Rutes amb iOS

Aplicació per registrar i gestionar recorreguts amb GPS basada al sistema operatiu mòbil iOS amb sincronització a la web.

Pedro Nofuentes Azcárate

Enginyeria Informàtica

Consultor: Jordi Ceballos Villach

30/05/2011

A Noelia, per ser ella amb mi.
A Maite i Carmelo, els meus pares, per ser sempre un punt de llum inclús als moments més foscos.

Resum

Les plataformes que presten serveis a Internet estan suposant una canvi de paradigma sobre la computació, donant el que s'ha anomenat la computació al núvol. Aquesta computació planteja el pas de les operacions i les dades des d'un sistema local a un sistema de servidors amb el què l'usuari interacciona des d'un terminal, podent ser aquest un ordinador amb navegador, un dispositiu mòbil, un sistema encastat, etc.

Una arquitectura desenvolupada per donar resposta a la major demanda de recursos dels servidors és l'anomenada plataforma com a servei. Aquesta arquitectura no ofereix un recurs com un servidor o un servidor virtual, sinó que permet canviar dinàmicament el nombre de recursos segons la necessitat puntual, afegint o minvant capacitat. Exemples són la plataforma Goolge App Engine o els serveis d'Amazon Web Services[1] .

Per altra banda l'evolució dels dels dispositius mòbils ha suposat un augment de les seues capacitats sobretot donada per l'evolució dels sistemes operatius. En concret l'aparició d'iOS va suposar una evolució cap al major aprofitament i desenvolupament d'aplicacions que estendrien les funcionalitats del dispositiu.

El projecte que es presenta pretén implementar un sistema que aporte funcionalitats a través d'un dispositiu mòbil a un usuari i que puga sincronitzar-ho a un servidor, donant així la possibilitat de consultar les dades a través d'altra interfície.

El sistema implementat consta de:

- Una aplicació mòbil per enregistrar recorreguts amb el GPS del dispositiu, visualitzar-los, modificar-los i sincronitzar-los amb el servidor.
- Una aplicació web que permet rebre les dades del dispositiu, emmagatzemar-les, mostrar-les i editar-les.

L'aplicació mòbil es desenvolupa sobre la plataforma iOS amb el llenguatge iOS i l'aplicació web sobre Google App Engine amb el llenguatge Python.

[1] http://aws.amazon.com/es/

Índex

1. Introducció

1.1 Justificació del projecte

El Projecte Final de Carrera es desenvolupa dins l'àmbit del desenvolupament d'aplicacions per a dispositius mòbils. El projecte pretén plasmar tot el cicle d'implementació d'una solució dividint-lo en tres parts: anàlisi, disseny i implementació, així com aplicar els coneixements adquirits a les diferents assignatures cursades.

L'objecte és desenvolupar una aplicació mòbil que faça ús de les característiques més comunes de l'entorn de programació com és la gestió de vistes, permanència de les dades, localització i connexió a Internet. