoviePlayerController* class is used. Playing videos is possible thanks to this framework.
5. *AVFoundation.framework*: it is used by means of the *AVAudioPlayer* class. Playing audio is possible thanks to this framework.
6. *Foundation.framework*: *NSURL, NSXMLParser, XMLParser, NSFileManager* are used. Files, urls and XML parsing are supplied by this framework.
7. *UIKit.framework*: *UIAlertView, UIImageView* are used. These classes are useful to show alerts or display images.

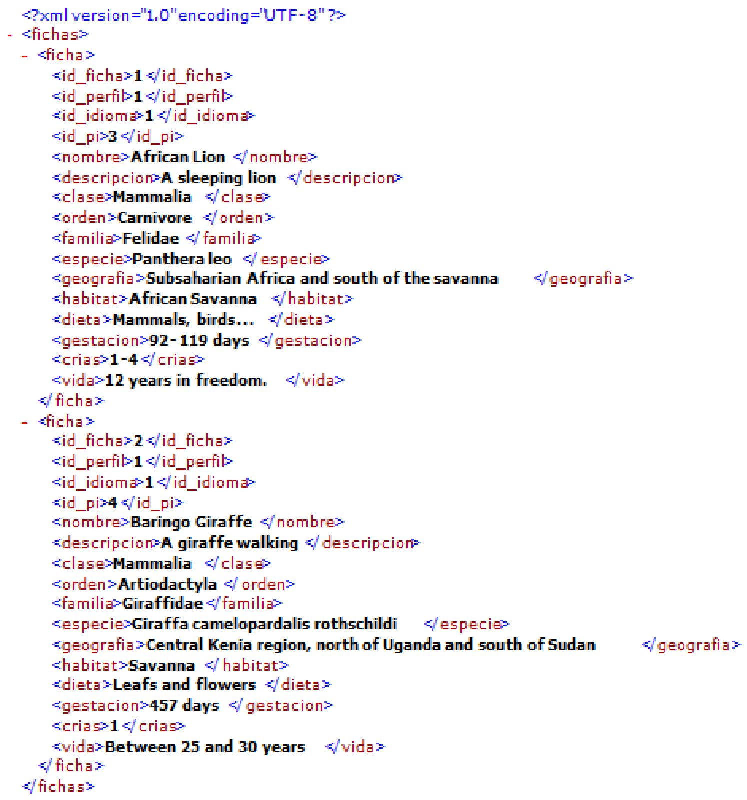
Appendix 2

XML Files structure

This Appendix consists of a series of XML structure examples. They can be used as a guide to design the ASP script.



```
  <?xml version="1.0" encoding="UTF-8" ?>
- <puntos_de_interes>
  - <punto_de_interes>
      <id_pi>3</id_pi>
      <id_zona>1</id_zona>
      <id_tipopi>1</id_tipopi>
      <descripcion>A lion</descripcion>
      <id_foto>3</id_foto>
    </punto_de_interes>
  </puntos_de_interes>
```

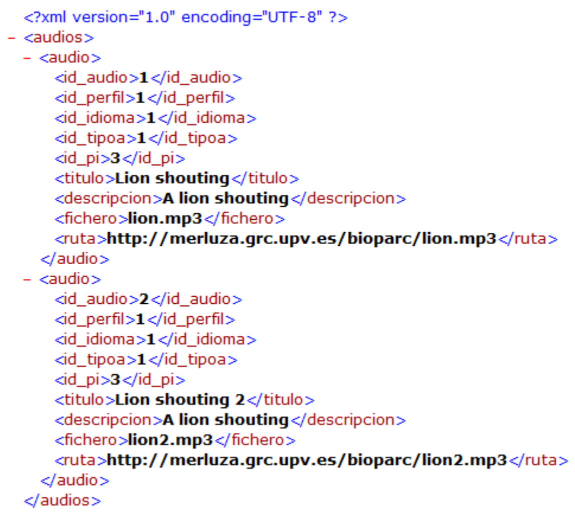


```
  <?xml version="1.0"encoding="UTF-8" ?>
- <fichas>
  - <ficha>
      <id_ficha>1 </id_ficha>
      <id_perfil>1 </id_perfil>
      <id_idioma>1 </id_idioma>
      <id_pi>3 </id_pi>
      <nombre>African Lion </nombre>
      <descripcion>A sleeping lion </descripcion>
      <clase>Mammalia </clase>
      <orden>Carnivore </orden>
      <familia>Felidae </familia>
      <especie>Panthera leo </especie>
      <geografia>Subsaharian Africa and south of the savanna     </geografia>
      <habitat>African Savanna </habitat>
      <dieta>Mammals, birds... </dieta>
      <gestacion>92-119 days </gestacion>
      <crias>1-4 </crias>
      <vida>12 years in freedom.  </vida>
    </ficha>
  - <ficha>
      <id_ficha>2 </id_ficha>
      <id_perfil>1 </id_perfil>
      <id_idioma>1 </id_idioma>
      <id_pi>4 </id_pi>
      <nombre>Baringo Giraffe </nombre>
      <descripcion>A giraffe walking </descripcion>
      <clase>Mammalia </clase>
      <orden>Artiodactyla </orden>
      <familia>Giraffidae </familia>
      <especie>Giraffa camelopardalis rothschildi    </especie>
      <geografia>Central Kenia region, north of Uganda and south of Sudan     </geografia>
      <habitat>Savanna </habitat>
      <dieta>Leafs and flowers </dieta>
      <gestacion>457 days </gestacion>
      <crias>1 </crias>
      <vida>Between 25 and 30 years   </vida>
    </ficha>
  </fichas>
  <?xml version="1.0" encoding="UTF-8" ?>
- <fotos>
  - <foto>
      <id_foto>1</id_foto>
      <id_pi>3</id_pi>
      <titulo>Lion</titulo>
      <descripcion>A lion sleeping</descripcion>
      <fichero>lion.jpg</fichero>
      <ruta>http://merluza.grc.upv.es/bioparc/lion.jpg</ruta>
    </foto>
  </fotos>
```

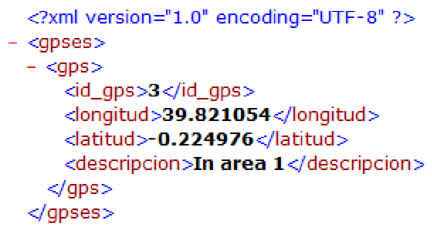
```
  <?xml version="1.0" encoding="UTF-8" ?>
- <biologias>
  - <biologia>
      <id_biologia>1</id_biologia>
      <id_pi>3</id_pi>
      <id_idioma>1</id_idioma>
      <id_perfil>1</id_perfil>
      <id_tipob>1</id_tipob>
      <descripcion>The lion is the king</descripcion>
    </biologia>
  </biologias>
```

```
  <?xml version="1.0" encoding="UTF-8" ?>
- <curiosidades>
  - <curiosidad>
      <id_curiosidad>1</id_curiosidad>
      <id_pi>3</id_pi>
      <id_idioma>1</id_idioma>
      <id_perfil>1</id_perfil>
      <descripcion>The lion likes sleeping after lunch</descripcion>
    </curiosidad>
  </curiosidades>
```

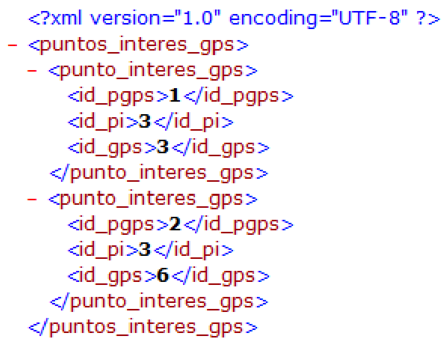
```
  <?xml version="1.0" encoding="UTF-8" ?>
- <videos>
  - <video>
      <id_video>1</id_video>
      <id_perfil>1</id_perfil>
      <id_idioma>1</id_idioma>
      <id_tipov>1</id_tipov>
      <id_pi>3</id_pi>
      <titulo>Lions</titulo>
      <descripcion>Lions fighting</descripcion>
      <fichero>lion.mov</fichero>
      <ruta>http://merluza.grc.upv.es/bioparc/lion.mov</ruta>
    </video>
  - <video>
      <id_video>2</id_video>
      <id_perfil>1</id_perfil>
      <id_idioma>1</id_idioma>
      <id_tipov>1</id_tipov>
      <id_pi>3</id_pi>
      <titulo>Second Lion</titulo>
      <descripcion>Lion on the rock</descripcion>
      <fichero>lion2.mov</fichero>
      <ruta>http://merluza.grc.upv.es/bioparc/lion2.mov</ruta>
    </video>
  </videos>
```



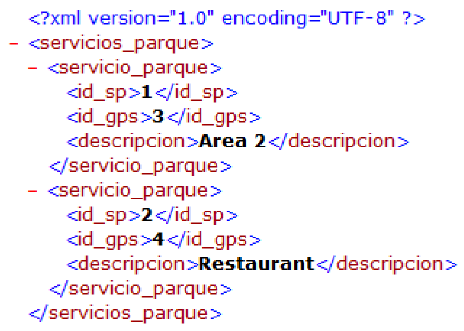
```
  <?xml version="1.0" encoding="UTF-8" ?>
- <audios>
  - <audio>
      <id_audio>1</id_audio>
      <id_perfil>1</id_perfil>
      <id_idioma>1</id_idioma>
      <id_tipoa>1</id_tipoa>
      <id_pi>3</id_pi>
      <titulo>Lion shouting</titulo>
      <descripcion>A lion shouting</descripcion>
      <fichero>lion.mp3</fichero>
      <ruta>http://merluza.grc.upv.es/bioparc/lion.mp3</ruta>
    </audio>
  - <audio>
      <id_audio>2</id_audio>
      <id_perfil>1</id_perfil>
      <id_idioma>1</id_idioma>
      <id_tipoa>1</id_tipoa>
      <id_pi>3</id_pi>
      <titulo>Lion shouting 2</titulo>
      <descripcion>A lion shouting</descripcion>
      <fichero>lion2.mp3</fichero>
      <ruta>http://merluza.grc.upv.es/bioparc/lion2.mp3</ruta>
    </audio>
  </audios>
```



```
  <?xml version="1.0" encoding="UTF-8" ?>
- <gpses>
  - <gps>
      <id_gps>3</id_gps>
      <longitud>39.821054</longitud>
      <latitud>-0.224976</latitud>
      <descripcion>In area 1</descripcion>
    </gps>
  </gpses>
```



```
  <?xml version="1.0" encoding="UTF-8" ?>
- <puntos_interes_gps>
  - <punto_interes_gps>
      <id_pgps>1</id_pgps>
      <id_pi>3</id_pi>
      <id_gps>3</id_gps>
    </punto_interes_gps>
  - <punto_interes_gps>
      <id_pgps>2</id_pgps>
      <id_pi>3</id_pi>
      <id_gps>6</id_gps>
    </punto_interes_gps>
  </puntos_interes_gps>
```



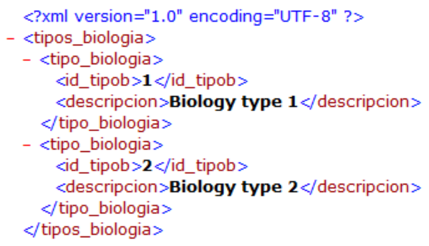
```
  <?xml version="1.0" encoding="UTF-8" ?>
- <servicios_parque>
  - <servicio_parque>
      <id_sp>1</id_sp>
      <id_gps>3</id_gps>
      <descripcion>Area 2</descripcion>
    </servicio_parque>
  - <servicio_parque>
      <id_sp>2</id_sp>
      <id_gps>4</id_gps>
      <descripcion>Restaurant</descripcion>
    </servicio_parque>
  </servicios_parque>
  <?xml version="1.0" encoding="UTF-8" ?>
- <perfiles>
  - <perfil>
      <id_perfil>1</id_perfil>
      <descripcion>Child</descripcion>
    </perfil>
  - <perfil>
      <id_perfil>2</id_perfil>
      <descripcion>Student</descripcion>
    </perfil>
  </perfiles>
```

```
  <?xml version="1.0" encoding="UTF-8" ?>
- <zonas>
  - <zona>
      <id_zona>1</id_zona>
      <descripcion>Savanna</descripcion>
    </zona>
  - <zona>
      <id_zona>2</id_zona>
      <descripcion>Equatorial Forest</descripcion>
    </zona>
  </zonas>
```

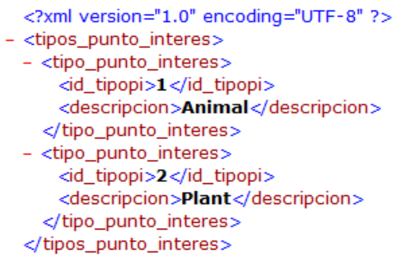
```
  <?xml version="1.0" encoding="UTF-8" ?>
- <idiomas>
  - <idioma>
      <id_idioma>1</id_idioma>
      <descripcion>Spanish</descripcion>
    </idioma>
  - <idioma>
      <id_idioma>2</id_idioma>
      <descripcion>English</descripcion>
    </idioma>
  </idiomas>
```

```
  <?xml version="1.0" encoding="UTF-8" ?>
- <tipos_audio>
  - <tipo_audio>
      <id_tipoa>1</id_tipoa>
      <descripcion>Educational</descripcion>
    </tipo_audio>
  - <tipo_audio>
      <id_tipoa>2</id_tipoa>
      <descripcion>Fighting</descripcion>
    </tipo_audio>
  </tipos_audio>
```

```
  <?xml version="1.0" encoding="UTF-8" ?>
- <tipos_video>
  - <tipo_video>
      <id_tipov>1</id_tipov>
      <descripcion>Educational</descripcion>
    </tipo_video>
  - <tipo_video>
      <id_tipov>2</id_tipov>
      <descripcion>Eating</descripcion>
    </tipo_video>
  </tipos_video>
```



```
  <?xml version="1.0" encoding="UTF-8" ?>
- <tipos_biologia>
  - <tipo_biologia>
      <id_tipob>1</id_tipob>
      <descripcion>Biology type 1</descripcion>
    </tipo_biologia>
  - <tipo_biologia>
      <id_tipob>2</id_tipob>
      <descripcion>Biology type 2</descripcion>
    </tipo_biologia>
  </tipos_biologia>
```



```
  <?xml version="1.0" encoding="UTF-8" ?>
- <tipos_punto_interes>
  - <tipo_punto_interes>
      <id_tipopi>1</id_tipopi>
      <descripcion>Animal</descripcion>
    </tipo_punto_interes>
  - <tipo_punto_interes>
      <id_tipopi>2</id_tipopi>
      <descripcion>Plant</descripcion>
    </tipo_punto_interes>
  </tipos_punto_interes>
```

Appendix 3

Basic User Tutorial

This Annex represents a useful and brief guide about the usage of this application. Ease of usage is one of the main goals of any system, and it has been kept in mind throughout all the steps in developing the application within this thesis.

The user should previously know how to use a simple iPhone mobile phone.

An icon on iPhone will appear indicating that the application is ready to be used. After starting the application, a view appears encouraging the user to select the flag corresponding to the language he prefers and an info button. What is more, the user can choose between local information (WIFI OFF) and up to date remote information (WIFI ON). See figure 62. Tapping on the info button, will result in displaying information about the thesis. See figure 63.
Next a welcome view appears. The user must tap the profile that best fits him. The information presented will vary depending on which profile he has chosen.
Then the different Bioparc areas are displayed. The user should tap the area he is particularly interested in. The GPS option will be further commented.

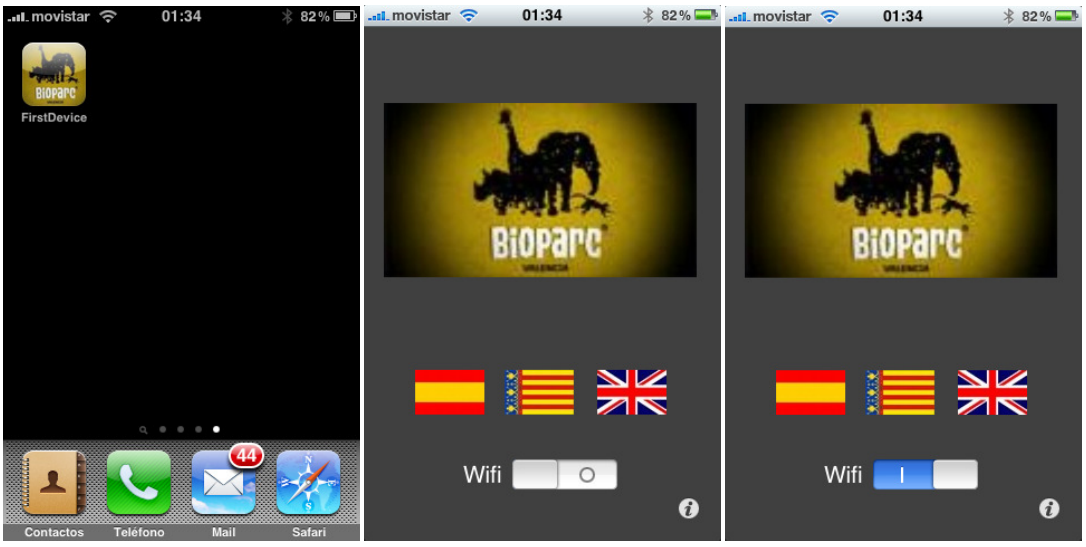


Figure 62. Icon, language and WIFI option.

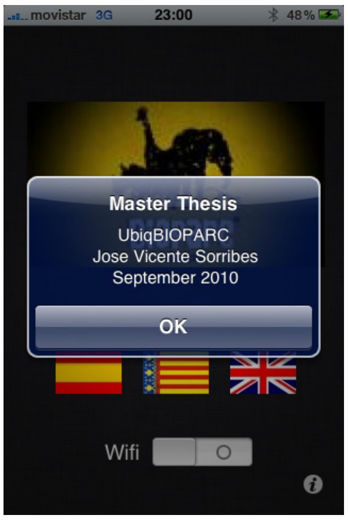


Figure 63. Application's Info message.

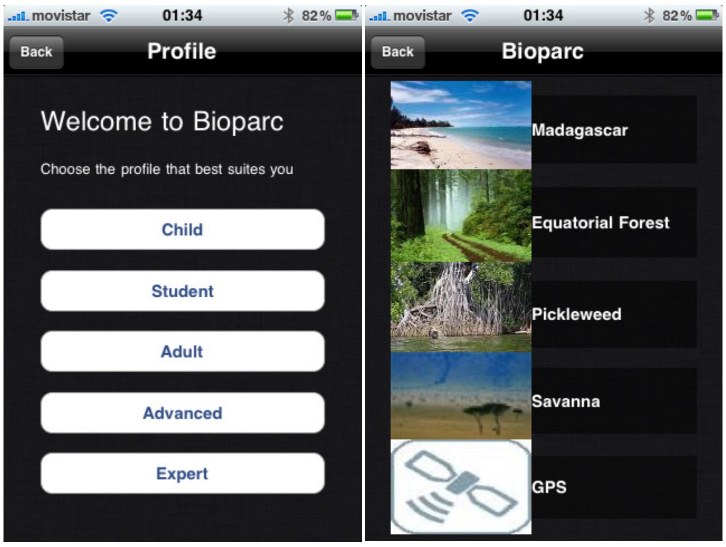


Figure 64. Profile, areas and GPS options.

All the preference settings have already done. So the information the application will show will depend on those preferences.
The next interface shows a list of animals with a photo, the name and a brief description. The user can change to the list of other points of interest such as plants or landscapes by tapping the corresponding button below.
The lists are scrolling enabled so that the user can view more animals, plants or landscapes by scrolling those lists up or down.

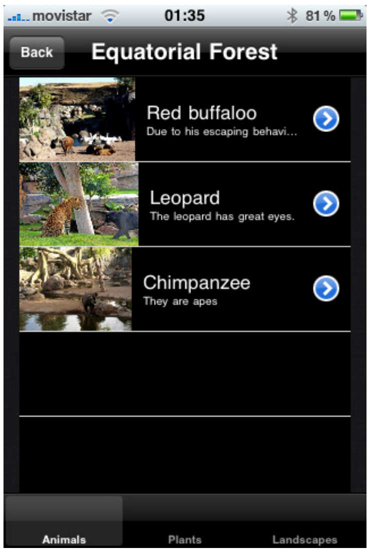
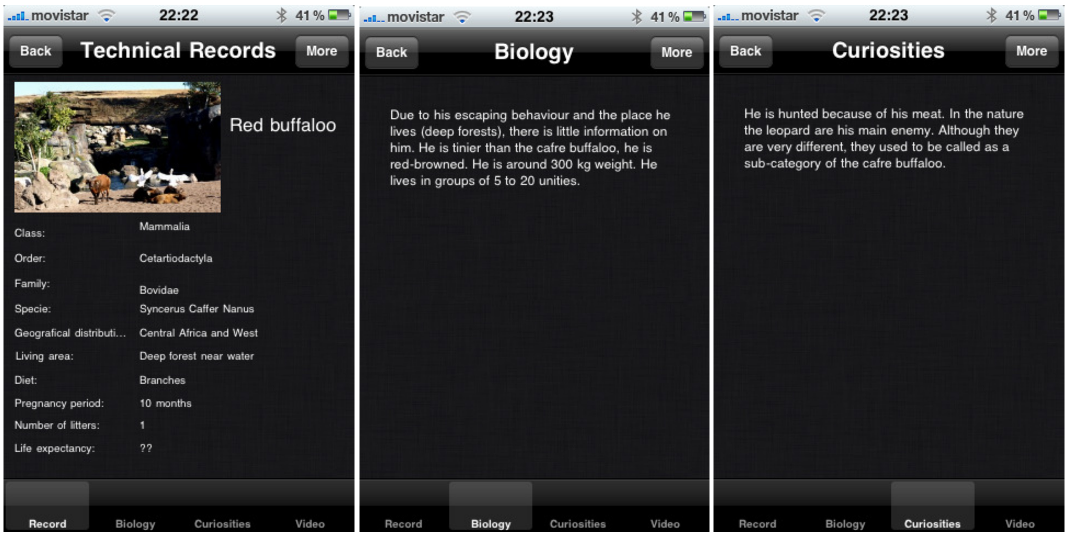


Figure 65. Points of interest list.

After tapping a particular point of interest, a new view with information about it will appear in different sections. Depending on the type of point of interest selected, that is animal, plant or landscape, the number of sections could vary.
The animal information is the most complete one. This new interface is composed of 4 different buttons below it, and a "more" button above. The first one leads us to the Technical Record view. The rest of the buttons are linked to related information gaps like Biology, Curiosities, or Video information.
So if you tap a button, the corresponding information will appear.
The Technical Record view shows the name, a photo and information distributed among different fields in a typical record. Biology and Curiosities information can also be seen. The user can navigate through the Biology or Curiosities text by scrolling it up or down. All this information will vary depending on the preferences (language and profile) the user fixed in the application's first two views.



Red buffaloo

Class:	Mammalia
Order:	Cetartiodactyla
Family:	Bovidae
Specie:	Syncerus Caffer Nanus
Geografical distributi...	Central Africa and West
Living area:	Deep forest near water
Diet:	Branches
Pregnancy period:	10 months
Number of litters:	1
Life expectancy:	??

Due to his escaping behaviour and the place he lives (deep forests), there is little information on him. He is tinier than the cafre buffaloo, he is red-browned. He is around 300 kg weight. He lives in groups of 5 to 20 unities.

He is hunted because of his meat. In the nature the leopard are his main enemy. Although they are very different, they used to be called as a sub-category of the cafre buffaloo.

Figure 66. Technical record, biology and curiosities view.

If you tap on the Video button, a list of videos will appear, indicating a title and a brief description. Once more, the user can navigate through the list of videos by scrolling up or down.
When the user chooses a video, it will start playing. He will be able to pause it, go forwards or backwards on time, or stop it. Tapping the *Done* button the video will stop and the application will go back to the list of videos view.

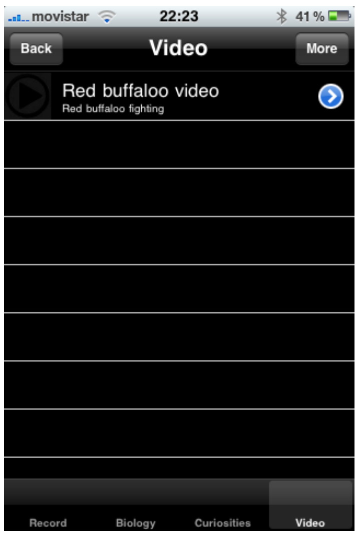


Figure 67. List of videos.

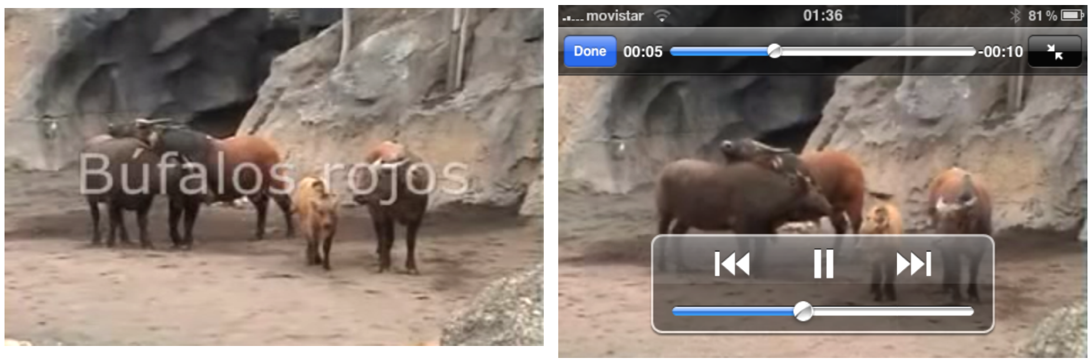


Figure 68. Video player.

If you choose to tap on the "more" button, two more views will appear: audio and photos. Tapping on the Audio button, a list of audio files will appear including the title and a brief description on each one. Once more, the list allows the user to scroll up or down. When the user taps an audio file, it will start playing. Tapping again will cause stopping it. The user can start and/or stop it all the times he desires. See figure 69.

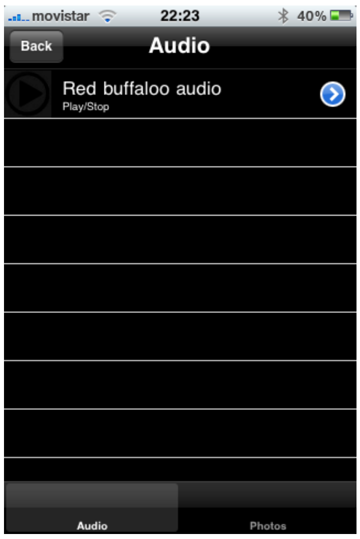


Figure 69. List of audio files.

Photo option is also available, displaying a photo list containing titles and brief descriptions. After selecting one of them, the photo can be seen. See figure 70.

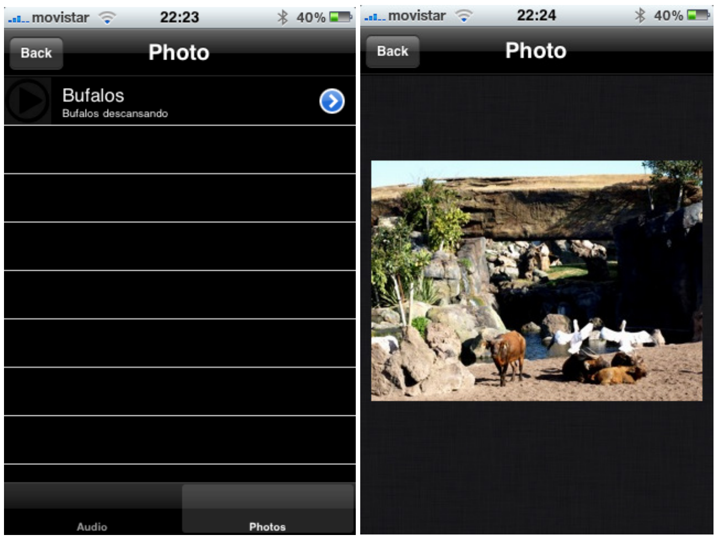
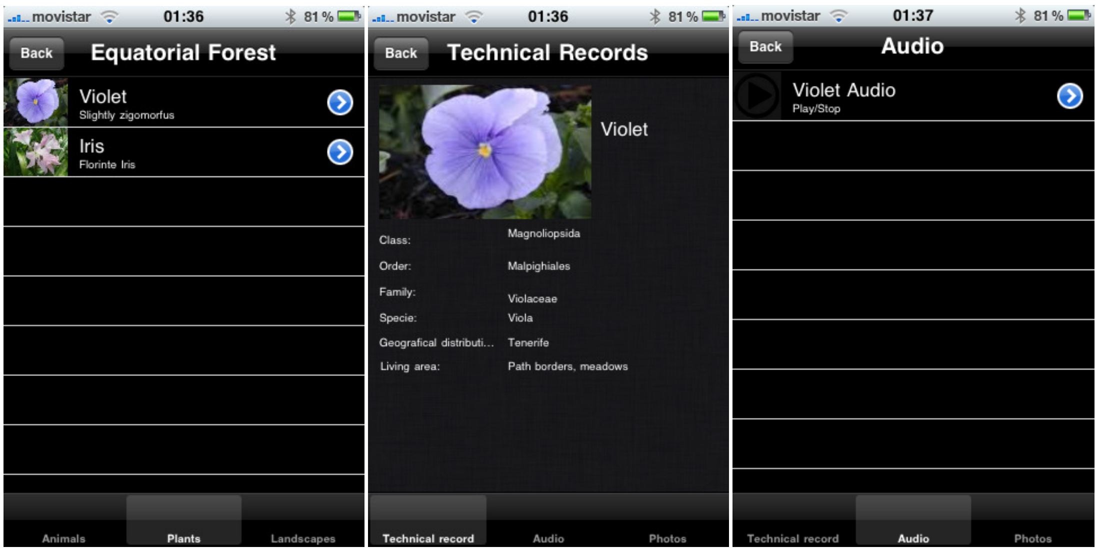


Figure 70. Photo lists and photo show view.

If the user is particularly keen on plants or landscapes, similar information is available. Choosing the plants option the multimedia information available would be: photos, audio, and a technical record section consisting of name, description, class, order, family, specie, geographical distribution and environment fields.
If the user chooses the landscapes option, the only sections that appear are photos, video and audio.
For both of these options, the usage is similar to the Animals option.



Class:	Magnoliopsida
Order:	Malpighiales
Family:	Violaceae
Specie:	Viola
Geografical distributi...	Tenerife
Living area:	Path borders, meadows

Figure 71. List of plants, technical record, and audio.

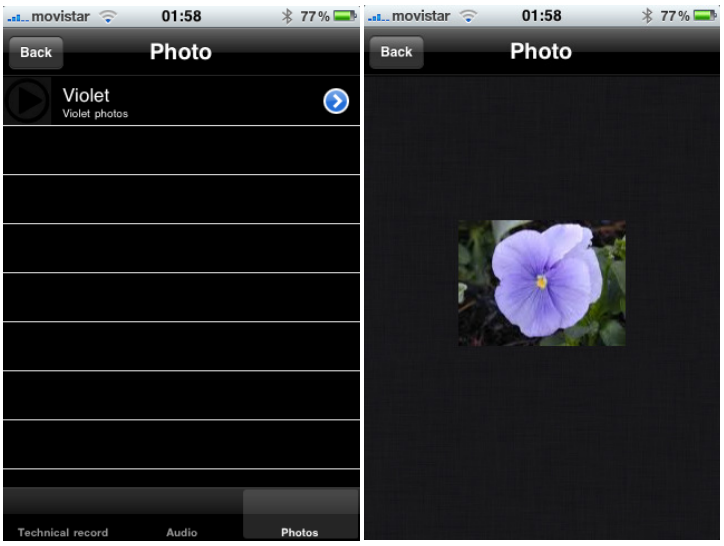


Figure 72. Plant photo option.

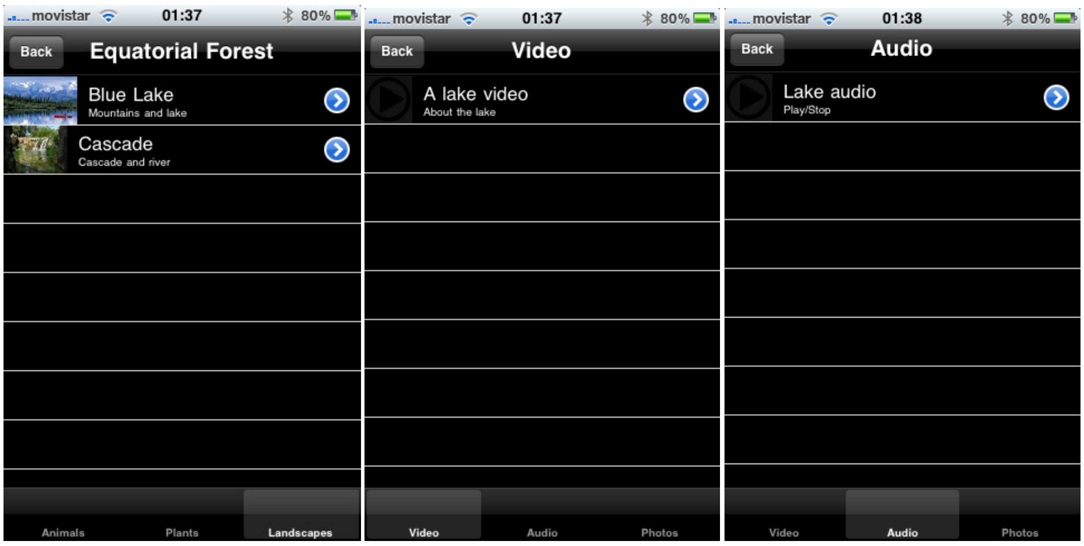


Figure 73. List of landscapes, video and audio.

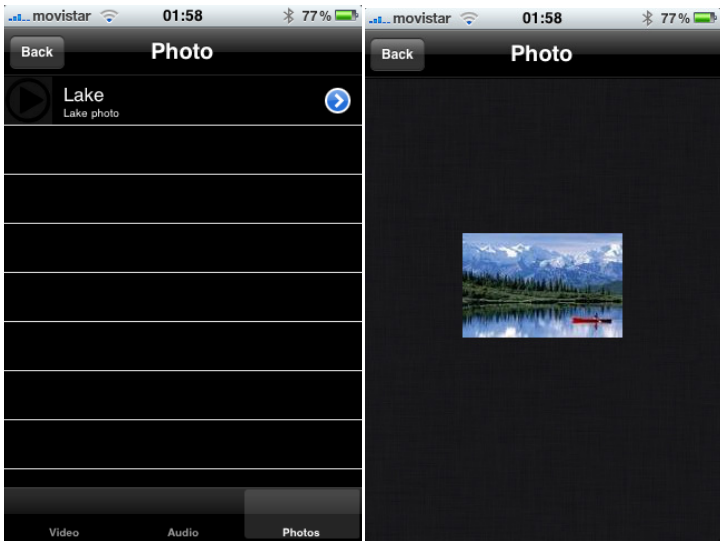


Figure 74. Landscape photo option.

A more advanced option is available by tapping the GPS button in the following interface

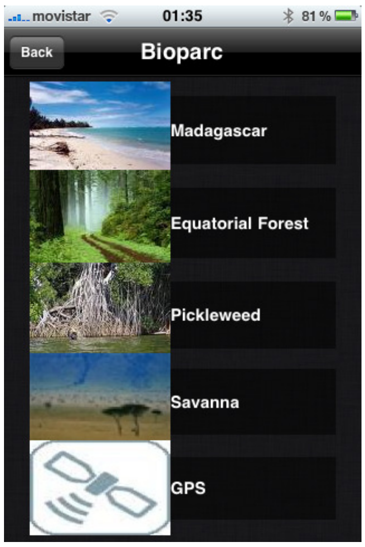


Figure 75. GPS option.

This option allows users to search for animals around the park, indicating their position in a GPS map and the distance to them. Furthermore the nearest animal to the user can also be displayed in another map indicating the distance to it.
The GPS searcher option will appear when GPS searcher button is selected. Then the user should tap on the *Search* button. A new interface will appear. It contains a search field and a list of animals. The user can just tap the animal he is interested in or tap the search field. A contextual keyboard will appear helping the user introduce the animal's name. As he introduces letters the list will be filtered showing only the animals whose name starts with the group of letters he introduced. Finally, the user must tap the animal or tap *Search* button. See figures 76 and 77.

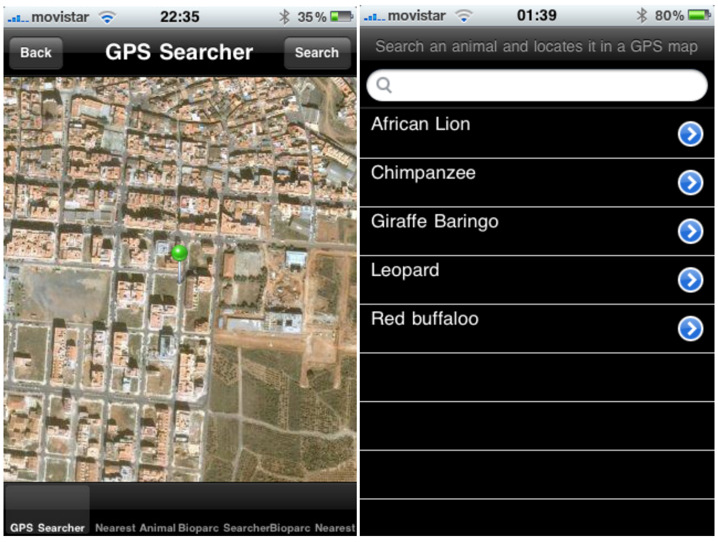


Figure 76. GPS Searcher (I).

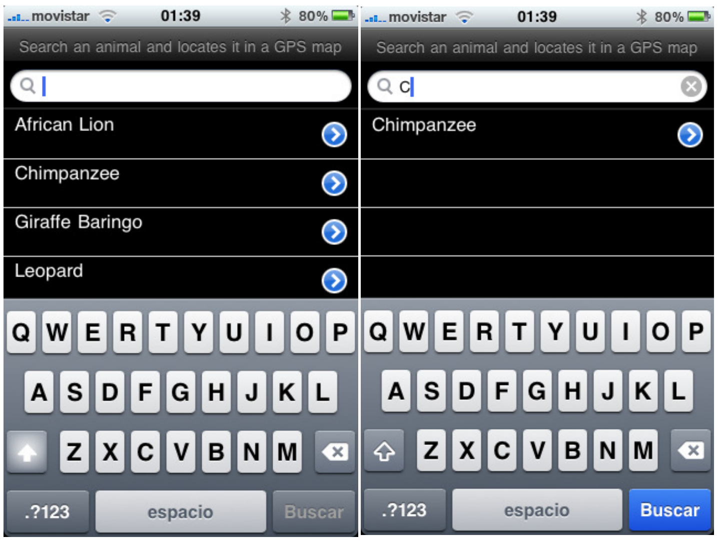


Figure 77. GPS Searcher (II).

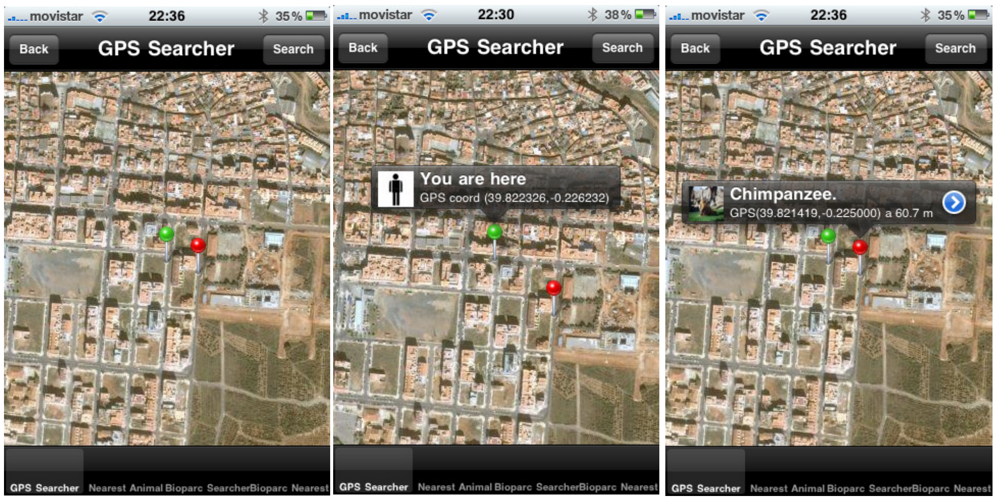
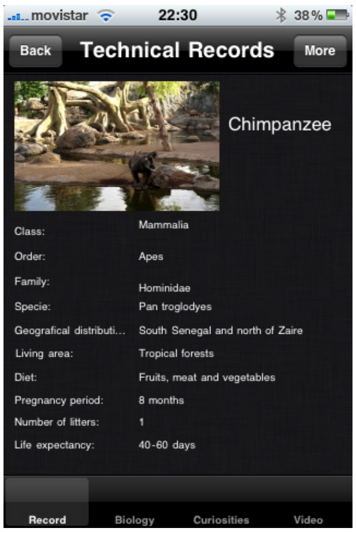

movistar 22:36 35 %
Back
GPS Searcher
Search
movistar 22:30 38 %
You are here
GPS coord (39.822326,-0.226232)
movistar 22:36 35 %
Chimpanzee.
GPS(39.821419,-0.225000) a 60.7 m
GPS Searcher Nearest Animal Bioparc SearcherBioparc Nearest

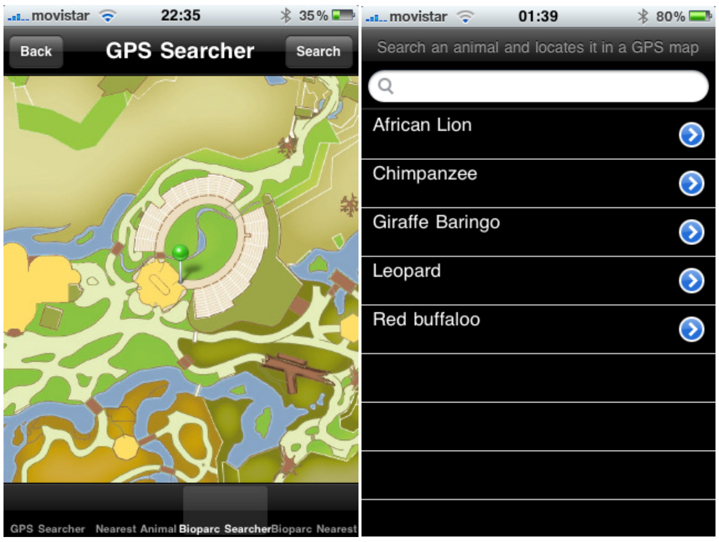
Figure 78. GPS Searcher (III).



Chimpanzee

Class:	Mammalia
Order:	Apes
Family:	Hominidae
Specie:	Pan troglodyes
Geografical distributi...	South Senegal and north of Zaire
Living area:	Tropical forests
Diet:	Fruits, meat and vegetables
Pregnancy period:	8 months
Number of litters:	1
Life expectancy:	40-60 days

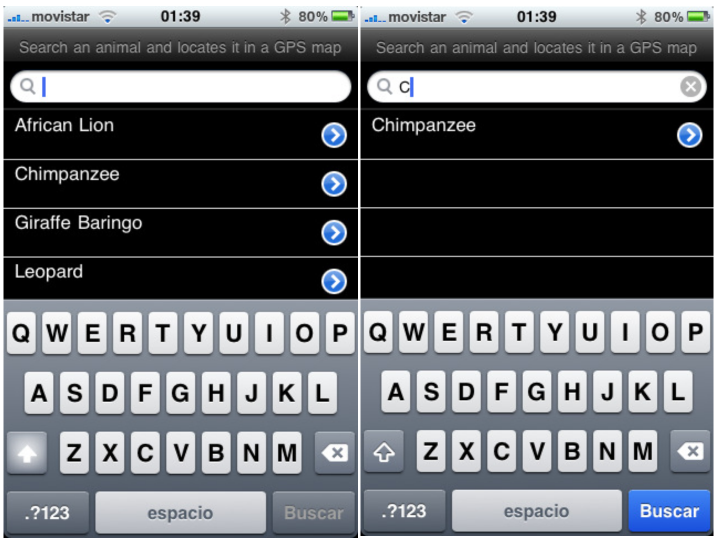
Figure 79. GPS Searcher (IV).



Search an animal and locates it in a GPS map

African Lion
Chimpanzee
Giraffe Baringo
Leopard
Red buffaloo

Figure 80. GPS custom map Searcher (I).



Search an animal and locates it in a GPS map

African Lion
Chimpanzee
Giraffe Baringo
Leopard

Search an animal and locates it in a GPS map

c

Chimpanzee

Figure 81. GPS custom map Searcher (II).

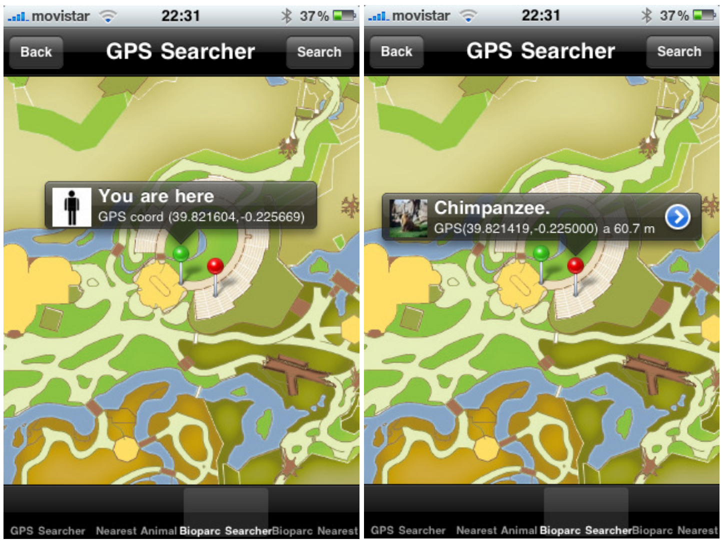
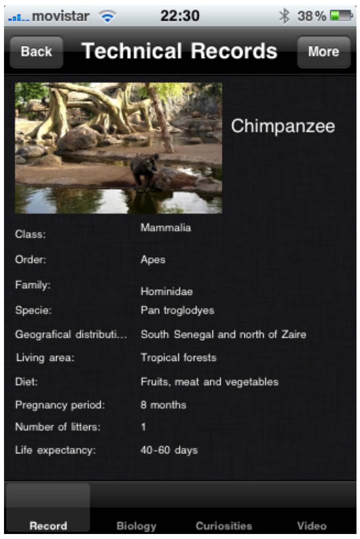


Figure 82. GPS custom map Searcher (III).



Chimpanzee

Class:	Mammalia
Order:	Apes
Family:	Hominidae
Specie:	Pan troglodyes
Geografical distributi...	South Senegal and north of Zaire
Living area:	Tropical forests
Diet:	Fruits, meat and vegetables
Pregnancy period:	8 months
Number of litters:	1
Life expectancy:	40-60 days

Figure 83. GPS custom Searcher (IV).

The result will be a full-custom satellite GPS map with two pins. The green one indicates the user's position. When it is tapped a "You are here" message and a user icon will appear. The red pin corresponds to the searched animal position. If the user taps on it, the animal's name and the distance to it will appear. See figure 78. By tapping the arrow button, the view of information on this animal will be displayed, allowing the users to navigate through the Technical record, Biology, Curiosities, Video and Audio views, as it was pointed out before. See figure 79. What is more, the user can scroll the map or zoom in or out.
The user has the option to use the same utility viewing a park custom map instead of a google map. See figures 80, 81, 82 and 83.

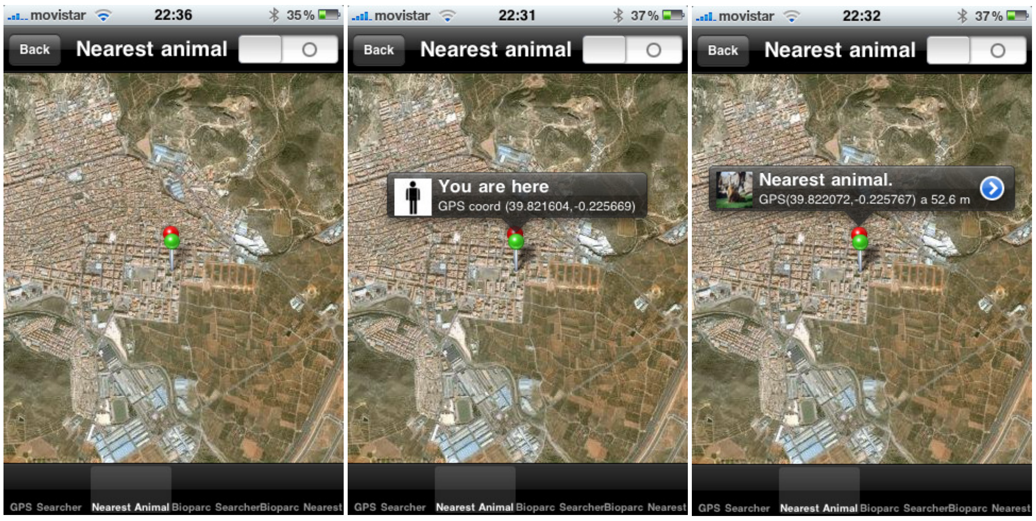
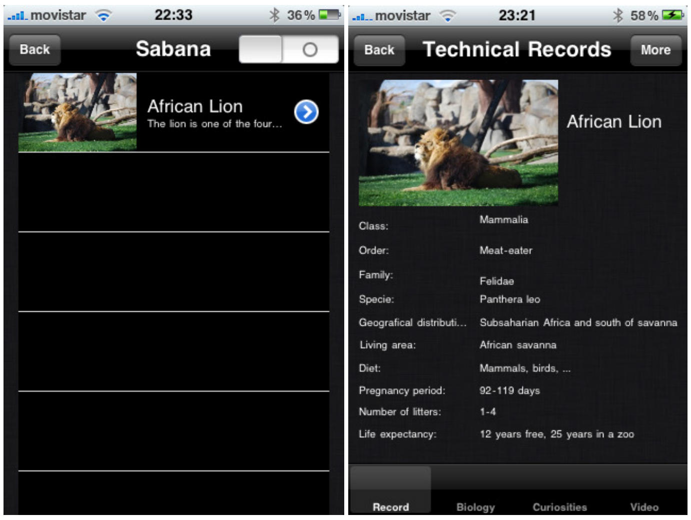


Figure 84. Nearest animal map.



Class:	Mammalia
Order:	Meat-eater
Family:	Felidae
Specie:	Panthera leo
Geografical distributi...	Subsaharian Africa and south of savanna
Living area:	African savanna
Diet:	Mammals, birds, ...
Pregnancy period:	92-119 days
Number of litters:	1-4
Life expectancy:	12 years free, 25 years in a zoo

Figure 85. Nearest animal list and technical record.

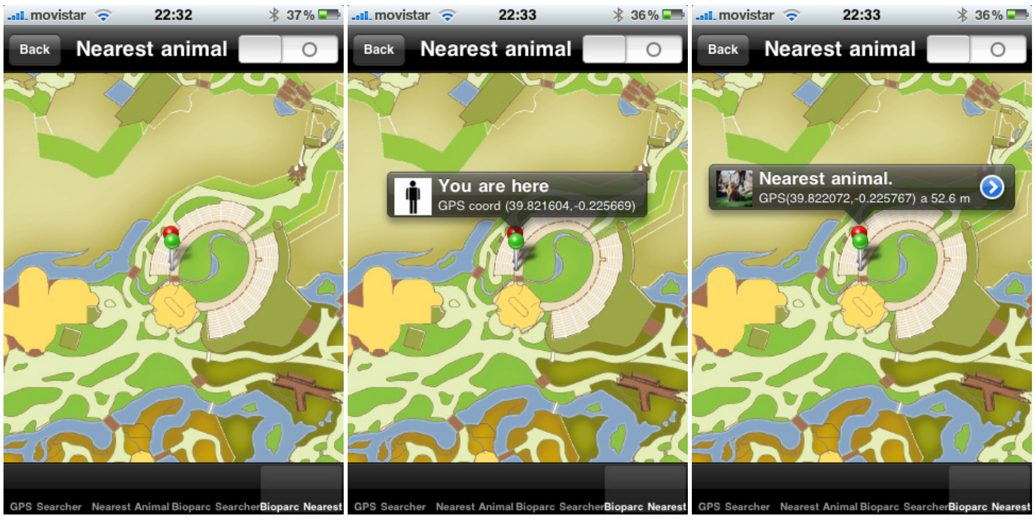
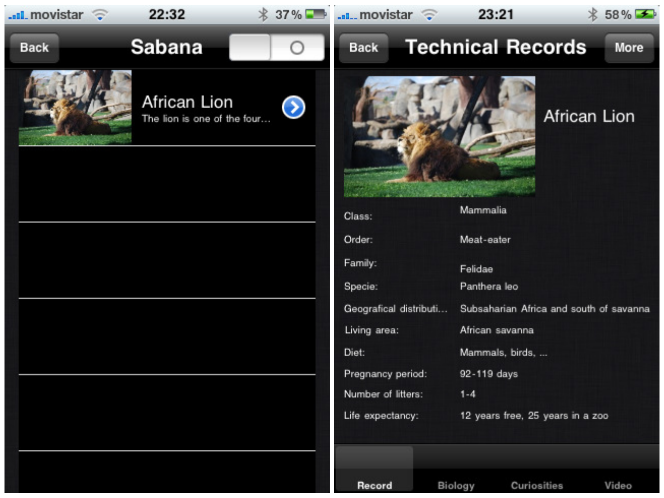


Figure 86. Nearest animal custom map.



Class:	Mammalia
Order:	Meat-eater
Family:	Felidae
Specie:	Panthera leo
Geografical distributi...	Subsaharian Africa and south of savanna
Living area:	African savanna
Diet:	Mammals, birds, ...
Pregnancy period:	92-119 days
Number of litters:	1-4
Life expectancy:	12 years free, 25 years in a zoo

Figure 87. Nearest animal list and technical record.

If the user wants to try the Nearest animal option, he should tap on the corresponding button.
The view will automatically show a satellite map and two pins once more. The green pin indicates the user's position. When he taps on it, a "You are here" message and a user icon will be displayed. The red pin indicates the nearest animal position. When tapping on it, a "Nearest animal" message and the distance to it will appear. See figure 84. If the user taps the arrow button, a list of the nearest animals group will be displayed. This list will change depending on the nearest animal region at that particular moment. The user can tap one of them to see its Technical Record, Biology options, and so on. See figure 85.

Once more, a park custom map is available within this utility. See figures 86 and 87.

The coolest option is integrated in the Nearest Animal option. It is the context-aware option. When the user is less than a few metres long from the nearest animal region, it automatically shows the list interface. The most surprising thing is that he will see on the device the same list of animals that he is physically looking at in the park. When the user moves or changes its orientation, the list will automatically adapt its contents to the animals he is looking at, without the users intervention.

An example of the functionality is the following.
Say the user is standing at the center of a path which has two areas at both sides of it. To the left he has monkeys and gorillas. To the right he has lions and tigers. If the user turns left to see the animals, a list with monkeys and gorillas will appear. If he turns right, a list with lions and tigers will automatically be displayed, without his intervention.
What is more, if he goes on walking towards a giraffes and impalas area and he is near enough, the interface will automatically show a list with giraffe and impalas.

The most amazing point is that the user does not have to do anything, just walk and enjoy the application.

To enable this mode, the user should choose the switch ON option in the Nearest Animal interface or in the List of Animals interface. See figure 88 and 89.
If the user chooses this option in the Nearest Animal interface and taps on the arrow button (see figure 88), a list of animals will be shown whose contents will depend on the user's orientation. Figure 89 shows the results, when the user points to an area with an African Lion and when the user points to an area without animals.
Once more, this option could be chosen by selecting the switch in the park custom map. See figure 90.

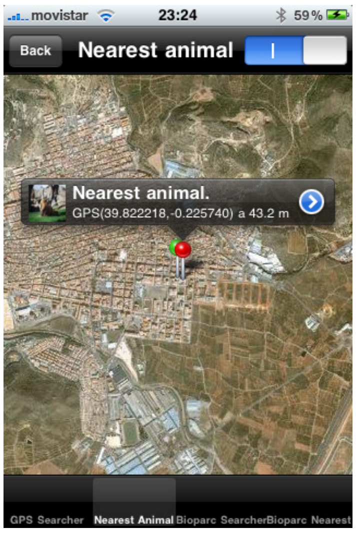


Figure 88. Orientation enabled.

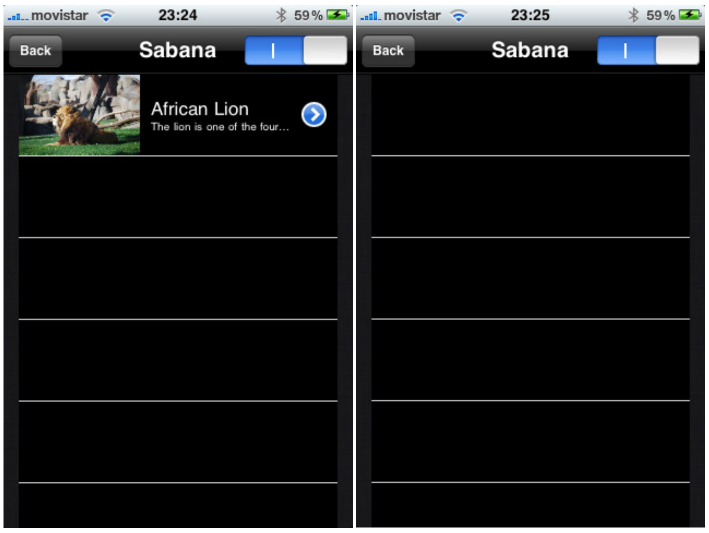


Figure 89. Context-aware results (an area containing an African Lion and an area containing nothing).

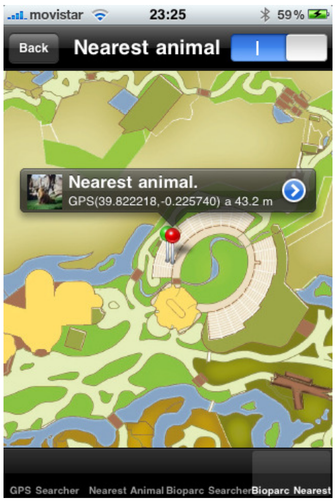


Figure 90. Orientation enabled in custom map.

Appendix 4

Steps to install the iPhone SDK IDE

iPhone SDK is the IDE used to develop applications for iPhone device.

First of all, we must install the iPhone SDK IDE. We must own a Mac computer, and have Mac OSX 10.5.7 or later installed.

In this thesis, iPhone SDK 3.1 is used. It requires Mac OSX 10.5.7 Leopard or later.
Once we have the computer and the operating system installed, we ought to download iPhone SDK installer package from Apple's website.
It consists of a .dmg file that must be double-clicked to begin the installation process.
After double-clicking on it, a new window will appear, as it is shown in the following screenshot.

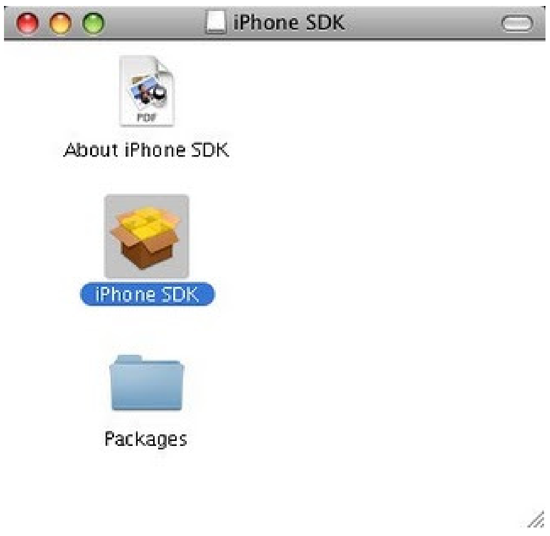
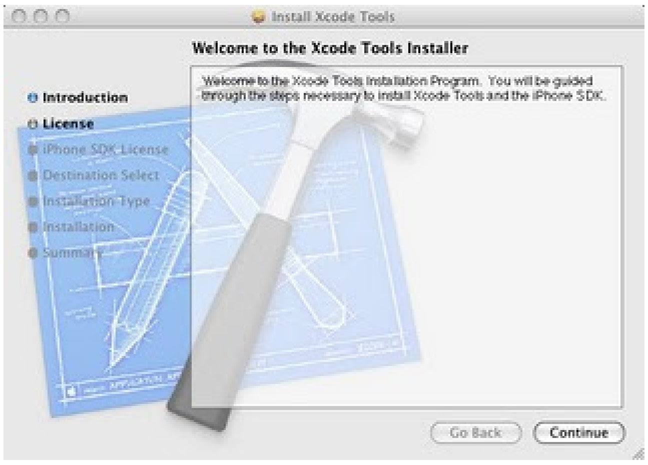


Figure 91. The .dmg package.

Then the developer must double-click the iPhone SDK package.

A wizard will start installing the environment.
A welcome window will appear. See the following screenshot. Next, we must press the *Continue* button.

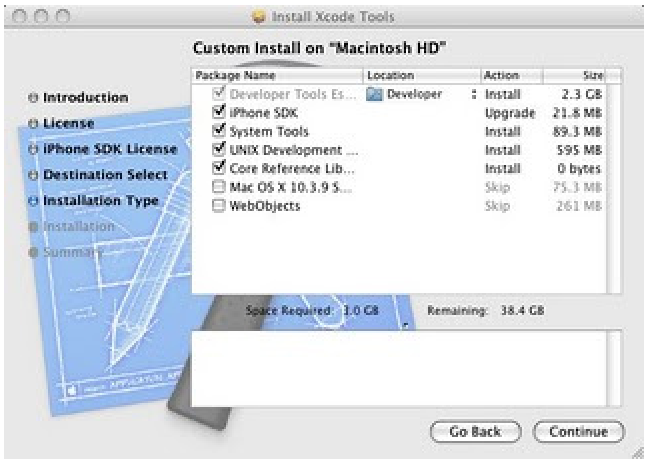


Welcome to the Xcode Tools Installer

Welcome to the Xcode Tools Installation Program. You will be guided through the steps necessary to install Xcode Tools and the iPhone SDK.

Figure 92. The wizard.

Next a new window will appear that allows us to choose a custom install. The appearance of these options depend on the iPhone SDK version we want to install. A typical installation screenshot is the following.



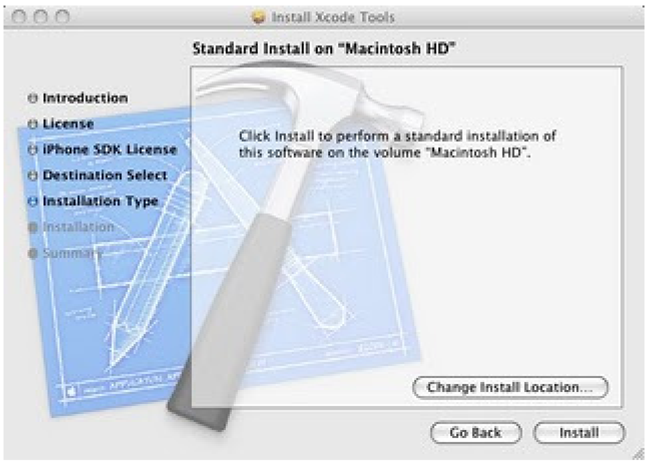
Custom Install on "Macintosh HD"

Package Name	Location	Action	Size
✓ Developer Tools Es...	Developer	Install	2.3 GB
✓ iPhone SDK		Upgrade	21.8 MB
✓ System Tools		Install	89.3 MB
✓ UNIX Development ...		Install	595 MB
✓ Core Reference Lib...		Install	0 bytes
☐ Mac OS X 10.3.9 S...		Skip	75.3 MB
☐ WebObjects		Skip	261 MB

Space Required: 3.0 GB Remaining: 38.4 GB

Figure 93. Custom install.

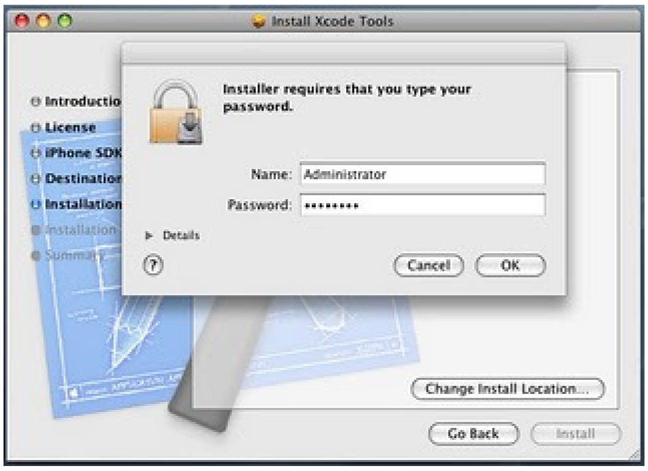
You should at least select *Developer Tools Essential* and *iPhone SDK* options for the environment to work, but you may select any of the others additionally. After clicking on *Continue*, a new window will appear giving information on the installation the wizard is going to start. Just click on the *Install* button



Click Install to perform a standard installation of this software on the volume "Macintosh HD".

Figure 94. Standard install.

Sometimes the wizard may require you to enter the *Name* and *Password.* After entering the required information, click *OK* and *Install*

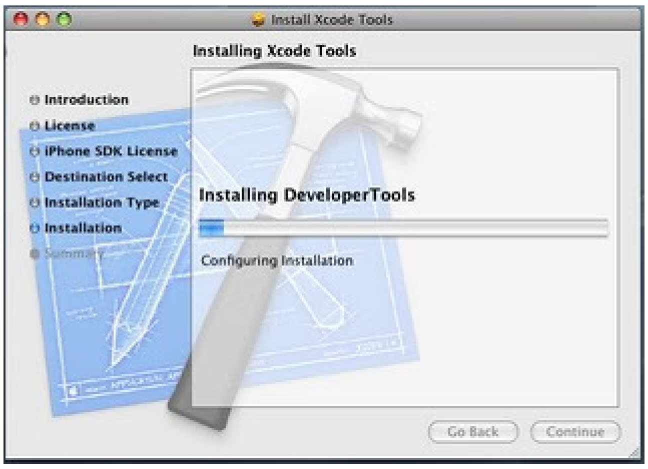


Installer requires that you type your password.

Name: Administrator
Password: ••••••••

Figure 95. Installer requiring password.

The installation process will begin.

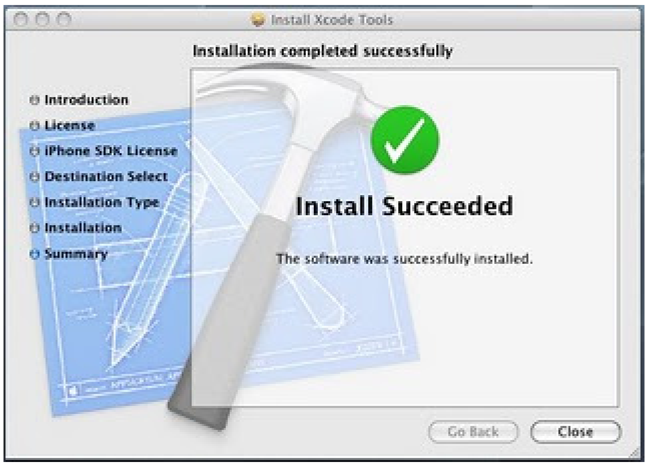


Installing DeveloperTools

Configuring Installation

Figure 96. Installation in progress.

This process may take about 3 hours or more, depending on the version. So be patient. Finally, when the installation is completed successfully, a message appears indicating so.



Install Succeeded

The software was successfully installed.

Figure 97. Installation completed successfully.

Click the *Close* button and try to run Xcode, Interface Builder and iPhone Simulator to check if the installation was really successful.

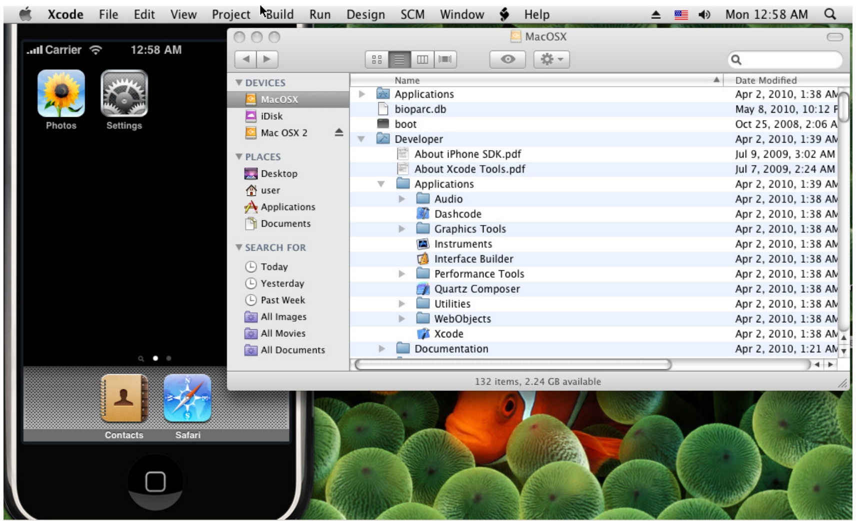
Appendix 5

A brief development guide: Building a simple Application with iPhone SDK

In this Annex, the basic steps to develop a simple application using iPhone SDK are shown.

To be precise, among all the possible applications, a simple app in which the user will be able to press a button in the screen and the device replies with a message has been chosen.

1. In his first step, the developer must start the IDE, double-clicking XCode application, Interface Builder and iPhone Simulator.
   Xcode is the environment used to manage a project, that is, the different files that a project consists of and the different multimedia items as images or videos, and allows the programmer to develop the code using Objective-C language.
   Interface Builder is the tool by which the interfaces are designed.
   Between these two programs there are dependencies, as the interfaces contain objects that are managed by the Xcode code, and Xcode contains .xib (or .nib) files corresponding to the interfaces.
   Besides, iPhone simulator is a tool that allows developers to simulate the functionality of our application in the computer, not in the device. It is useful to debug and run the application, before sending it to the iPhone device.
   Xcode and Interface Builder are usually found in the */Developer/Applications* directory, as it is shown in the following figure.

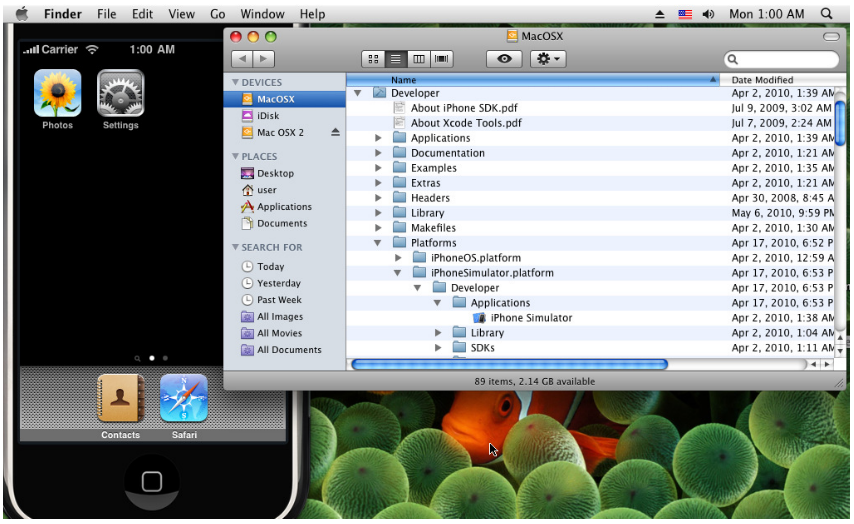


Name	Date Modified
Applications	Apr 2, 2010, 1:38 AM
bioparc.db	May 8, 2010, 10:12
boot	Oct 25, 2008, 2:06 A
Developer	Apr 2, 2010, 1:39 AM
About iPhone SDK.pdf	Jul 9, 2009, 3:02 AM
About Xcode Tools.pdf	Jul 7, 2009, 2:24 AM
Applications	Apr 2, 2010, 1:39 AM
Audio	Apr 2, 2010, 1:38 AM
Dashcode	Apr 2, 2010, 1:38 AM
Graphics Tools	Apr 2, 2010, 1:38 AM
Instruments	Apr 2, 2010, 1:38 AM
Interface Builder	Apr 2, 2010, 1:38 AM
Performance Tools	Apr 2, 2010, 1:38 AM
Quartz Composer	Apr 2, 2010, 1:38 AM
Utilities	Apr 2, 2010, 1:38 AM
WebObjects	Apr 2, 2010, 1:38 AM
Xcode	Apr 2, 2010, 1:38 AM
Documentation	Apr 2, 2010, 1:21 AM

132 items, 2.24 GB available

Figure 98. Xcode and Interface Builder location.

iPhone Simulator is found in the */Developer/Platforms/iPhoneSimulator.platform/Developer/Applications* directory.

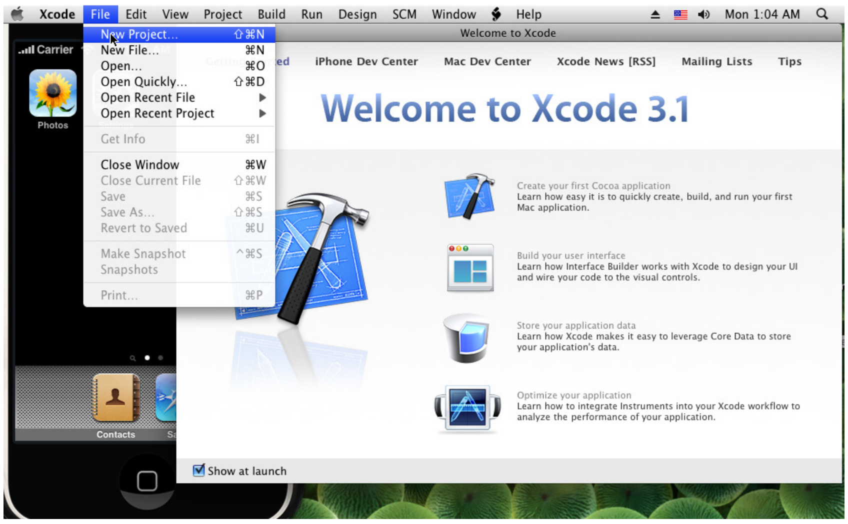


Name	Date Modified
Developer	Apr 2, 2010, 1:39 AM
About iPhone SDK.pdf	Jul 9, 2009, 3:02 AM
About Xcode Tools.pdf	Jul 7, 2009, 2:24 AM
Applications	Apr 2, 2010, 1:39 AM
Documentation	Apr 2, 2010, 1:21 AM
Examples	Apr 2, 2010, 1:35 AM
Extras	Apr 2, 2010, 1:21 AM
Headers	Apr 30, 2008, 8:45 A
Library	May 6, 2010, 9:59 PM
Makefiles	Apr 2, 2010, 1:30 AM
Platforms	Apr 17, 2010, 6:52 P
iPhoneOS.platform	Apr 2, 2010, 12:59 A
iPhoneSimulator.platform	Apr 17, 2010, 6:53 P
Developer	Apr 17, 2010, 6:53 P
Applications	Apr 17, 2010, 6:53 P
iPhone Simulator	Apr 2, 2010, 1:38 AM
Library	Apr 2, 2010, 1:04 AM
SDKs	Apr 2, 2010, 1:11 AM

89 items, 2.14 GB available

Figure 99. iPhone simulator location.

2. Then, the developer must create a new project clicking *File-New Project.*



Create your first Cocoa application
Learn how easy it is to quickly create, build, and run your first Mac application.

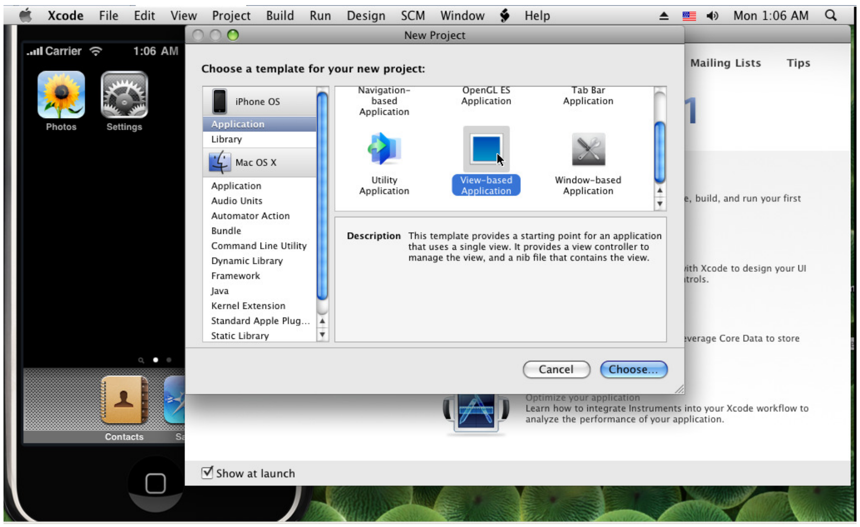
Build your user interface
Learn how Interface Builder works with Xcode to design your UI and wire your code to the visual controls.

Store your application data
Learn how Xcode makes it easy to leverage Core Data to store your application's data.

Optimize your application
Learn how to integrate Instruments into your Xcode workflow to analyze the performance of your application.

Figure 100. New Project creation.

The *View-based Application* option should be selected, in this occasion.

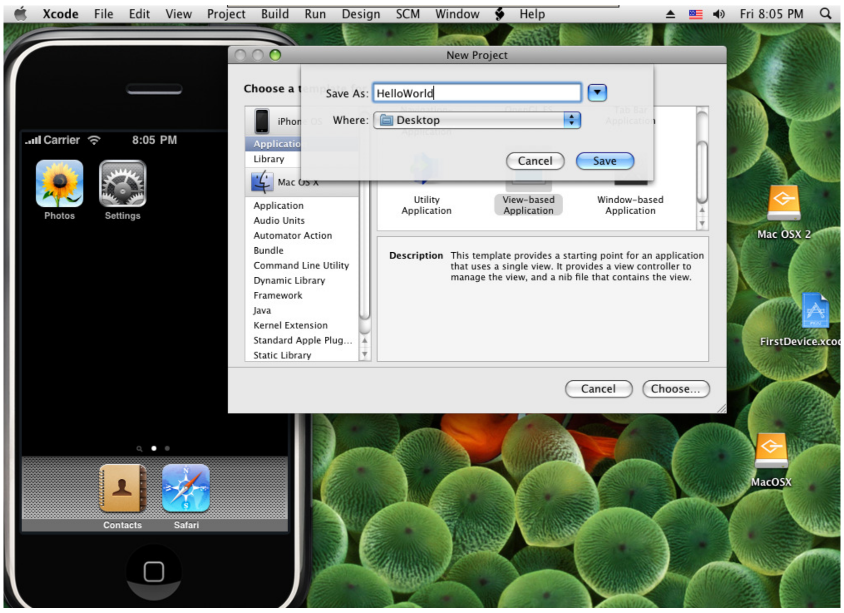


Choose a template for your new project:

Description This template provides a starting point for an application that uses a single view. It provides a view controller to manage the view, and a nib file that contains the view.

Figure 101. View-based application.

Then the *Choose* button must be pressed.
And a name should be introduced. In this case, the name given is *HelloWorld*.
In the *Where* option, the place where the project will be saved is chosen. At this moment, we can save it in the *Desktop* and press *Save*.



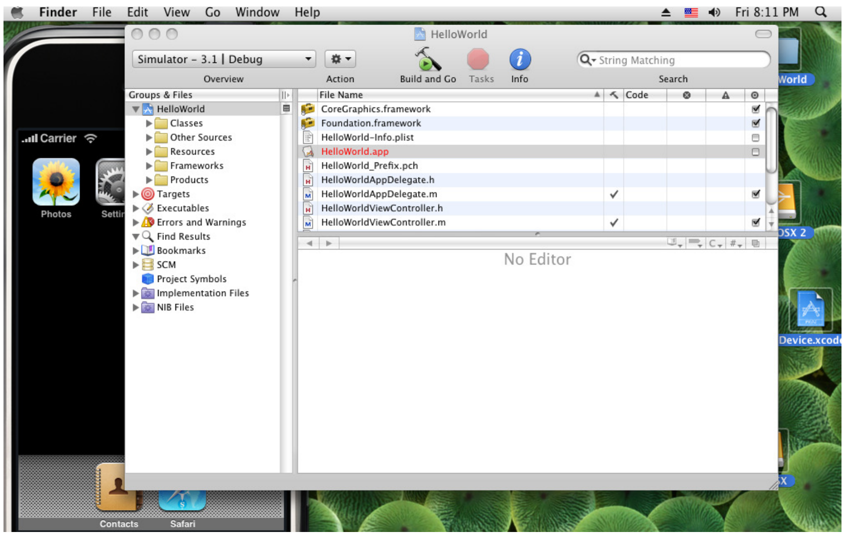
Save As: HelloWorld
Where: Desktop

Description This template provides a starting point for an application that uses a single view. It provides a view controller to manage the view, and a nib file that contains the view.

Figure 102. New project name.

Finally *Choose* must be pressed.

The Xcode environment begins with the name of the Project that we established *HelloWorld* and a set of folders at the left side that contain the project files. Among these folders we will use the *Classes* folder, where 4 files contain the project code, and *Resources,* where the *.xib* (or *.nib*) files are located. .xib files contain the interfaces.

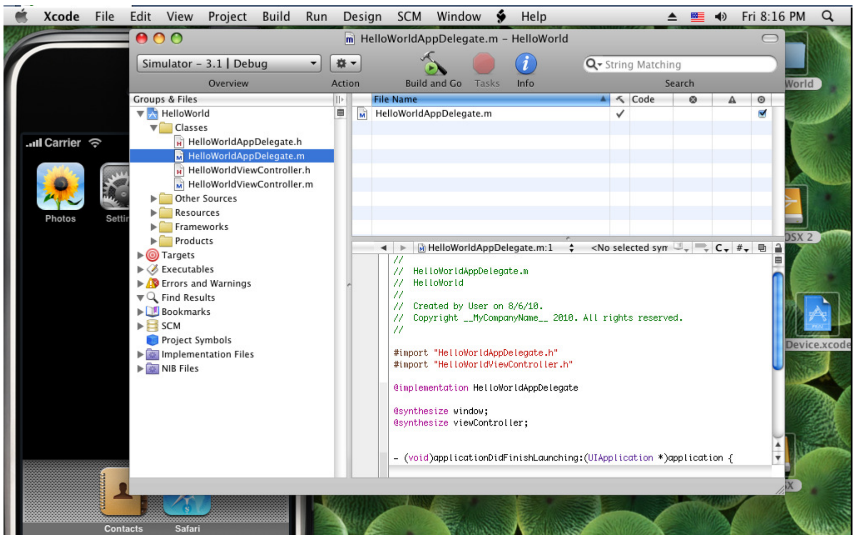


Groups & Files
- HelloWorld
  - Classes
  - Other Sources
  - Resources
  - Frameworks
  - Products
- Targets
- Executables
- Errors and Warnings
- Find Results
- Bookmarks
- SCM
- Project Symbols
- Implementation Files
- NIB Files

File Name	Code
CoreGraphics.framework	
Foundation.framework	
HelloWorld-Info.plist	
HelloWorld.app	
HelloWorld_Prefix.pch	
HelloWorldAppDelegate.h	
HelloWorldAppDelegate.m	✓
HelloWorldViewController.h	
HelloWorldViewController.m	✓

Figure 103. Xcode environment.

3. If we expand the *Classes* folder, we will notice that iPhone SDK has generated 4 files automatically.



```
//
//  HelloWorldAppDelegate.m
//  HelloWorld
//
//  Created by User on 8/6/10.
//  Copyright __MyCompanyName__ 2010. All rights reserved.
//

#import "HelloWorldAppDelegate.h"
#import "HelloWorldViewController.h"

@implementation HelloWorldAppDelegate

@synthesize window;
@synthesize viewController;


- (void)applicationDidFinishLaunching:(UIApplication *)application {
```

Figure 104. Files automatically generated.

These files are:

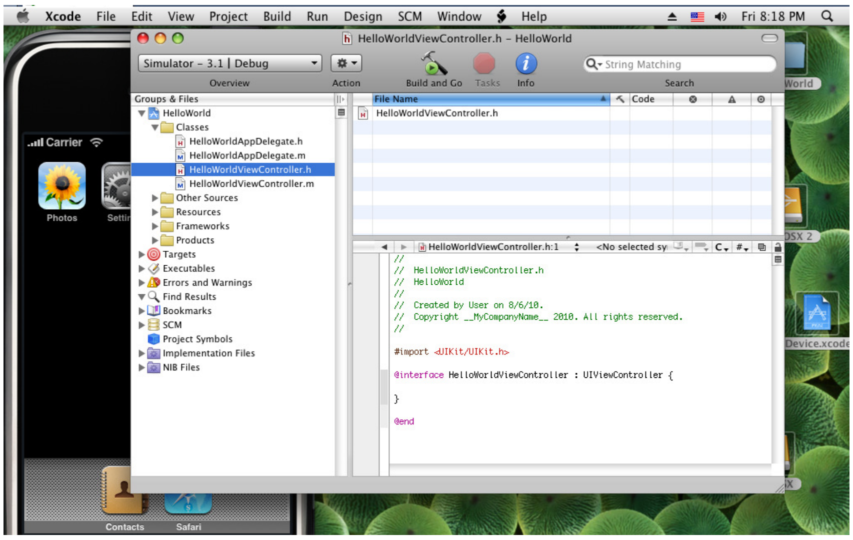
- *HelloWorldAppDelegate.h*
- *HelloWorldAppDelegate.m*
- *HelloWorldViewController.h*
- *HelloWorldViewController.m*

The first two ones won't be changed at the moment, and they contain the management of the interface window and the controller.

Next two ones contain the interface controller; this interface is the one we are developing, that is, they will specify the functionality of the objects that we have in the interface and the actions that will be done. Among these actions, pressing the button and updating the text will be considered in this application.

The *.h* files contain the interface and the *.m* files contain the implementation.

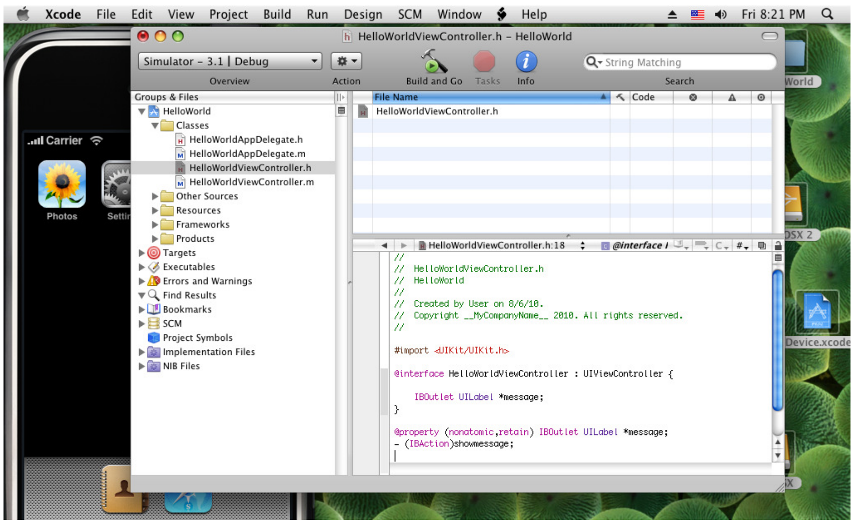
4. We should modify both files *HelloWorldViewController* to implement our application.
   First of all, we will select the file *HelloWorldViewController.h*



```
//
//  HelloWorldViewController.h
//  HelloWorld
//
//  Created by User on 8/6/10.
//  Copyright __MyCompanyName__ 2010. All rights reserved.
//

#import <UIKit/UIKit.h>

@interface HelloWorldViewController : UIViewController {

}

@end
```

Figure 105. HelloWorldViewController.h.

And we add the following code in the right side of the window:



```
//
//  HelloWorldViewController.h
//  HelloWorld
//
//  Created by User on 8/6/10.
//  Copyright __MyCompanyName__ 2010. All rights reserved.
//

#import <UIKit/UIKit.h>

@interface HelloWorldViewController : UIViewController {

    IBOutlet UILabel *message;
}

@property (nonatomic,retain) IBOutlet UILabel *message;
- (IBAction)showmessage;
```

Figure 106. Adding lines of code.

The line

*IBOutlet UILabel *message*

indicates us that in the interface we will have a label where the message will be shown.

The line

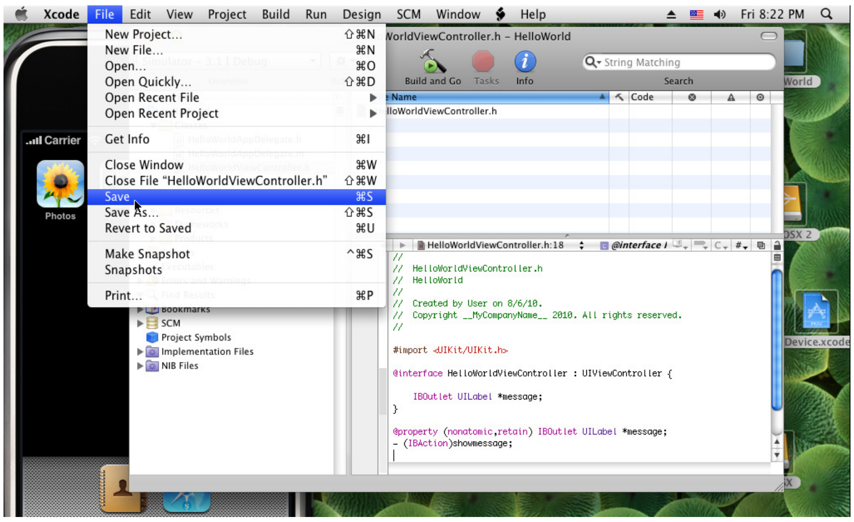
*@property (nonatomic,retain) IBOutlet UILabel *message*

establishes the properties and we should include it always.

Finally, the line

-(IBAction)showmessage;

specifies the action that will be done when the button is pressed; in this case a message will be written.

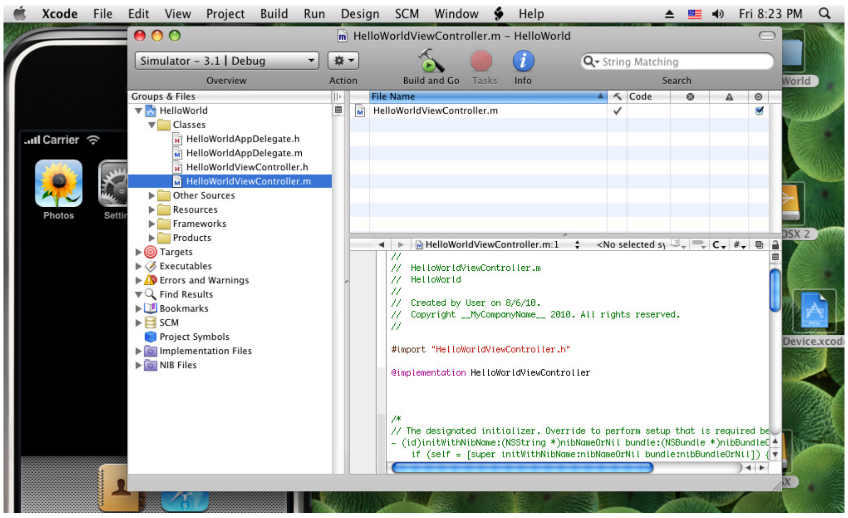


HelloWorldViewController.h:18

```
//
//  HelloWorldViewController.h
//  HelloWorld
//
//  Created by User on 8/6/10.
//  Copyright __MyCompanyName__ 2010. All rights reserved.
//

#import <UIKit/UIKit.h>

@interface HelloWorldViewController : UIViewController {

    IBOutlet UILabel *message;
}

@property (nonatomic,retain) IBOutlet UILabel *message;
- (IBAction)showmessage;
```

Figure 107. Saving the changes.

And the file will be saved, choosing *File-Save*.
Next we have to select the *HelloWorldViewController.m* file

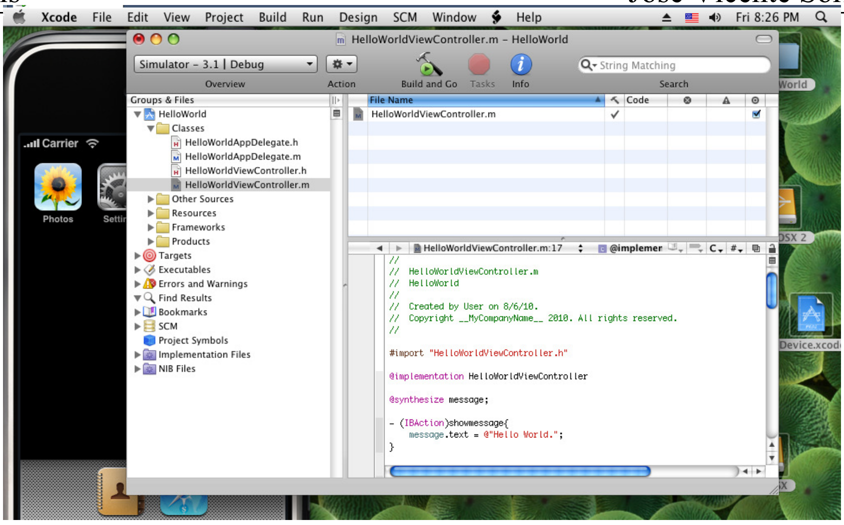


HelloWorldViewController.m:1

```
//
//  HelloWorldViewController.m
//  HelloWorld
//
//  Created by User on 8/6/10.
//  Copyright __MyCompanyName__ 2010. All rights reserved.
//

#import "HelloWorldViewController.h"

@implementation HelloWorldViewController



/*
// The designated initializer. Override to perform setup that is required be
- (id)initWithNibName:(NSString *)nibNameOrNil bundle:(NSBundle *)nibBundleO
    if (self = [super initWithNibName:nibNameOrNil bundle:nibBundleOrNil]) {
```

Figure 108. HelloWorldViewController.m.

And we add the following lines of code:



```
//
//  HelloWorldViewController.m
//  HelloWorld
//
//  Created by User on 8/6/10.
//  Copyright __MyCompanyName__ 2010. All rights reserved.
//

#import "HelloWorldViewController.h"

@implementation HelloWorldViewController

@synthesize message;

- (IBAction)showmessage{
    message.text = @"Hello World.";
}
```

Figure 109. Adding lines of code.

The line

@synthesize message

is also necessary for each variable used.

The *property* declaration provides a clear specification of how the accessory method will behave.

The word *synthesize* is used to tell the debugger that it should synthesize the set/get methods for the property.

Next, the following method has been introduced.

-(IBAction)showmessage{

message.text = @"Hello world.";

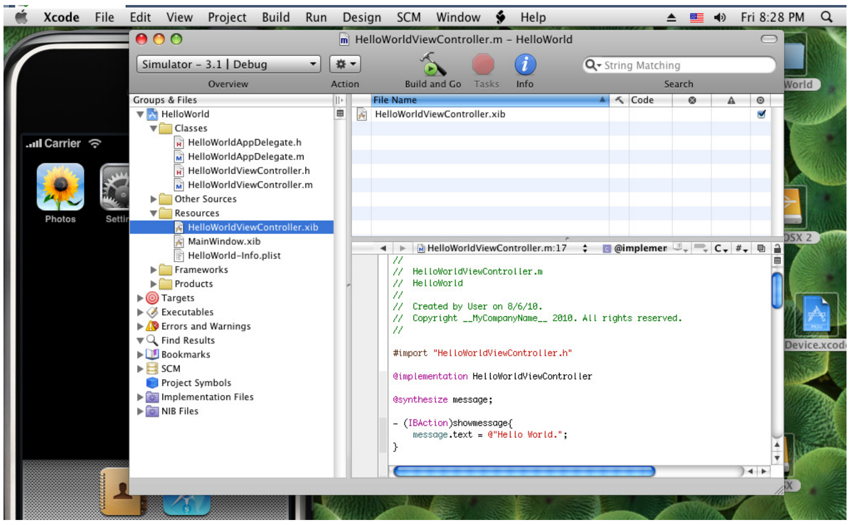
}

This function, indicates which action should be done when the user presses the button; in this case it should change the label text message showing a "Hello world" message.

@"Hello World" is written because that is the way a string literal is written.

We save the changes with *File-Save*.

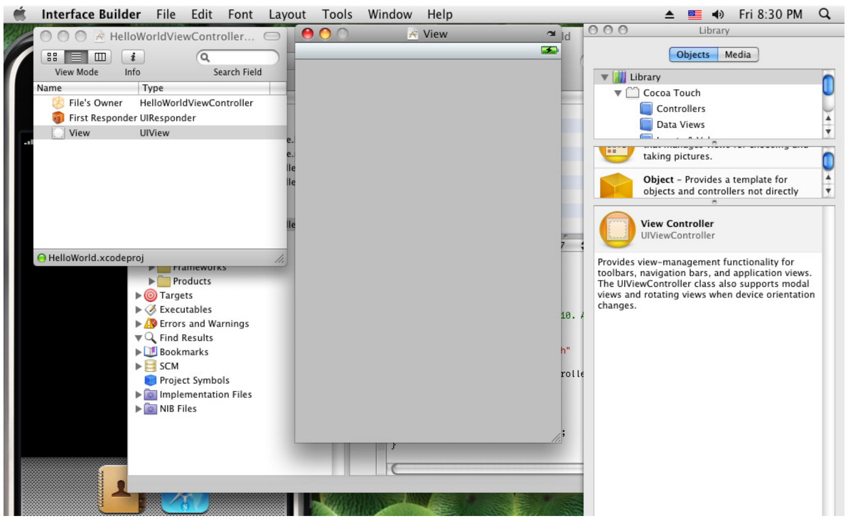
5. All the necessary code has been already written; now we start designing the interface.
   We open the *Resources* folder.



```
//
//  HelloWorldViewController.m
//  HelloWorld
//
//  Created by User on 8/6/10.
//  Copyright __MyCompanyName__ 2010. All rights reserved.
//

#import "HelloWorldViewController.h"

@implementation HelloWorldViewController

@synthesize message;

- (IBAction)showmessage{
    message.text = @"Hello World.";
}
```

Figure 110. Resources folder.

And we select the *HelloWorldViewController.xib* file double-clicing on it.

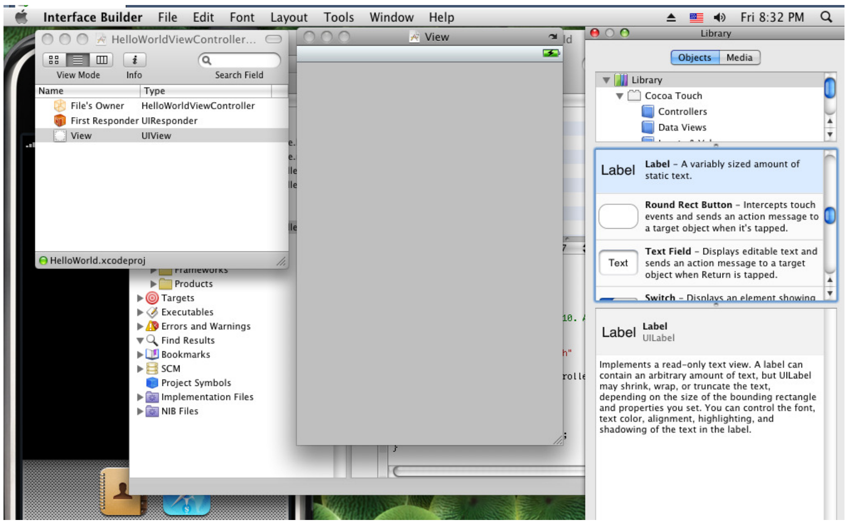

Interface Builder File Edit Font Layout Tools Window Help
HelloWorldViewController...
View Mode
Info
Search Field
Name
Type
File's Owner HelloWorldViewController
First Responder UIResponder
View UIView
HelloWorld.xcodeproj
View
Library
Objects
Media
Library
Cocoa Touch
Controllers
Data Views
Object – Provides a template for objects and controllers not directly
View Controller
UIViewController

Provides view-management functionality for toolbars, navigation bars, and application views. The UIViewController class also supports modal views and rotating views when device orientation changes.

Figure 111. Interface Builder.

Automatically, Interface Builder starts, showing several characteristic windows that will always appear.
At the left side we can see a window that we will use to link the Xcode code to the interface objects.
At the middle we have a *View* window, where we will design the interface.
At the right side the *Library* window can be seen, where we can choose the interface objects.

6. We insert a label object and a button (*Round Rect Button*).
It should simply be selected in the *Library* window and we drag and drop it to the *View* window.



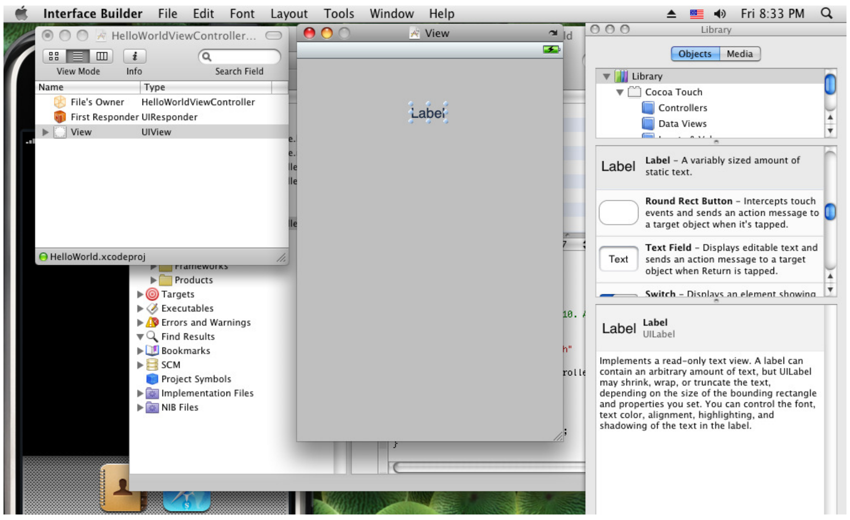
Label – A variably sized amount of static text.

Round Rect Button – Intercepts touch events and sends an action message to a target object when it's tapped.

Text Field – Displays editable text and sends an action message to a target object when Return is tapped.

Implements a read-only text view. A label can contain an arbitrary amount of text, but UILabel may shrink, wrap, or truncate the text, depending on the size of the bounding rectangle and properties you set. You can control the font, text color, alignment, highlighting, and shadowing of the text in the label.

Figure 112. Label object.



Label – A variably sized amount of static text.

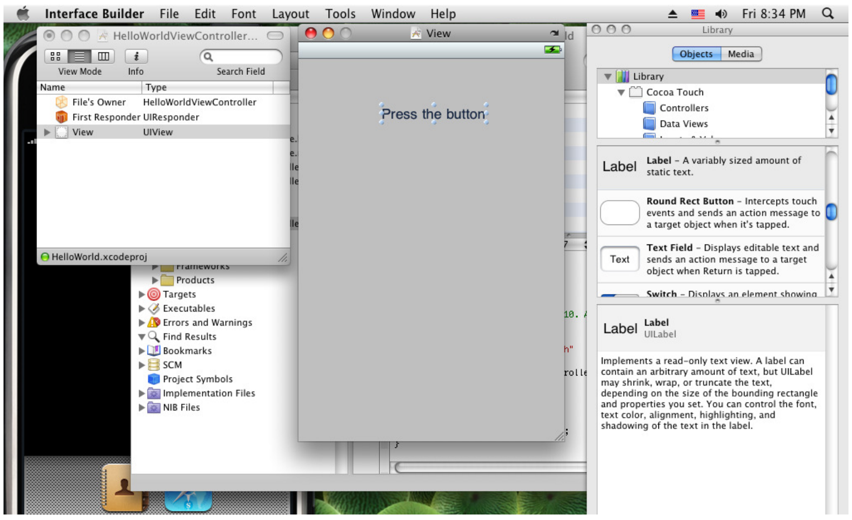
Round Rect Button – Intercepts touch events and sends an action message to a target object when it's tapped.

Text Field – Displays editable text and sends an action message to a target object when Return is tapped.

Implements a read-only text view. A label can contain an arbitrary amount of text, but UILabel may shrink, wrap, or truncate the text, depending on the size of the bounding rectangle and properties you set. You can control the font, text color, alignment, highlighting, and shadowing of the text in the label.

Figure 113. Drag and drop label object.

By double-clicking the label, we can change its content.
Let's write *Press the button.*



Name	Type
File's Owner	HelloWorldViewController
First Responder	UIResponder
View	UIView

Label – A variably sized amount of static text.

Round Rect Button – Intercepts touch events and sends an action message to a target object when it's tapped.

Text Field – Displays editable text and sends an action message to a target object when Return is tapped.

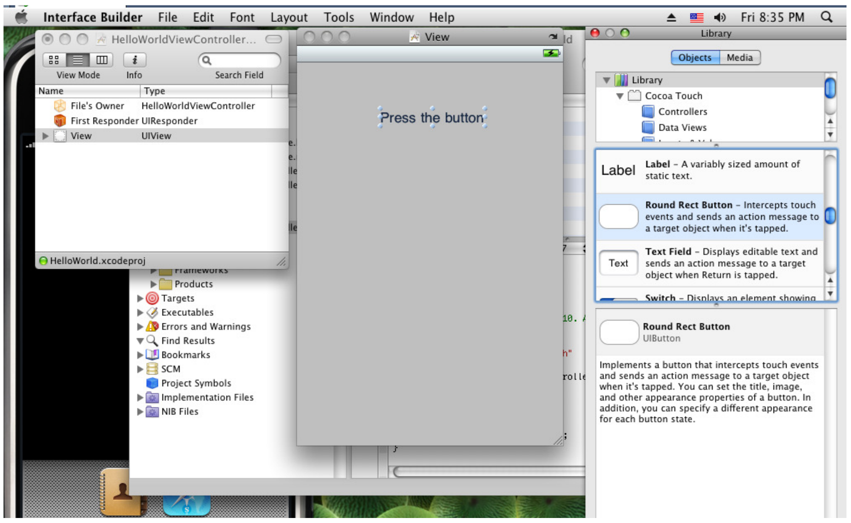
Label
UILabel

Implements a read-only text view. A label can contain an arbitrary amount of text, but UILabel may shrink, wrap, or truncate the text, depending on the size of the bounding rectangle and properties you set. You can control the font, text color, alignment, highlighting, and shadowing of the text in the label.

Figure 114. Content label changing.

We can change the location in the interface pressing the left button of the mouse on it and dragging and dropping it.
We can also change the size by pressing the left button of the mouse on the points that appear around it and dragging.

Then, we must select *Round Rect Button* in the *Library* window and drag and drop it to the *View* window.



Name	Type
File's Owner	HelloWorldViewController
First Responder	UIResponder
View	UIView

Label – A variably sized amount of static text.

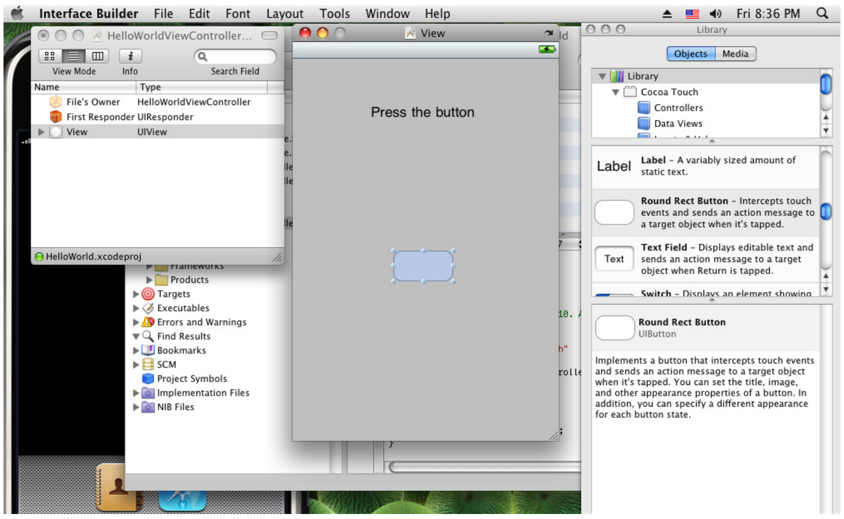
Round Rect Button – Intercepts touch events and sends an action message to a target object when it's tapped.

Text Field – Displays editable text and sends an action message to a target object when Return is tapped.

Round Rect Button
UIButton

Implements a button that intercepts touch events and sends an action message to a target object when it's tapped. You can set the title, image, and other appearance properties of a button. In addition, you can specify a different appearance for each button state.

Figure 115. Round rect button object.



Label – A variably sized amount of static text.

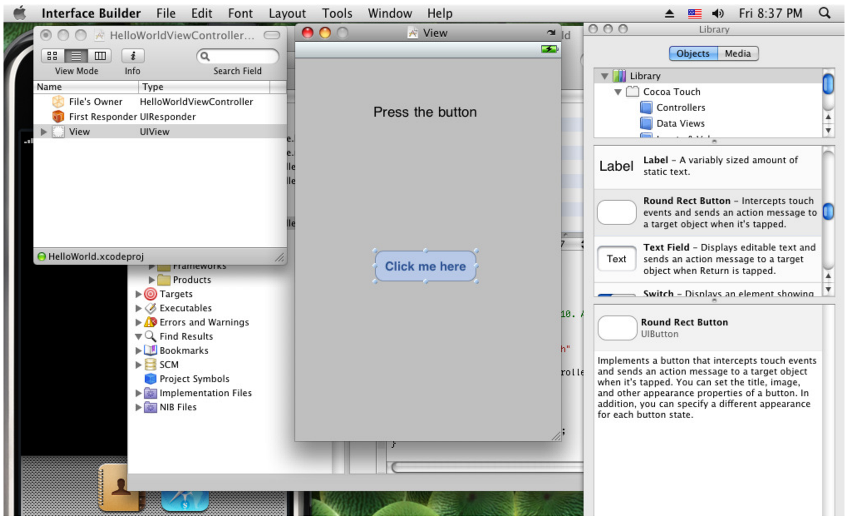
Round Rect Button – Intercepts touch events and sends an action message to a target object when it's tapped.

Text Field – Displays editable text and sends an action message to a target object when Return is tapped.

Implements a button that intercepts touch events and sends an action message to a target object when it's tapped. You can set the title, image, and other appearance properties of a button. In addition, you can specify a different appearance for each button state.

Figure 116. Round rect button in the view.

By double-clicking the button in the *View* window we can change the name and finally press *Enter*
Let's call it *Click me here*.



Label – A variably sized amount of static text.

Round Rect Button – Intercepts touch events and sends an action message to a target object when it's tapped.

Text Field – Displays editable text and sends an action message to a target object when Return is tapped.

Implements a button that intercepts touch events and sends an action message to a target object when it's tapped. You can set the title, image, and other appearance properties of a button. In addition, you can specify a different appearance for each button state.

Figure 117. Content button changing.

7. The interface is now completely designed.
   At this moment, we should tell Interface Builder what has to do the application when the button is pressed.

If we remember, in Xcode we have created a label variable called *message* and a function that changed the content of the label, whose name was *showmessage*.
The way to achieve this functionality is the following:
Select *File's owner* window at the left side, and press the right button of the mouse, and a new window will appear.

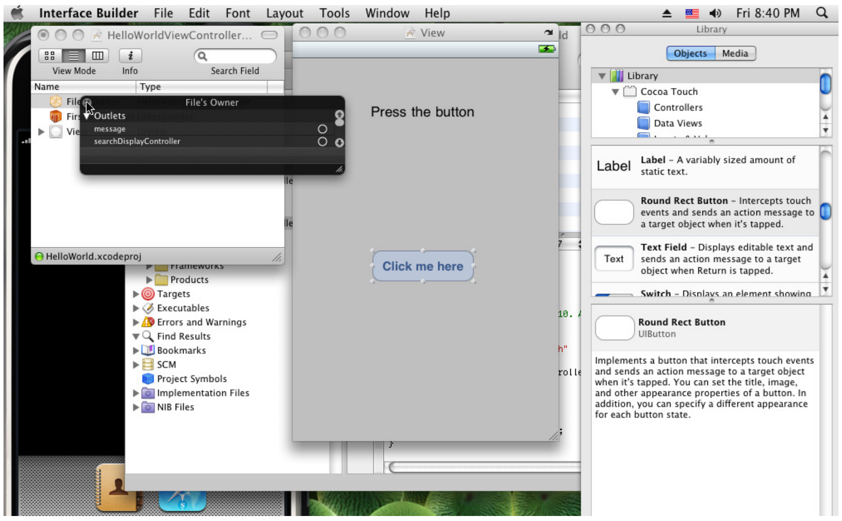
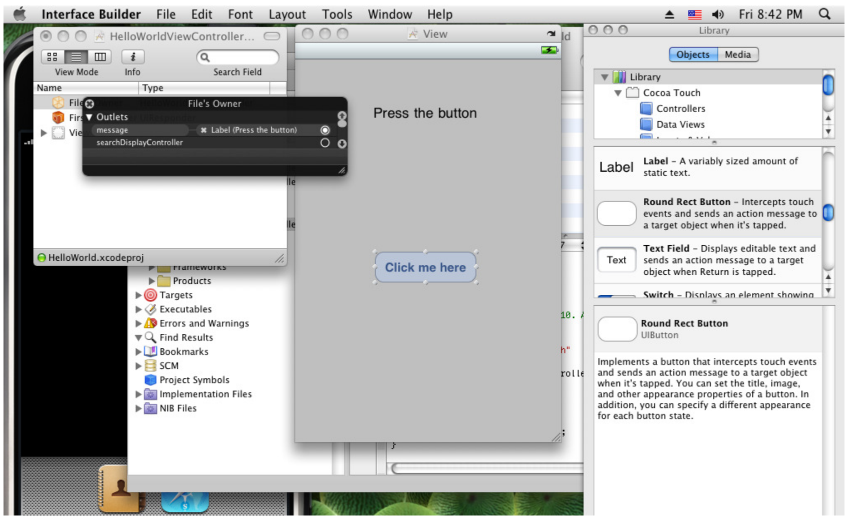


Figure 118. File's owner.

This window shows the elements that we declared in Xcode. In this case it shows *message* that is the variable declared in Xcode. At the right side of *message* we can see a circle.
Let's click on it with the left button of the mouse and drag to the interface label. When the label is blue-colored, we stop pressing the left button of the mouse.


Interface Builder File Edit Font Layout Tools Window Help
File's Owner
Outlets
message
Label (Press the button)
searchDisplayController
Press the button
Click me here
Library
Objects Media
Label – A variably sized amount of static text.
Round Rect Button – Intercepts touch events and sends an action message to a target object when it's tapped.
Text Field – Displays editable text and sends an action message to a target object when Return is tapped.
Round Rect Button
UIButton
Figure 119. Linking label object to variable.

In the previous window, at the right of *message* it appears *Label (Press the button)*. What we have just done is to link the label in Interface Builder to the variable *message* created in Xcode.
Then, we have to do the same for the button.
So we press the arrow at the right of the window, until it appears *showmessage*. After that we click on the circle at its right and drag to the button in the *View* window. When the button is blue-colored, we stop pressing the left button of the mouse. A contextual menu will appear and we must choose *Touch Up Inside*, that indicates Interface Builder that the action defined by *showmessage* in Xcode will be executed when the button is pressed.

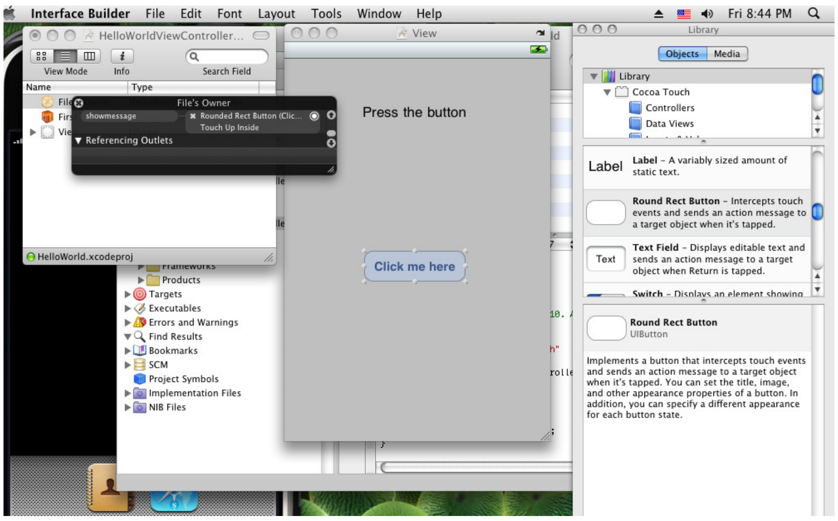
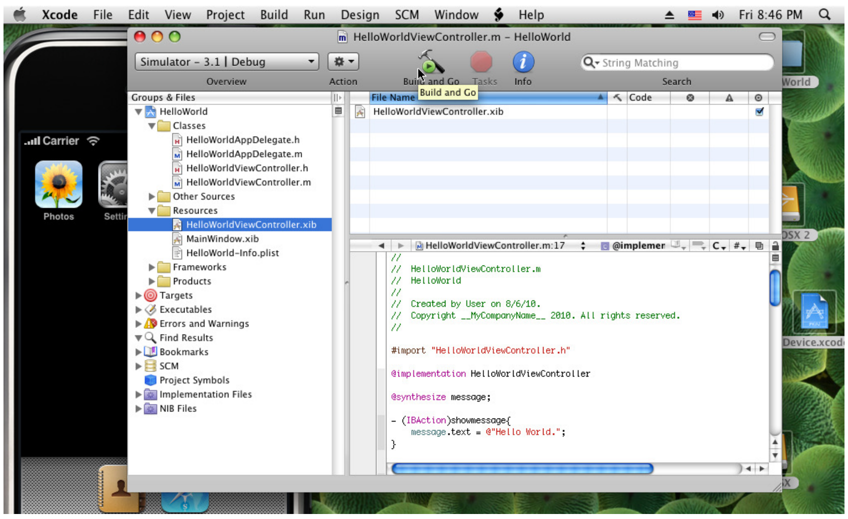


Figure 120. Linking button object to a variable.

A *Rounded Rect Button (Click me here) Touch Up Inside* will appear indicating that this action is linked to the action of pressing the button.
Finally let's close the window and save the changes with *File-Save*. Close Interface Builder with *File-Close*, and go back to Xcode.

8. We now have the application finished.
   At this moment, let's press *Build & Go*, in the upper part of the Xcode window, to build and run the application.
   The application will run in the iPhone Simulator.



```
//
//  HelloWorldViewController.m
//  HelloWorld
//
//  Created by User on 8/6/10.
//  Copyright __MyCompanyName__ 2010. All rights reserved.
//

#import "HelloWorldViewController.h"

@implementation HelloWorldViewController

@synthesize message;

- (IBAction)showmessage{
    message.text = @"Hello World.";
}
```

Figure 121. Build and go button.

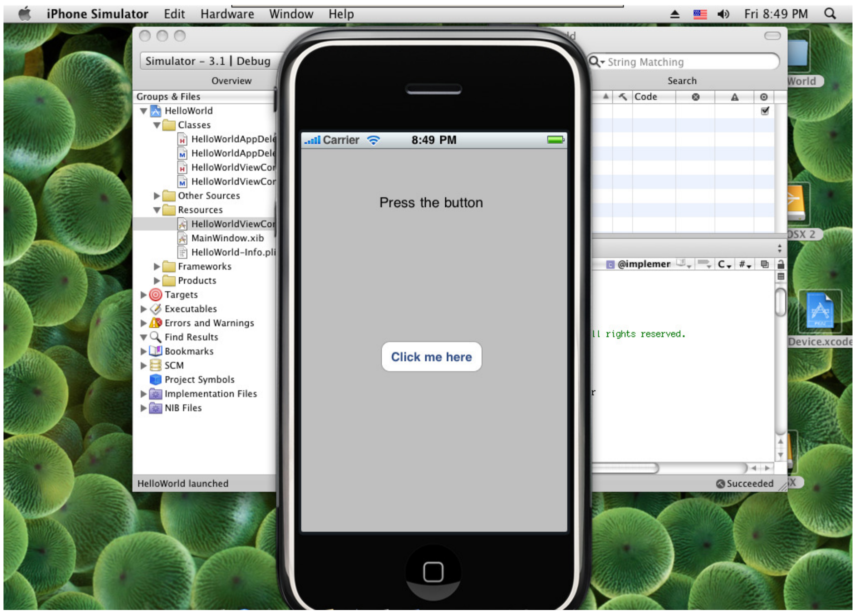


Figure 122. Application running in the simulator.

It will appear the initial message *Press the button* and the button with the *Click me here* message.

If we press the button, we'll notice that the text changes and "*Hello world*" will appear.

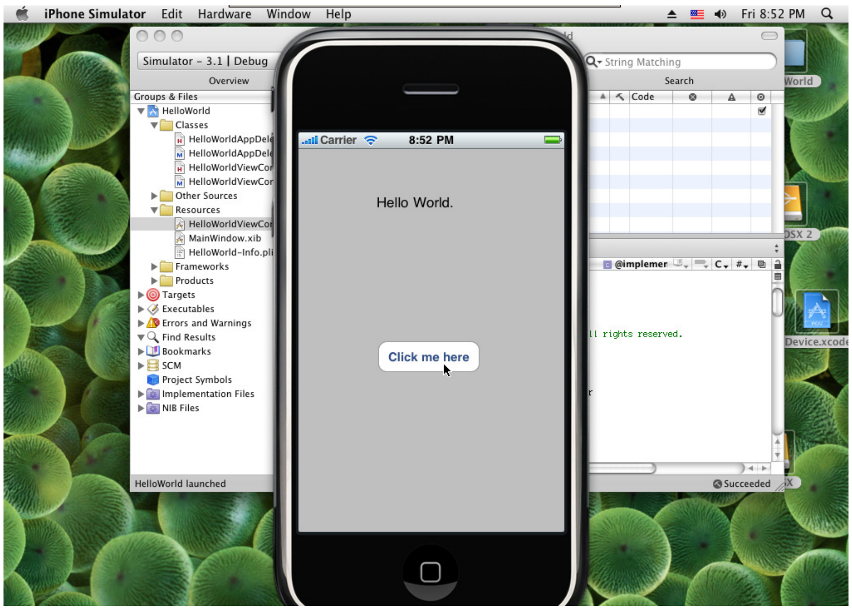
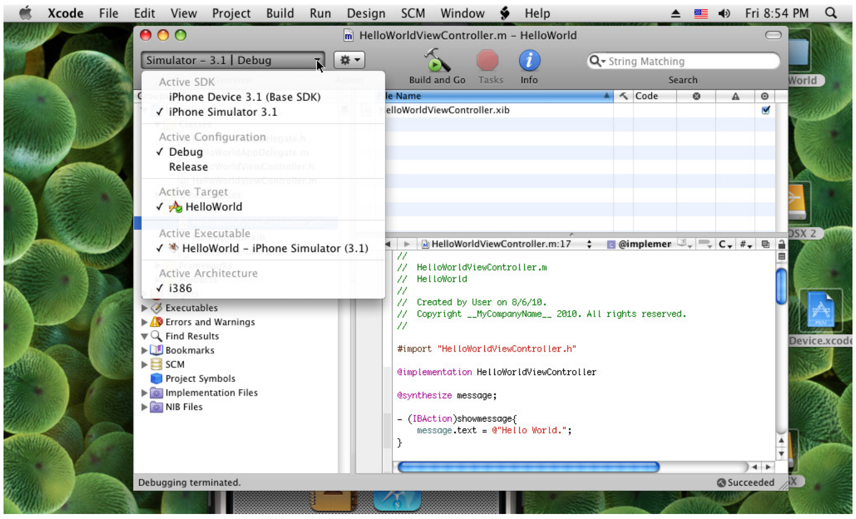


Figure 123. Hello world message after clicking button.

9. To finish the application, the button at the bottom of the simulator window must be pressed.

10. The reader may notice that the application is running, but this is not the last step, as we must load the application on the iPhone device. So we have to get a certificate from Apple, paying 99$ / year.
Once we have the certificate, and properly configured Xcode to use it, and before building and running the application we should choose several options in the *Simulator-3.1| Debug* menu.



Active SDK
iPhone Device 3.1 (Base SDK)
✓ iPhone Simulator 3.1

Active Configuration
✓ Debug
Release

Active Target
✓ HelloWorld

Active Executable
✓ HelloWorld – iPhone Simulator (3.1)

Active Architecture
✓ i386

```
//
//  HelloWorldViewController.m
//  HelloWorld
//
//  Created by User on 8/6/10.
//  Copyright __MyCompanyName__ 2010. All rights reserved.
//

#import "HelloWorldViewController.h"

@implementation HelloWorldViewController

@synthesize message;

- (IBAction)showmessage{
    message.text = @"Hello World.";
}
```

Figure 124. On-device loading options.

Let’s choose *iPhone Device 3.1 (Base SDK)* and *Release* options, connect the device to the computer via the USB port and press *Build & Go.* The application icon should appear on the iPhone device. Tap it and the application should start running.

Appendix 6

Basic steps to generate and distribute an iPhone app

First of all, we should download and install the iPhone SDK environment as it is shown in Annex 3.

Then we ought to develop and build the application as in Annex 4. The application is now running on iPhone Simulator, but this is just a simulator. Next we ought to load our application into iPhone device to test it. This is the main goal of this Annex, which is structured in two basic steps.

1. Testing the app on the device:

Once the app is completely finished, we may want to test it on the device. This can be easily done.
The process can be seen in the following figure.

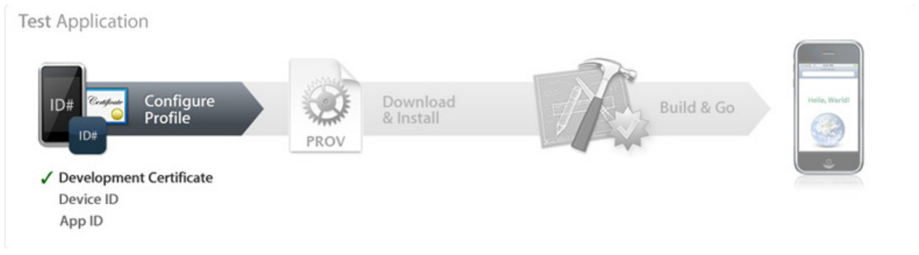


Figure 125. Process to test an application on the device.

1.1 About the process:

The first step consists of registering in the apple's web site.

First of all, you ought to **obtain your iPhone Development Certificate**:

In the 'Certificates' section of the iPhone Provisioning Portal, you can request individual iPhone Development Certificates. All iPhone applications must be signed by a valid certificate before they can be run on an Apple device. In order to sign applications for testing purposes, Team Members need an iPhone Development Certificate.

A digital identity is an electronic means of identification consisting of a secret "private key" and a shared "public key". This private key allows Xcode to sign your iOS application binary.
The digital certificates you request and download are electronic documents that associate your digital identity with other information, including your name, email address, or business. An iPhone Development Certificate is restricted to application development only and is valid for a limited amount of time. The Apple Certification Authority can also invalidate ("revoke") a certificate before it expires.
The following things can be done:

a) Generate a Certificate Signing Request
b) Submit a Certificate Signing Request for Approval
c) Certificate Signing Requests should be approved.
d) Download and Install a Development Certificate.
e) Save your Private Key and Transfer to other Systems.

Second, you should assign an **Apple device** to your team.
The Devices section of the iPhone Provisioning Portal allows you to enter the Apple devices that you will be using for your iOS development. In order to debug your iOS application on an Apple device, a Team Agent or Team Admin must first enter the Unique Device Identifier (UDID) for each iPhone and iPod touch into the Provisioning Portal. The UDID is a 40 character string that is tied to a single device, similar to a serial number. These UDIDs are included in the provisioning profiles created later. You can input up to 200 devices for your development team.
The following things can be done:
a) Locate a unique device ID
b) Add individual devices
c) Bulk upload of devices
d) Remove devices from your development team.
e) Edit devices on your development team
f) Install iOS.

Then, you should create an **app ID**
An APP ID is a unique identifier that iOS uses to allow your application to connect to the Apple Push Notification service, share keychain data between applications, and communicate with external hardware accessories you wish to pair your iOS application with. In order to install your application on an iOS based device, you will need to create an App ID.
Each app ID consists of a universally unique 10 character “Bundle Seed ID” prefix generated by Apple and a “Bundle Identifier” suffix that is enter by a Team Admin in the Provisioning Portal. The recommended practice is to use a reverse-domain name style string for the “Bundle Identifier” portion of the App ID. An example App ID would be: 8E549T7128.com.apple.AddressBook.

If you are creating a suite of applications that will share the same Keychain access (e.g. sharing passwords between applications) or have no Keychain Access requirements, you can create a single App ID for your entire application suite utilizing a trailing asterisk as a wild-card character. The wild-card character must be the last character in the App ID string. Example App IDs for this situation could be: R2T24EVAEE.com.domainname.* or R2T24EVAEE.*
In these wild-card situations, you will simply replace the asterisk with whatever string you wish in the CF Bundle Identifier field in Xcode.
The following things can be done.

g) Generate an App ID.
h) Register an App ID for Apple Push Notification Service.
i) Register an App ID for in App Purchases and Game Center.

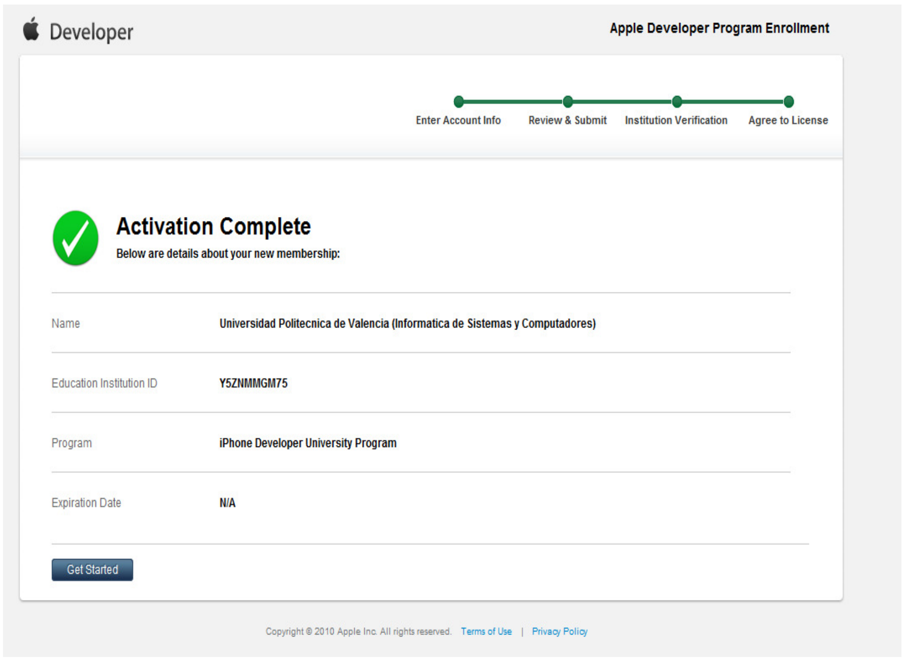
Finally, you should create and download a **Development Provisioning Profile**
A Provisioning Profile is a collection of digital entities that uniquely ties developers and devices to an authorized iPhone Development Team and enables a device to be used for testing. A Development Provisioning Profile must be installed on each device on which you wish to run your application code. Each Development Provisioning Profile will contain a set of iPhone Development Certificates, Unique Device Identifiers and an App ID.
Devices specified within the provisioning profile can be used for testing only by those individuals whose iPhone Development Certificates are included in the profile. A single device can contain multiple provisioning profiles.
You can do the following:

j) Create a Development Provisioning Profile
k) Install a Development Provisioning Profile
l) Build and install a Development Application.

A few things have to be done **in Xcode** before loading the application into the device. These will be explained later.

1.2 Steps to load the application into a device

1 In Apple's iPhone developer web site, we must fill in a form applying Apple for a license. In this case, free licensing for a University Program has been applied for.
2. In that form, enter account information on the University Institution or yours, review the information entered and submit it.
3. An Institution verification of a few weeks will be started.
4. Finally, after carefully verification, Apple agrees to license, and an Activation Complete message is received.


Developer

Apple Developer Program Enrollment

Enter Account Info · Review & Submit · Institution Verification · Agree to License

Activation Complete

Below are details about your new membership:

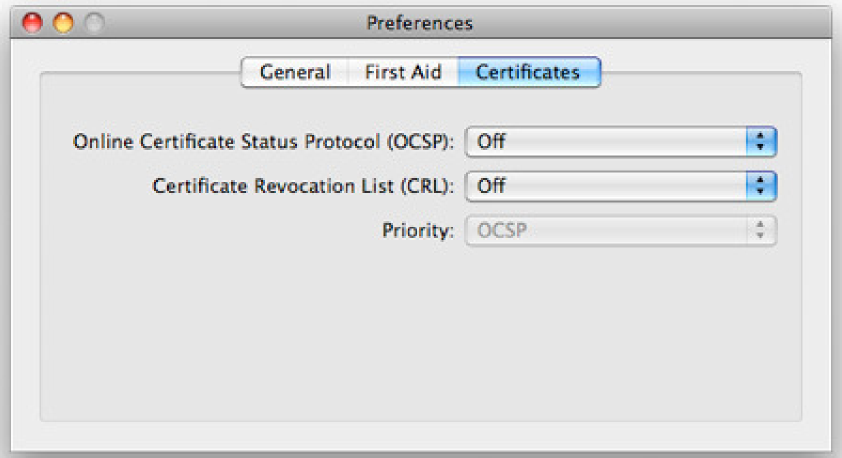
Name	Universidad Politecnica de Valencia (Informatica de Sistemas y Computadores)
Education Institution ID	Y5ZNMMGM75
Program	iPhone Developer University Program
Expiration Date	N/A

Get Started

Copyright © 2010 Apple Inc. All rights reserved. Terms of Use | Privacy Policy

Figure 126. Activation message.

5. Once Apple has sent the license agreement, the next step is to generate a certificate signing request.
6. In the Applications folder of your Mac, open the Utilities folder and launch Keychain Access.
7. In the Preferences menu, set Online Certificate Status Protocol (OSCP) and Certificate Revocation List (CRL) to “off”.


Preferences

General · First Aid · Certificates

Online Certificate Status Protocol (OCSP): Off

Certificate Revocation List (CRL): Off

Priority: OCSP

Figure 127. Keychain Access Preferences menu.

8. Choose Keychain Access - Certificate Assistant – Request a Certificate from a Certificate Authority

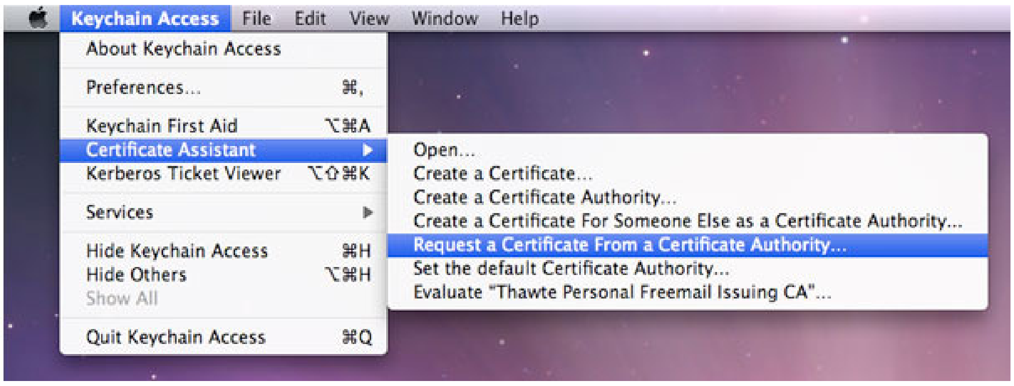
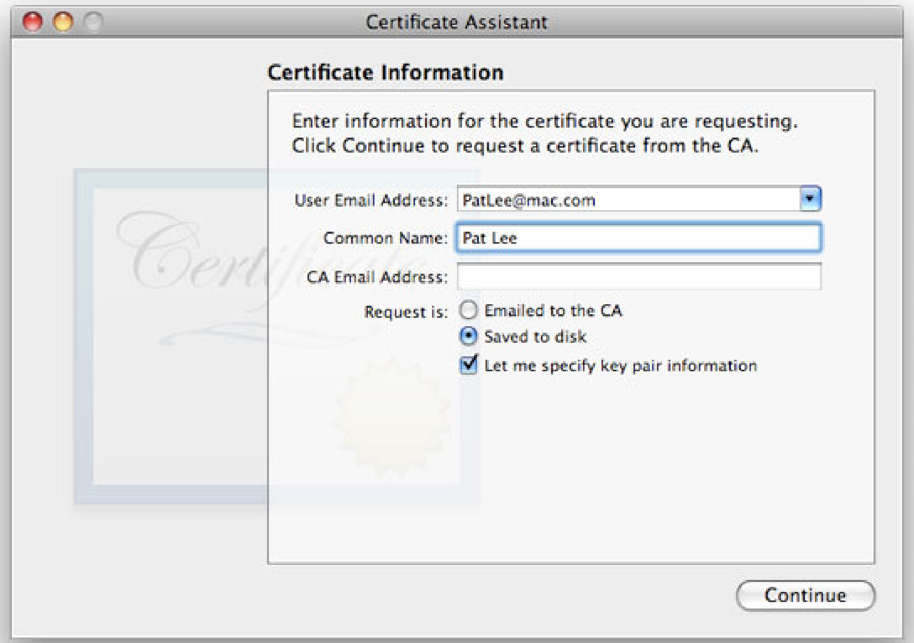

Keychain Access
File
Edit
View
Window
Help
About Keychain Access
Preferences... ⌘,
Keychain First Aid ⌥⌘A
Certificate Assistant
Kerberos Ticket Viewer ⌥⇧⌘K
Services
Hide Keychain Access ⌘H
Hide Others ⌥⌘H
Show All
Quit Keychain Access ⌘Q
Open...
Create a Certificate...
Create a Certificate Authority...
Create a Certificate For Someone Else as a Certificate Authority...
Request a Certificate From a Certificate Authority...
Set the default Certificate Authority...
Evaluate "Thawte Personal Freemail Issuing CA"...

Figure 128. Certificate Assistant.

9. In the User Email Address field, enter your email address. It should match to the information submitted when you registered as an iPhone Developer.
10. In the Common Name field enter your name. The name should match to the information submitted when you registered.
11. No CA (Certificate Authority) Email Address is required.
12. Select the "Saved to Disk" and select "Let me specify key pair information" and click "Continue"


Certificate Assistant

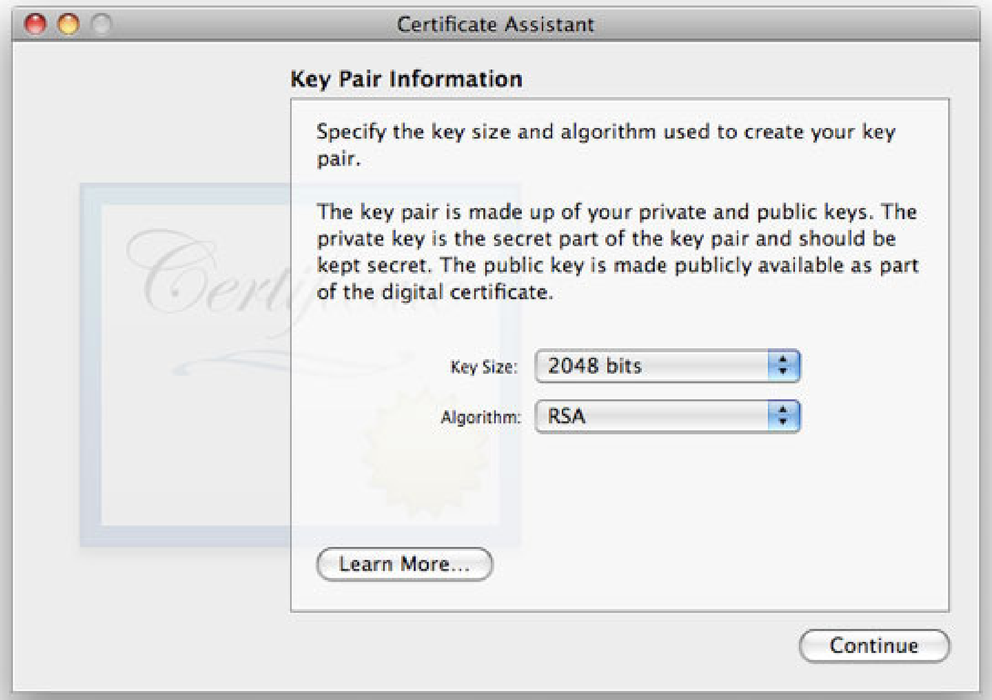
Certificate Information

Enter information for the certificate you are requesting.
Click Continue to request a certificate from the CA.

User Email Address: PatLee@mac.com
Common Name: Pat Lee
CA Email Address:
Request is: ○ Emailed to the CA [✓ Saved to disk]
[✓ Let me specify key pair information]

Figure 129. Certificate Assistant-Certificate information.

13. If "Let me specify key pair" was selected, specify a file name and click "Save". In the following screen select "2048 bits" for the Key Size and "RSA" for the Algorithm. Click "Continue".



Key Pair Information

Specify the key size and algorithm used to create your key pair.

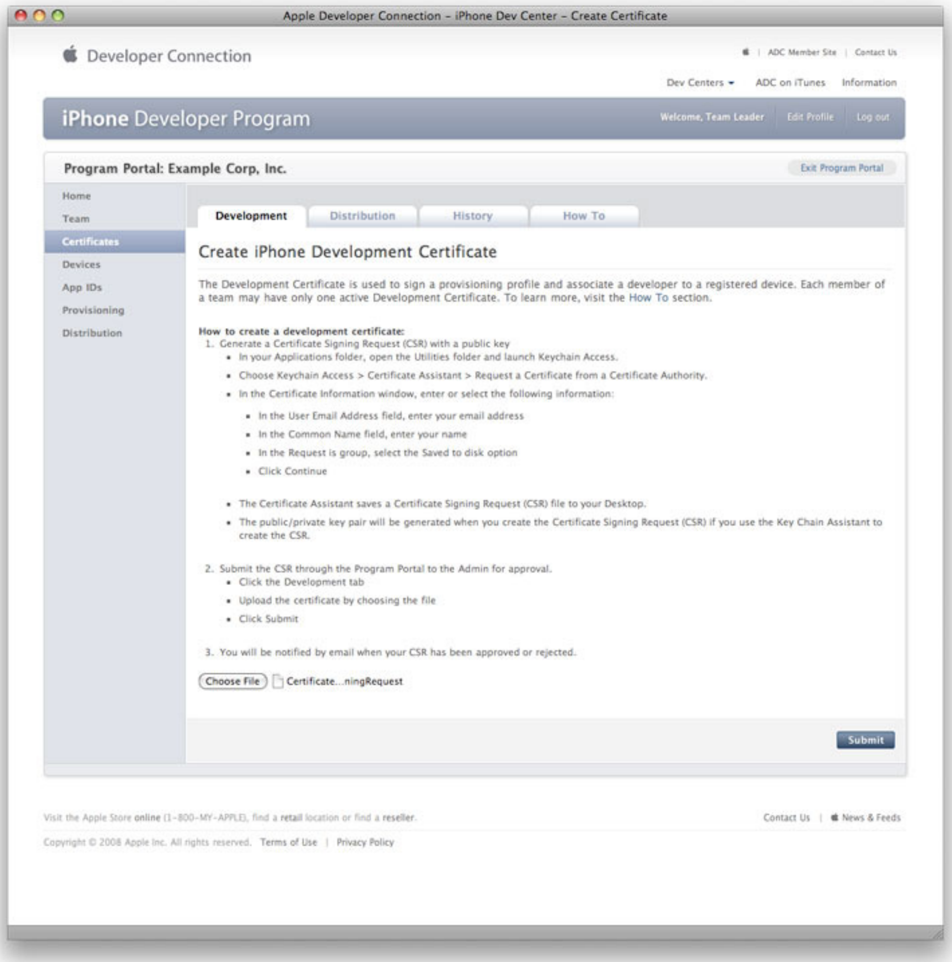
The key pair is made up of your private and public keys. The private key is the secret part of the key pair and should be kept secret. The public key is made publicly available as part of the digital certificate.

Key Size: 2048 bits

Algorithm: RSA

Figure 130. Certificate Assistant – Key Pair information.

14. The Certificate Assistant will create a CSR file on your desktop.
15. Then you should submit a Certificate Signing Request for Approval
16. After creating a CSR, log in to the iPhone Provisioning Portal and navigate to “Certificates” – “Development” and click “Add Certificate”.
17. Click the “Choose file” button, select your CSR and click ”Submit”. If the key size was not set to 2048 bits during the CSR creation process, the Portal will reject the CSR.
18. Upon submission, Team Admins will be notified via email of the certificate request.
19. Once your CSR is approved or rejected by a Team Admin, you will be notified via email of the change in your certificate status.



Program Portal: Example Corp, Inc.

Create iPhone Development Certificate

The Development Certificate is used to sign a provisioning profile and associate a developer to a registered device. Each member of a team may have only one active Development Certificate. To learn more, visit the How To section.

How to create a development certificate:

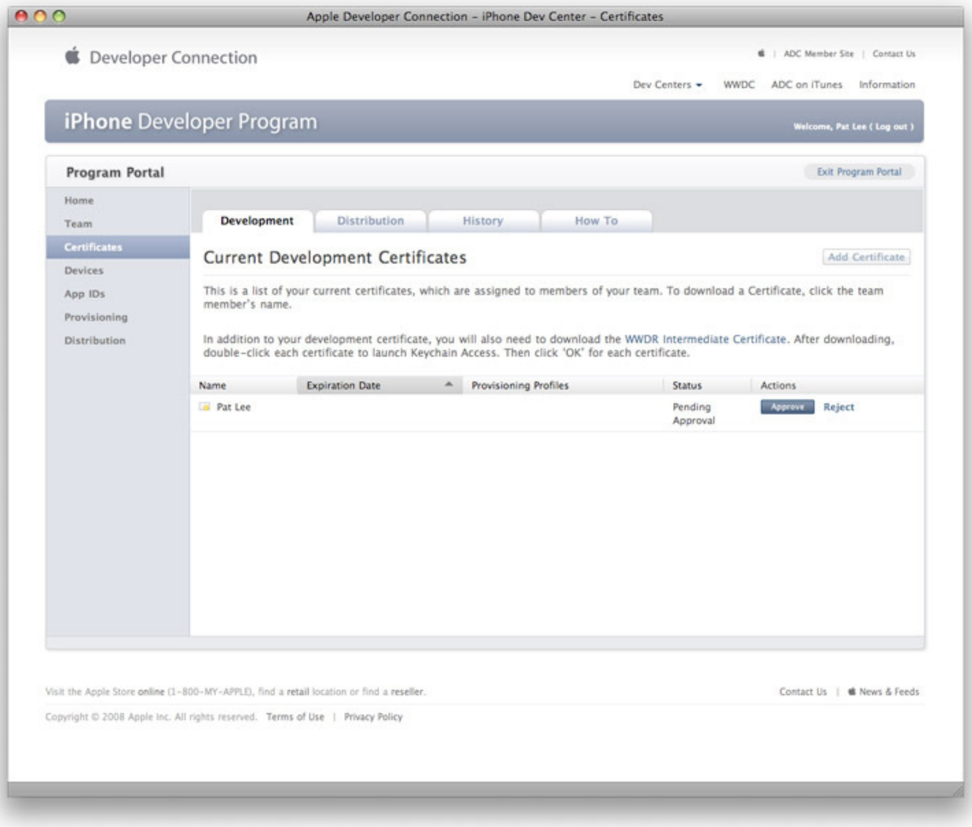
1. Generate a Certificate Signing Request (CSR) with a public key
   - In your Applications folder, open the Utilities folder and launch Keychain Access.
   - Choose Keychain Access > Certificate Assistant > Request a Certificate from a Certificate Authority.
   - In the Certificate Information window, enter or select the following information:
     - In the User Email Address field, enter your email address
     - In the Common Name field, enter your name
     - In the Request is group, select the Saved to disk option
     - Click Continue
   - The Certificate Assistant saves a Certificate Signing Request (CSR) file to your Desktop.
   - The public/private key pair will be generated when you create the Certificate Signing Request (CSR) if you use the Key Chain Assistant to create the CSR.
2. Submit the CSR through the Program Portal to the Admin for approval.
   - Click the Development tab
   - Upload the certificate by choosing the file
   - Click Submit
3. You will be notified by email when your CSR has been approved or rejected.

Visit the Apple Store online (1-800-MY-APPLE), find a retail location or find a reseller.

Copyright © 2008 Apple Inc. All rights reserved. Terms of Use | Privacy Policy

Figure 131. Creating iPhone development certificate.

20. Team Agents and Team Admins have the authority and responsibility to approve or reject all iPhone Development Certificate requests. In order to approve/reject Team Members' requests, all Team Admins should first submit their own CSR for approval.
21. After submitting a CSR for approval, Team Admins will be directed to the "Development" tab of the "Certificates" section. Here, CSRs can be approved or rejected by clicking the corresponding action next to each request.
22. Once a CSR is approved or rejected, the requesting Team Member is notified via email of the change in their certificate status. Each iPhone Development Certificate is available to both the Team Member who submitted the CSR for approval and to the Team Admins


Apple Developer Connection – iPhone Dev Center – Certificates

Developer Connection

ADC Member Site | Contact Us

Dev Centers WWDC ADC on iTunes Information

iPhone Developer Program

Welcome, Pat Lee (Log out)

Program Portal

Exit Program Portal

Home
Team
Certificates
Devices
App IDs
Provisioning
Distribution

Development Distribution History How To

Current Development Certificates

Add Certificate

This is a list of your current certificates, which are assigned to members of your team. To download a Certificate, click the team member's name.

In addition to your development certificate, you will also need to download the WWDR Intermediate Certificate. After downloading, double-click each certificate to launch Keychain Access. Then click 'OK' for each certificate.

Name	Expiration Date	Provisioning Profiles	Status	Actions
Pat Lee			Pending Approval	Approve Reject

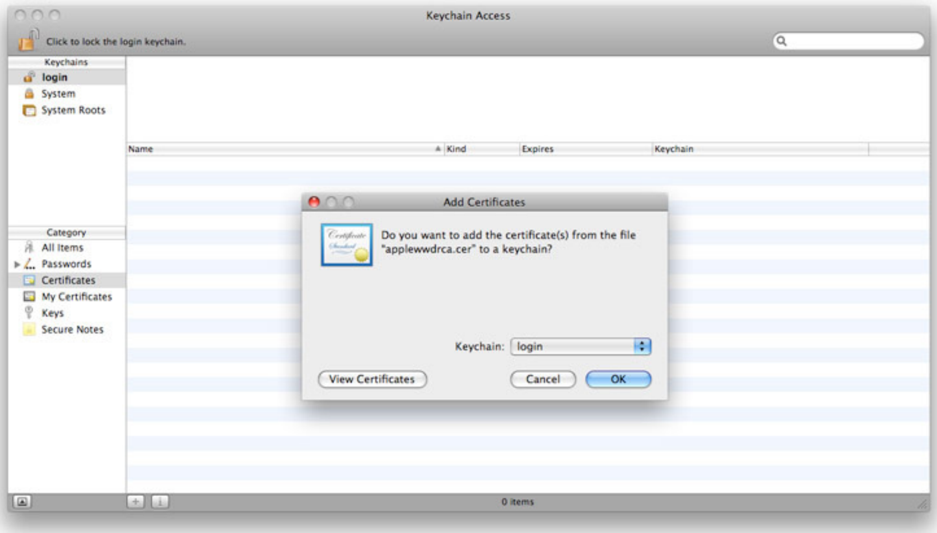
Visit the Apple Store online (1-800-MY-APPLE), find a retail location or find a reseller.

Contact Us | News & Feeds

Copyright © 2008 Apple Inc. All rights reserved. Terms of Use | Privacy Policy

Figure 132. Current development certificates.

23. Next we should download and install development certificates, doing the following.
24. In the “Certificates” – “Distribution” section of the Portal, control-click the WWDR Intermediate Certificate link and select “Saved Linked File to Downloads” to initiate download of the certificate.
25. On your local machine, double-click the WWDR Intermediate certificate to launch Keychain Access and install.
26. Upon CSR approval, Team Members and Team Admins can download their certificates via the “Certificates” section of the Provisioning Portal. Click “Download” next to the certificate name to download your iPhone Development Certificate to your local machine.
27. On your local machine, double-click the downloaded .cer file to launch Keychain Access and install your certificate.
28. Team Members can only download their own iPhone Development Certificates. Team Admins have the authority to download the public certificates of all of their Team Members. Apple never receives the private key for a CSR. The private keys are not available to anyone except the original key pair creator and are stored in the system keychain of that Team Member.



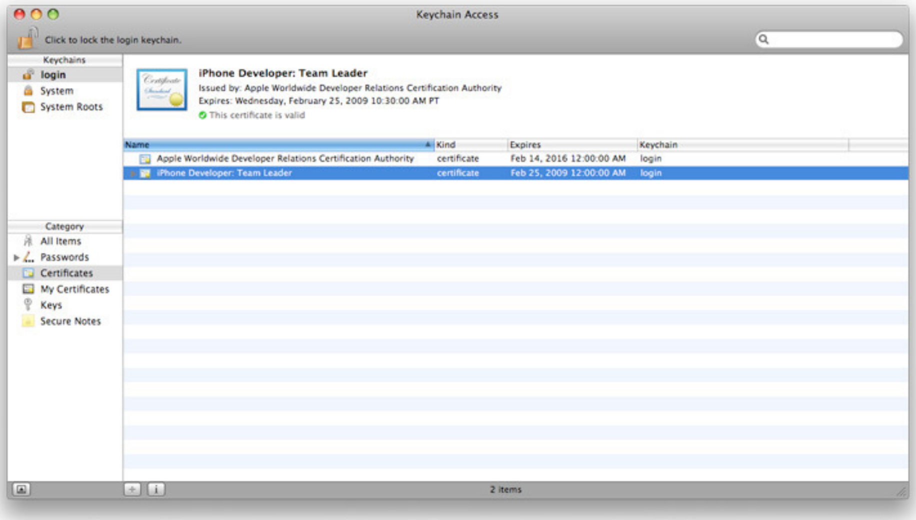
Do you want to add the certificate(s) from the file "applewwdrca.cer" to a keychain?

Keychain: login

Name	Kind	Expires	Keychain

0 items

Figure 133. Adding certificates.



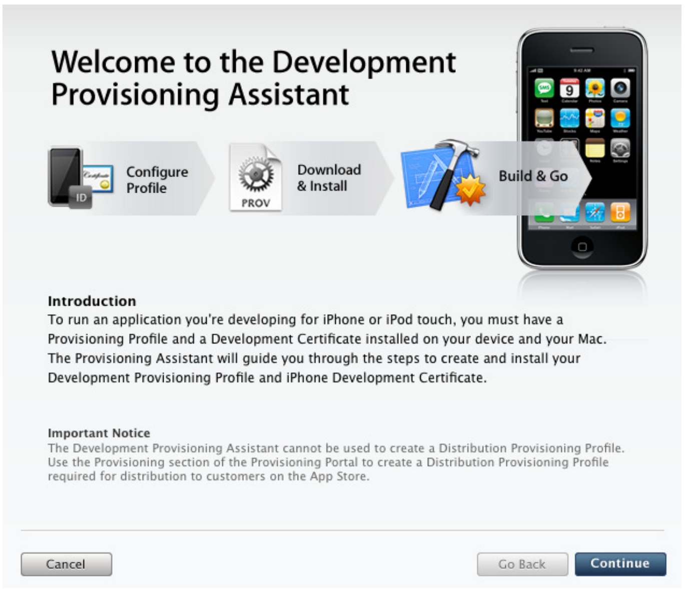
iPhone Developer: Team Leader
Issued by: Apple Worldwide Developer Relations Certification Authority
Expires: Wednesday, February 25, 2009 10:30:00 AM PT
This certificate is valid

Name	Kind	Expires	Keychain
Apple Worldwide Developer Relations Certification Authority	certificate	Feb 14, 2016 12:00:00 AM	login
iPhone Developer: Team Leader	certificate	Feb 25, 2009 12:00:00 AM	login

2 items

Figure 134. Available certificates.

29. In the following steps, an App ID, a Device ID and a Provisioning Profile must be set.
30. Click the "Launch Assistant" button from "iPhone Provisioning Portal" section. This wizard will guide you in the process to create and install a Provisioning Profile and iPhone Development Certificate, needed to build and install applications you are developing for iPhone.
31. A welcome window will appear. Click "Continue"



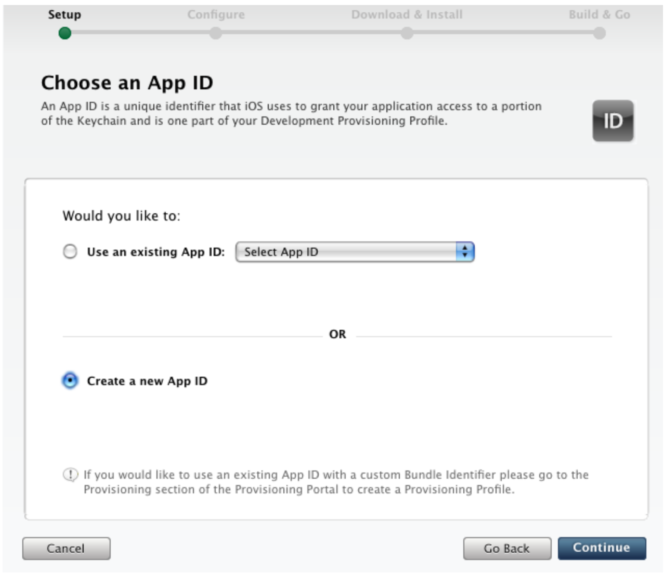
Welcome to the Development Provisioning Assistant

Introduction
To run an application you're developing for iPhone or iPod touch, you must have a Provisioning Profile and a Development Certificate installed on your device and your Mac. The Provisioning Assistant will guide you through the steps to create and install your Development Provisioning Profile and iPhone Development Certificate.

Important Notice
The Development Provisioning Assistant cannot be used to create a Distribution Provisioning Profile. Use the Provisioning section of the Provisioning Portal to create a Distribution Provisioning Profile required for distribution to customers on the App Store.

Figure 135. Launch assistant welcome.

32. Next the wizard will ask us to choose an existing App ID or create a new one. In the first two figures we create a new App ID, whereas in the third one we finally use an existing one.



Choose an App ID
An App ID is a unique identifier that iOS uses to grant your application access to a portion of the Keychain and is one part of your Development Provisioning Profile.

Would you like to:

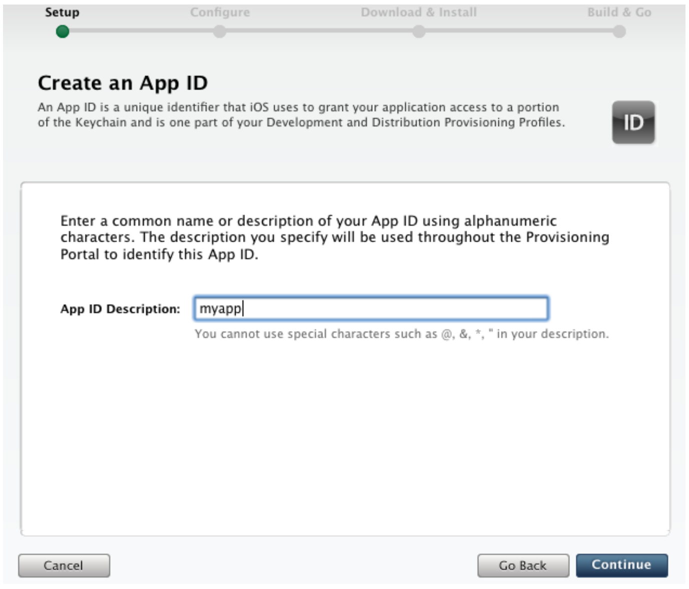
Use an existing App ID: Select App ID

OR

[✓ Create a new App ID]

If you would like to use an existing App ID with a custom Bundle Identifier please go to the Provisioning section of the Provisioning Portal to create a Provisioning Profile.

Figure 136. Create a new App ID.



Create an App ID

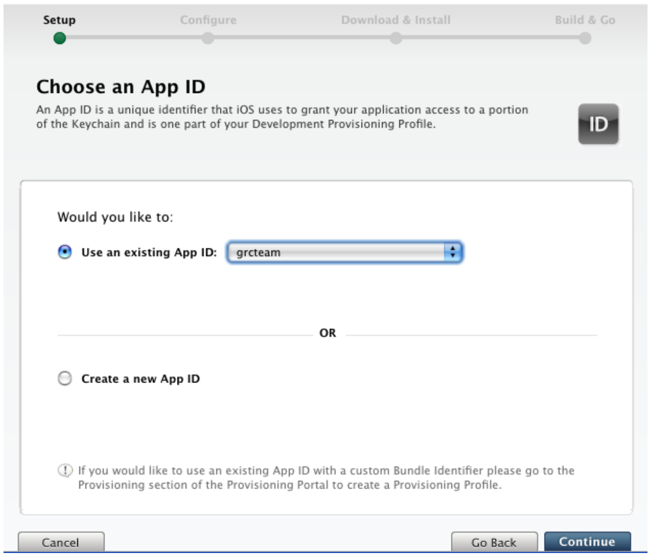
An App ID is a unique identifier that iOS uses to grant your application access to a portion of the Keychain and is one part of your Development and Distribution Provisioning Profiles.

Enter a common name or description of your App ID using alphanumeric characters. The description you specify will be used throughout the Provisioning Portal to identify this App ID.

App ID Description: myapp

You cannot use special characters such as @, &, *, " in your description.

Figure 137. Creating a new App ID.



Choose an App ID

An App ID is a unique identifier that iOS uses to grant your application access to a portion of the Keychain and is one part of your Development Provisioning Profile.

Would you like to:

[✓ Use an existing App ID:] grcteam

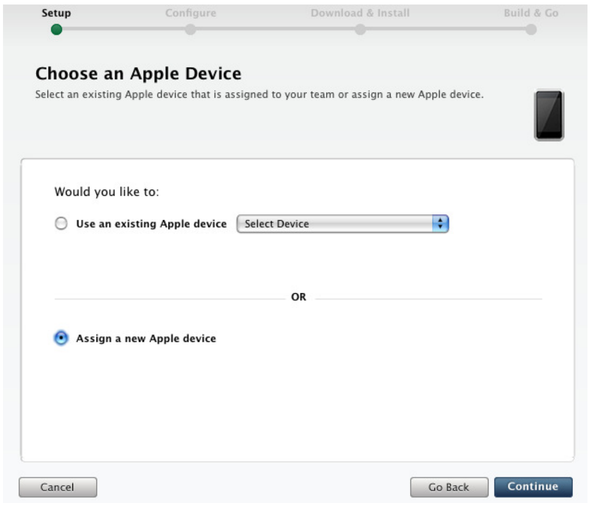
OR

Create a new App ID

If you would like to use an existing App ID with a custom Bundle Identifier please go to the Provisioning section of the Provisioning Portal to create a Provisioning Profile.

Figure 138. Use an existing App ID.

33. Next we ought to choose an App device or assign a new one. The first two figures show the way to assign a new App device, while the third figure shows how to choose an existing one.



Choose an Apple Device

Select an existing Apple device that is assigned to your team or assign a new Apple device.

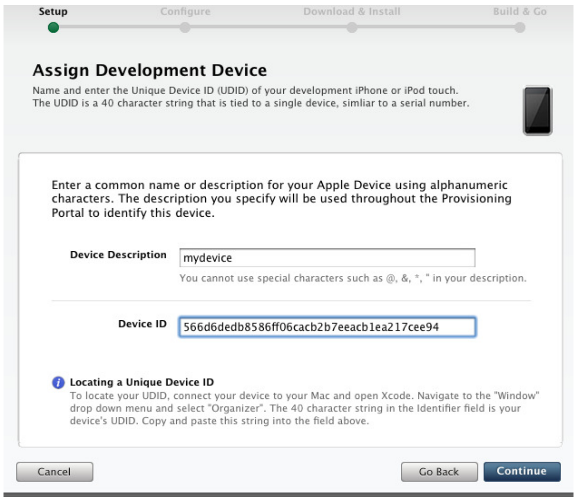
Would you like to:

Use an existing Apple device — Select Device

OR

[✓ Assign a new Apple device]

Figure 139. Assign a new Apple device.



Assign Development Device

Name and enter the Unique Device ID (UDID) of your development iPhone or iPod touch.
The UDID is a 40 character string that is tied to a single device, simliar to a serial number.

Enter a common name or description for your Apple Device using alphanumeric characters. The description you specify will be used throughout the Provisioning Portal to identify this device.

Device Description: mydevice

You cannot use special characters such as @, &, *, " in your description.

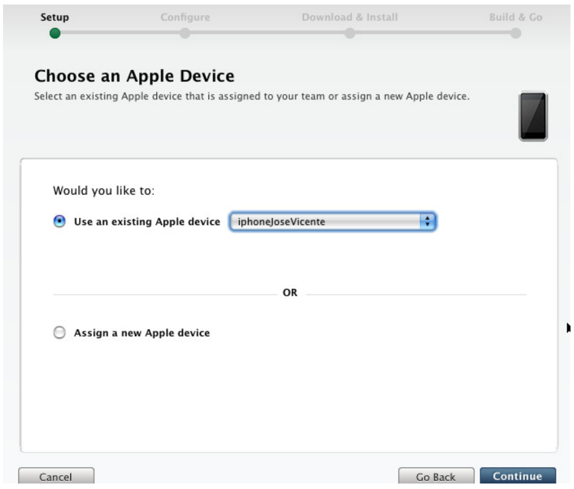
Device ID: 566d6dedb8586ff06cacb2b7eeacb1ea217cee94

Locating a Unique Device ID

To locate your UDID, connect your device to your Mac and open Xcode. Navigate to the "Window" drop down menu and select "Organizer". The 40 character string in the Identifier field is your device's UDID. Copy and paste this string into the field above.

Figure 140. Assigning a new Apple device.

The Device ID is your iPhone 40 character string serial number. The window shows information on how to obtain it.



Choose an Apple Device

Select an existing Apple device that is assigned to your team or assign a new Apple device.

Would you like to:

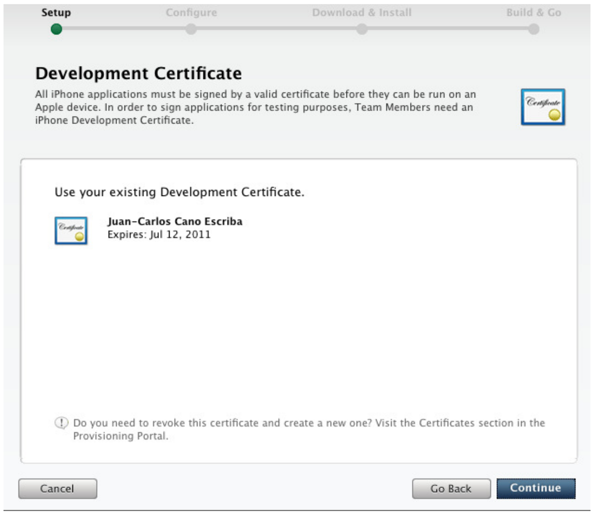
[✓ Use an existing Apple device] — iphoneJoseVicente

OR

Assign a new Apple device

Figure 141. Use an existing Apple device.

34. Then a window tells you to confirm the development certificate we created before.



Development Certificate

All iPhone applications must be signed by a valid certificate before they can be run on an Apple device. In order to sign applications for testing purposes, Team Members need an iPhone Development Certificate.

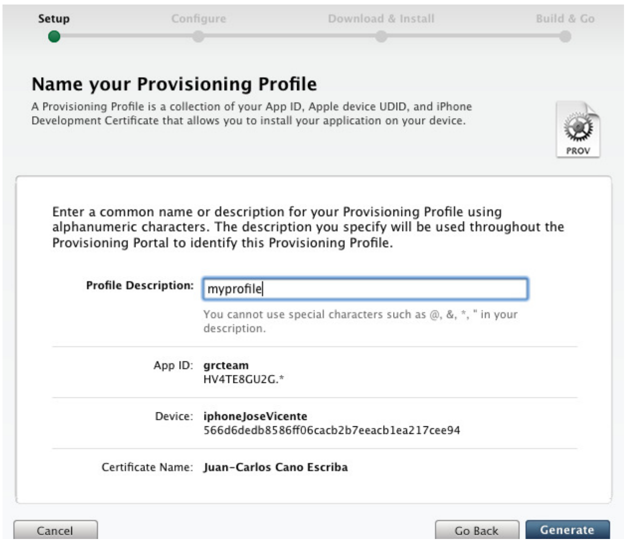
Use your existing Development Certificate.

Juan-Carlos Cano Escriba
Expires: Jul 12, 2011

Do you need to revoke this certificate and create a new one? Visit the Certificates section in the Provisioning Portal.

Figure 142. Confirm your certificate.

35. To create a Provisioning Profile, just give it a name. It will also display information on the App ID created, device assigned and certificate created for the developer to verify.



Name your Provisioning Profile

A Provisioning Profile is a collection of your App ID, Apple device UDID, and iPhone Development Certificate that allows you to install your application on your device.

Enter a common name or description for your Provisioning Profile using alphanumeric characters. The description you specify will be used throughout the Provisioning Portal to identify this Provisioning Profile.

Profile Description: myprofile

You cannot use special characters such as @, &, *, " in your description.

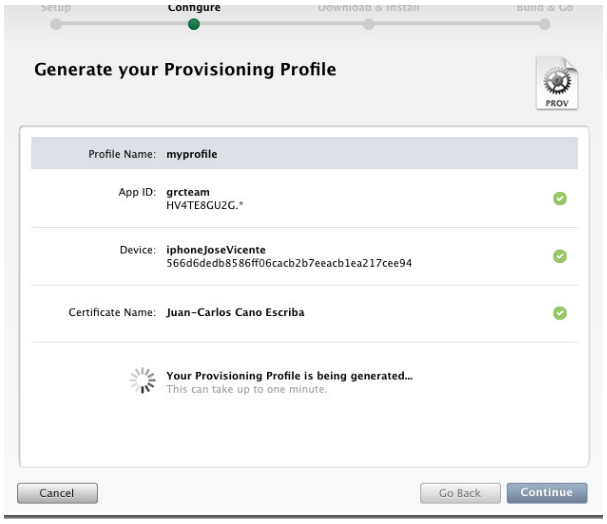
App ID: grcteam
HV4TE8GU2G.*

Device: iphoneJoseVicente
566d6dedb8586ff06cacb2b7eeacb1ea217cee94

Certificate Name: Juan-Carlos Cano Escriba

Figure 143. Create provisioning profile.

36. A process of generation begins. When it is finished, a view with all the data is shown.



Generate your Provisioning Profile

Profile Name: myprofile

App ID: grcteam HV4TE8GU2G.*

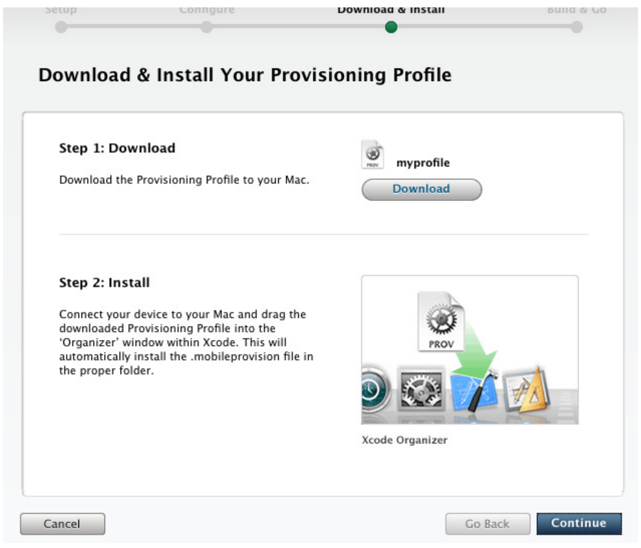
Device: iphoneJoseVicente 566d6dedb8586ff06cacb2b7eeacb1ea217cee94

Certificate Name: Juan-Carlos Cano Escriba

Your Provisioning Profile is being generated...
This can take up to one minute.

Figure 144. Confirm data.

37. Then you should download the profile to your Mac and install it in your device. Just follow the instructions that appear in the window.



Download & Install Your Provisioning Profile

Step 1: Download

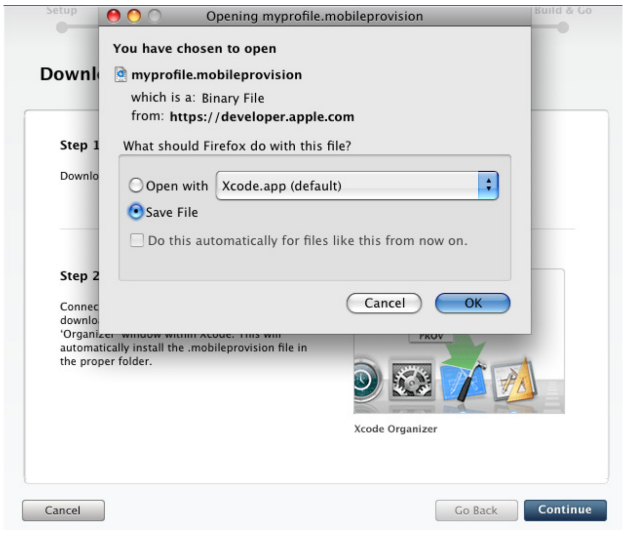
Download the Provisioning Profile to your Mac.

Step 2: Install

Connect your device to your Mac and drag the downloaded Provisioning Profile into the 'Organizer' window within Xcode. This will automatically install the .mobileprovision file in the proper folder.

Figure 145. Download and install the profile.

38. Push Download, and follow the instructions.



Opening myprofile.mobileprovision

You have chosen to open

myprofile.mobileprovision

which is a: Binary File
from: https://developer.apple.com

What should Firefox do with this file?

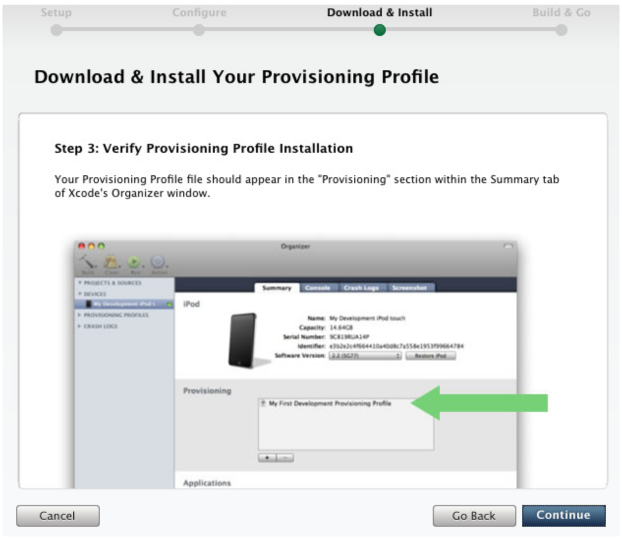
( ) Open with Xcode.app (default)
(•) Save File

[ ] Do this automatically for files like this from now on.

Cancel OK

Figure 146. Downloading the profile.

39. A new window asking you to verify the Provisioning Profile installation. Do what you're asked to. Read it carefully.



Download & Install Your Provisioning Profile

Step 3: Verify Provisioning Profile Installation

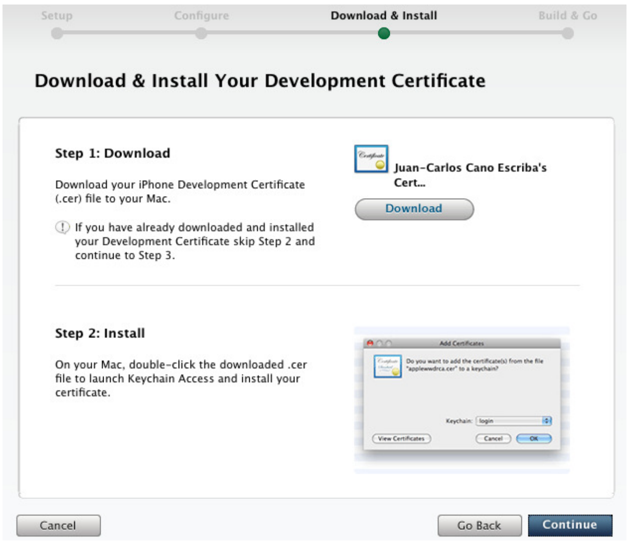
Your Provisioning Profile file should appear in the "Provisioning" section within the Summary tab of Xcode's Organizer window.

Name: My Development iPod touch
Capacity: 14.64GB
Serial Number: 9C819RUA14P
Identifier: e3b2e2c4f664410a40d8c7a558e1953f99664784
Software Version: 2.2 (5G77)

My First Development Provisioning Profile

Figure 147. Installing provisioning profile.

40. Now you're asked to Download and Install your Development Certificate. Once more do what you're asked to.



Download & Install Your Development Certificate

Step 1: Download

Download your iPhone Development Certificate (.cer) file to your Mac.

If you have already downloaded and installed your Development Certificate skip Step 2 and continue to Step 3.

Juan-Carlos Cano Escriba's Cert...

Step 2: Install

On your Mac, double-click the downloaded .cer file to launch Keychain Access and install your certificate.

Add Certificates

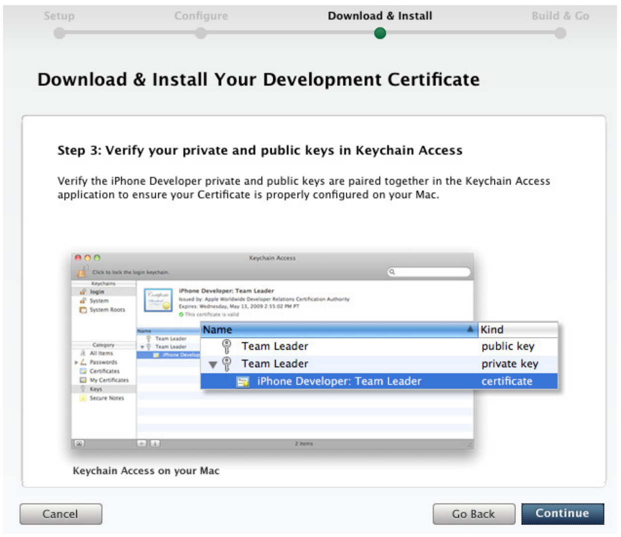
Do you want to add the certificate(s) from the file "applewwdrca.cer" to a keychain?

Keychain: login

View Certificates · Cancel · OK

Figure 148. Download and install development certificate.

After clicking "Download", the following window will appear. It is important to make this verification in the appropriate manner.



Download & Install Your Development Certificate

Step 3: Verify your private and public keys in Keychain Access

Verify the iPhone Developer private and public keys are paired together in the Keychain Access application to ensure your Certificate is properly configured on your Mac.

Keychain Access

Click to lock the login keychain.

Keychains: login, System, System Roots

Category: All Items, Passwords, Certificates, My Certificates, Keys, Secure Notes

iPhone Developer: Team Leader
Issued by: Apple Worldwide Developer Relations Certification Authority
Expires: Wednesday, May 13, 2009 2:55:02 PM PT
This certificate is valid

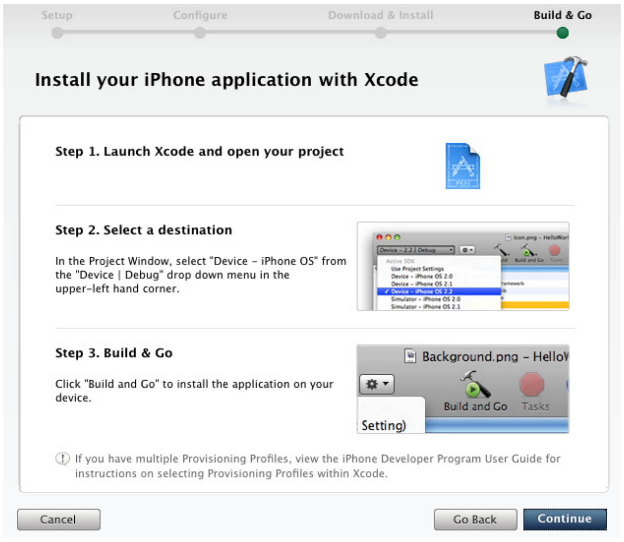
Name	Kind
Team Leader	public key
Team Leader	private key
iPhone Developer: Team Leader	certificate

2 items

Keychain Access on your Mac

Figure 149. Verifying private and public keys.

41. Information on how to load your application on the device is shown. Read it carefully. This step is going to be repeated several times until your application is completely tested.



Install your iPhone application with Xcode

Step 1. Launch Xcode and open your project

Step 2. Select a destination

In the Project Window, select "Device – iPhone OS" from the "Device | Debug" drop down menu in the upper-left hand corner.

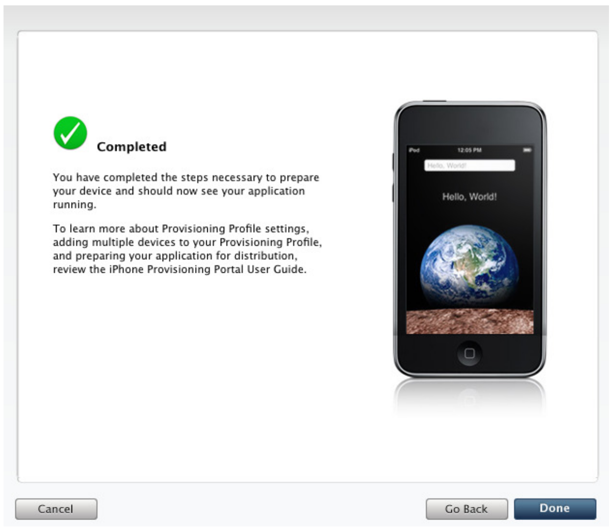
Step 3. Build & Go

Click "Build and Go" to install the application on your device.

If you have multiple Provisioning Profiles, view the iPhone Developer Program User Guide for instructions on selecting Provisioning Profiles within Xcode.

Figure 150. Installing the app with Xcode.

It tells you to launch Xcode, open your project, select “Device-Release” and click *Build & Go*. This will change the target from the simulator to the physical device. After clicking *Build & Go*, and waiting a few seconds (it depends on your project’s size) a new icon will appear in your iPhone device. Tap on it and your application should run.



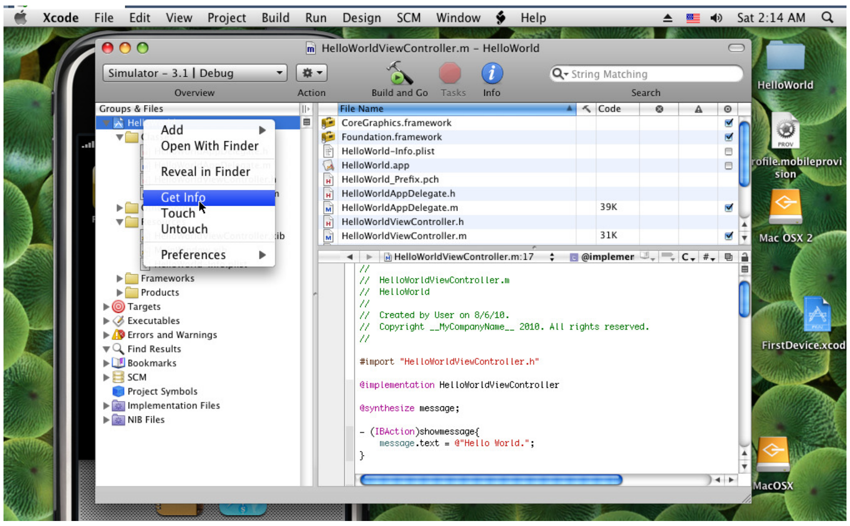
Completed

You have completed the steps necessary to prepare your device and should now see your application running.

To learn more about Provisioning Profile settings, adding multiple devices to your Provisioning Profile, and preparing your application for distribution, review the iPhone Provisioning Portal User Guide.

Figure 151. Process completed.

Something is missing between steps 40 and 41, that the wizard seems to forget.
You should, configure a few parameters before clicking Build & Go.
It is shown in the following figures.
Click the project name with the right mouse button and select Get Info. Follow the instructions.



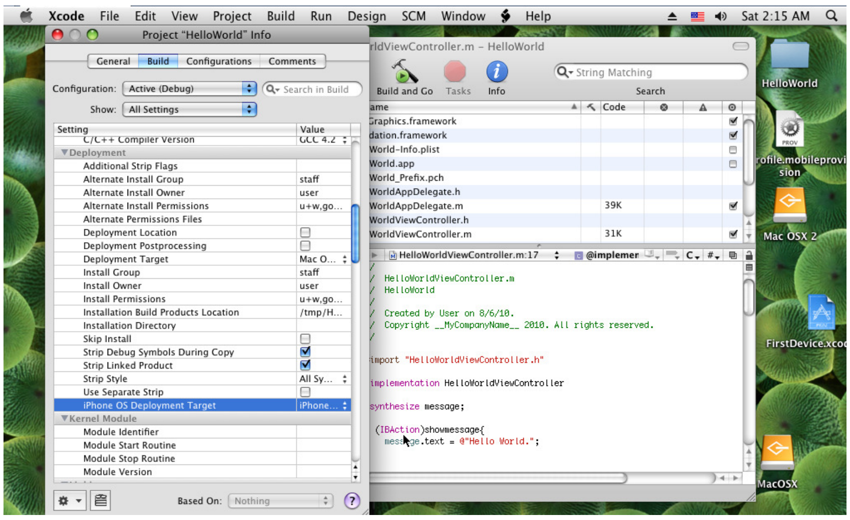
File Name	Code
CoreGraphics.framework	
Foundation.framework	
HelloWorld-Info.plist	
HelloWorld.app	
HelloWorld_Prefix.pch	
HelloWorldAppDelegate.h	
HelloWorldAppDelegate.m	39K
HelloWorldViewController.h	
HelloWorldViewController.m	31K

HelloWorldViewController.m:17 @implemen

```
//
//  HelloWorldViewController.m
//  HelloWorld
//
//  Created by User on 8/6/10.
//  Copyright __MyCompanyName__ 2010. All rights reserved.
//

#import "HelloWorldViewController.h"

@implementation HelloWorldViewController

@synthesize message;

- (IBAction)showmessage{
    message.text = @"Hello World.";
}
```

Figure 152. Configuring parameters.

Select the iPhone OS Deployment target, that is, the operating system version.



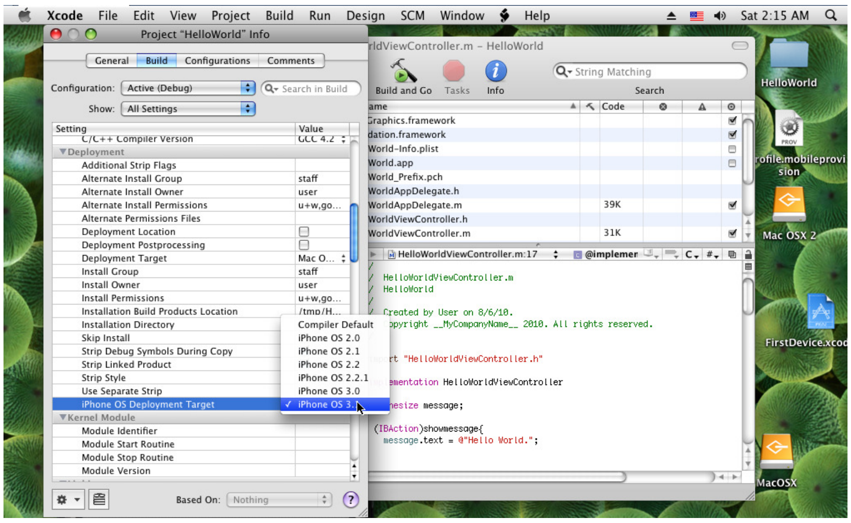
Configuration: Active (Debug)
Show: All Settings

Setting	Value
C/C++ Compiler Version	GCC 4.2
▼Deployment	
Additional Strip Flags	
Alternate Install Group	staff
Alternate Install Owner	user
Alternate Install Permissions	u+w,go…
Alternate Permissions Files	
Deployment Location	☐
Deployment Postprocessing	☐
Deployment Target	Mac O…
Install Group	staff
Install Owner	user
Install Permissions	u+w,go…
Installation Build Products Location	/tmp/H…
Installation Directory	
Skip Install	☐
Strip Debug Symbols During Copy	✓
Strip Linked Product	✓
Strip Style	All Sy…
Use Separate Strip	☐
iPhone OS Deployment Target	iPhone…
▼Kernel Module	
Module Identifier	
Module Start Routine	
Module Stop Routine	
Module Version	

Based On: Nothing

Figure 153. Selecting iOS version.

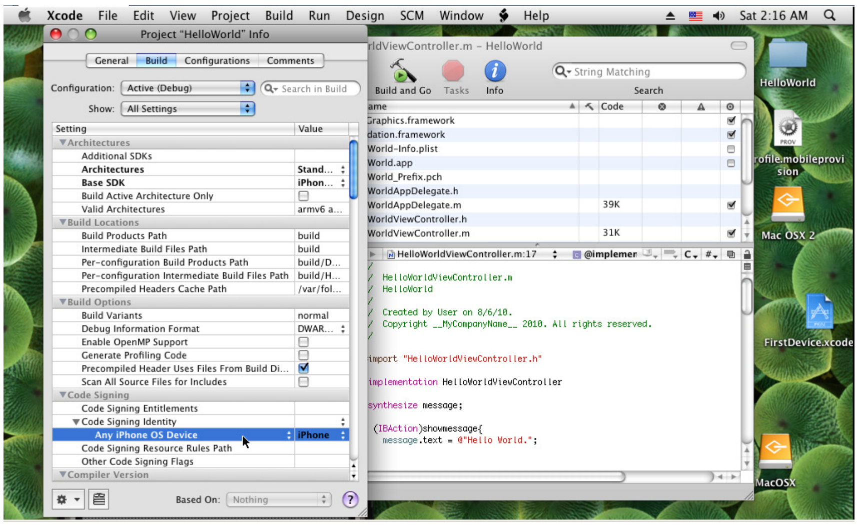
In this case, iPhone OS 3.1 is selected.



Setting	Value
C/C++ Compiler Version	GCC 4.2
▼Deployment	
Additional Strip Flags	
Alternate Install Group	staff
Alternate Install Owner	user
Alternate Install Permissions	u+w,go...
Alternate Permissions Files	
Deployment Location	☐
Deployment Postprocessing	☐
Deployment Target	Mac O...
Install Group	staff
Install Owner	user
Install Permissions	u+w,go...
Installation Build Products Location	/tmp/H...
Installation Directory	
Skip Install	
Strip Debug Symbols During Copy	
Strip Linked Product	
Strip Style	
Use Separate Strip	
iPhone OS Deployment Target	✓ iPhone OS 3.1
▼Kernel Module	
Module Identifier	
Module Start Routine	
Module Stop Routine	
Module Version	

Figure 154. iOS version.

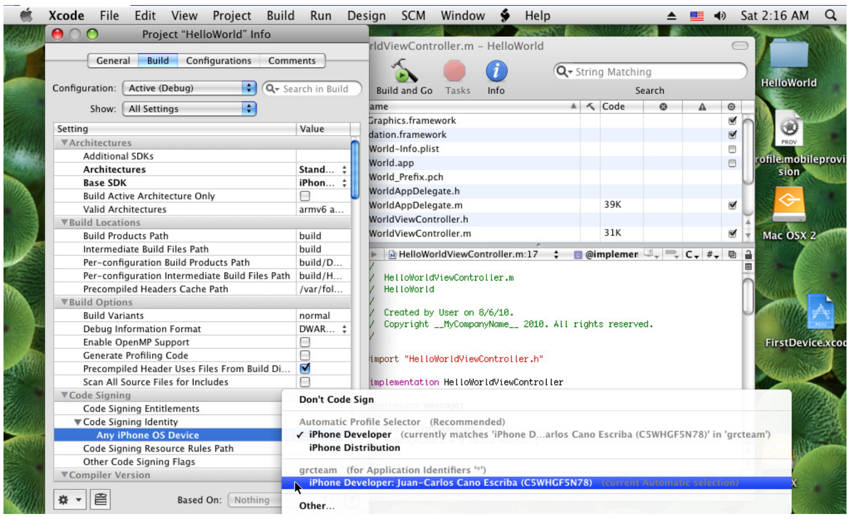
The following options are particularly interesting because it tells Xcode which is the certificate it must use.


Xcode File Edit View Project Build Run Design SCM Window Help
Sat 2:16 AM
Project "HelloWorld" Info
General Build Configurations Comments
Configuration: Active (Debug)
Show: All Settings
Search in Build
Based On: Nothing

Setting	Value
▼Architectures	
Additional SDKs	
Architectures	**Stand...**
Base SDK	**iPhon...**
Build Active Architecture Only	☐
Valid Architectures	armv6 a...
▼Build Locations	
Build Products Path	build
Intermediate Build Files Path	build
Per-configuration Build Products Path	build/D...
Per-configuration Intermediate Build Files Path	build/H...
Precompiled Headers Cache Path	/var/fol...
▼Build Options	
Build Variants	normal
Debug Information Format	DWAR...
Enable OpenMP Support	☐
Generate Profiling Code	☐
Precompiled Header Uses Files From Build Di...	☑
Scan All Source Files for Includes	☐
▼Code Signing	
Code Signing Entitlements	
▼Code Signing Identity	
Any iPhone OS Device	iPhone
Code Signing Resource Rules Path	
Other Code Signing Flags	
▼Compiler Version	

Figure 155. Selecting certificate.

In this case, we select *iPhone Developer Juan Carlos Cano Escribá*, which is the name of the certificate we previously created.


Xcode File Edit View Project Build Run Design SCM Window Help Sat 2:16 AM
Project "HelloWorld" Info
General Build Configurations Comments
Build and Go Tasks Info
HelloWorld
Mac OSX 2
FirstDevice.xcod

Configuration: Active (Debug)
Show: All Settings

Setting	Value
▼Architectures	
Additional SDKs	
Architectures	Stand…
Base SDK	iPhon…
Build Active Architecture Only	☐
Valid Architectures	armv6 a…
▼Build Locations	
Build Products Path	build
Intermediate Build Files Path	build
Per-configuration Build Products Path	build/D…
Per-configuration Intermediate Build Files Path	build/H…
Precompiled Headers Cache Path	/var/fol…
▼Build Options	
Build Variants	normal
Debug Information Format	DWAR…
Enable OpenMP Support	☐
Generate Profiling Code	☐
Precompiled Header Uses Files From Build Di…	✓
Scan All Source Files for Includes	☐
▼Code Signing	
Code Signing Entitlements	
▼Code Signing Identity	
Any iPhone OS Device	
Code Signing Resource Rules Path	
Other Code Signing Flags	
▼Compiler Version	

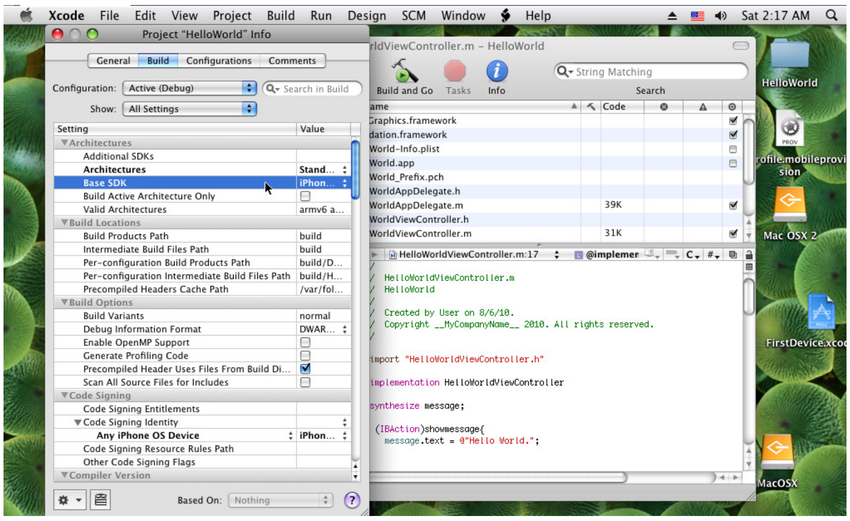
Based On: Nothing

```
Don't Code Sign

Automatic Profile Selector (Recommended)
✓ iPhone Developer (currently matches 'iPhone D...arlos Cano Escriba (C5WHGF5N78)' in 'grcteam')
  iPhone Distribution

grcteam (for Application Identifiers '*')
  iPhone Developer: Juan-Carlos Cano Escriba (C5WHGF5N78) (current Automatic selection)

Other...
```

```
HelloWorldViewController.m
HelloWorld

Created by User on 8/6/10.
Copyright __MyCompanyName__ 2010. All rights reserved.

import "HelloWorldViewController.h"

implementation HelloWorldViewController
```

Figure 156. Certificate to use.

Then you should change an Architecture option; in particular Base SDK.


Xcode File Edit View Project Build Run Design SCM Window Help Sat 2:17 AM
Project "HelloWorld" Info
General Build Configurations Comments
Build and Go Tasks Info
HelloWorld
Mac OSX 2
FirstDevice.xcod
MacOSX

Configuration: Active (Debug)
Show: All Settings

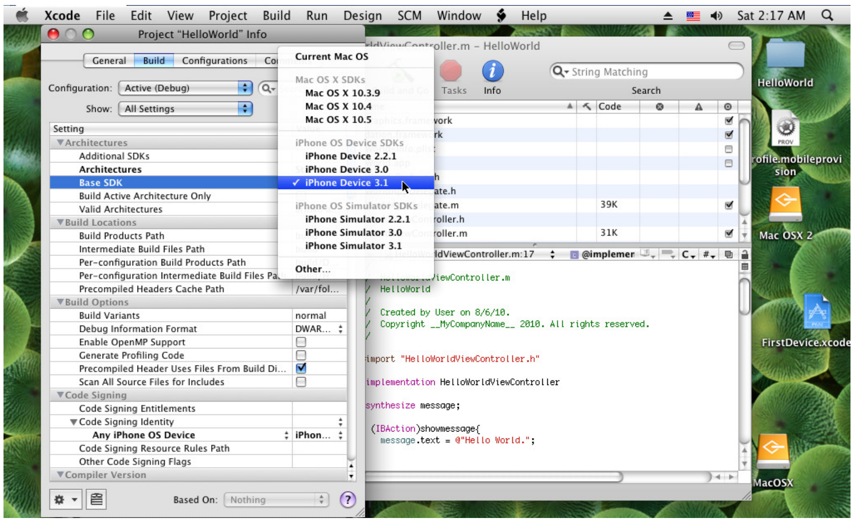
Setting	Value
▼Architectures	
Additional SDKs	
Architectures	Stand…
Base SDK	iPhon…
Build Active Architecture Only	☐
Valid Architectures	armv6 a…
▼Build Locations	
Build Products Path	build
Intermediate Build Files Path	build
Per-configuration Build Products Path	build/D…
Per-configuration Intermediate Build Files Path	build/H…
Precompiled Headers Cache Path	/var/fol…
▼Build Options	
Build Variants	normal
Debug Information Format	DWAR…
Enable OpenMP Support	☐
Generate Profiling Code	☐
Precompiled Header Uses Files From Build Di…	✓
Scan All Source Files for Includes	☐
▼Code Signing	
Code Signing Entitlements	
▼Code Signing Identity	
Any iPhone OS Device	iPhon…
Code Signing Resource Rules Path	
Other Code Signing Flags	
▼Compiler Version	

Based On: Nothing

```
HelloWorldViewController.m
HelloWorld

Created by User on 8/6/10.
Copyright __MyCompanyName__ 2010. All rights reserved.

import "HelloWorldViewController.h"

implementation HelloWorldViewController

synthesize message;

 (IBAction)showmessage{
  message.text = @"Hello World.";
```

Figure 157. Change Base SDK in Architectures.

You ought to choose the iPhone OS Device. In this case, iPhone Device 3.1 is selected.



Configuration: Active (Debug)
Show: All Settings

Setting	
▼Architectures	
Additional SDKs	
Architectures	
Base SDK	✓ iPhone Device 3.1
Build Active Architecture Only	
Valid Architectures	
▼Build Locations	
Build Products Path	
Intermediate Build Files Path	
Per-configuration Build Products Path	
Per-configuration Intermediate Build Files Pa	
Precompiled Headers Cache Path	/var/fol...
▼Build Options	
Build Variants	normal
Debug Information Format	DWAR...
Enable OpenMP Support	☐
Generate Profiling Code	☐
Precompiled Header Uses Files From Build Di...	✓
Scan All Source Files for Includes	☐
▼Code Signing	
Code Signing Entitlements	
▼Code Signing Identity	
Any iPhone OS Device	iPhon...
Code Signing Resource Rules Path	
Other Code Signing Flags	
▼Compiler Version	

Based On: Nothing

ldViewController.m – HelloWorld

ldViewController.m:17 @implemen

```
/  HelloWorldViewController.m
/  HelloWorld
/
/  Created by User on 8/6/10.
/  Copyright __MyCompanyName__ 2010. All rights reserved.
/

import "HelloWorldViewController.h"

implementation HelloWorldViewController

synthesize message;

 (IBAction)showmessage{
   message.text = @"Hello World.";
```

Figure 158. iPhone Device version.

After following these steps, your application is ready to be used in the iPhone device.

2. Distributing the app

After having tested our application, we may want to distribute it to a wide range of users. So we must send the application to Apple, that will evaluate if the application is suitable for distributing it. Basically, they analyze that the application does not crack down, that the application really does what the developer says it does, verify that the code follows some of their restrictions and a few things more. After receiving Apple's authorization, the app is ready to be transferred to the AppStore, where all the users can see and use. We can even distribute it for free or fix a price. It's up to the developer.

To be aware of the process to submit your application to AppStore, you should visit the program portal in Apple iPhone developer web site. There is plenty of information available to submit your application once it is completely tested and approved by Apple.

Open your Xcode project and check that you've set the active SDK to one of the device choices, like Device – 3.1. Next, you should choose a build configuration that uses your distribution (not your developer) certificate. Double-click on your target in the Groups & Files column on the left of the project window. The Target Info window will open. Click the Build tab and review your Code Signing Identity. It should be iPhone Distribution: followed by your name or company name.

Check your application identifier in the Properties tab.

The top-left of your project window also confirms your settings and configuration. You should have selected "Device – 3.1 | Distribution". This shows you the active SDK and configuration.
If your settings are correct but you aren't getting that upload finished properly, clean your builds. Choose Build > Clean (Command-Shift-K) and click Clean. Alternatively, you can manually trash the build folder in your Project from Finder. Once you've cleaned, build again.
This should produce an app that when zipped properly loads to iTunes Connect.

After concluding all the steps shown in the iPhone developer website, your application will be ready to be distributed using AppStore, a shared place by means of which all iPhone owners can see and purchase all the applications available.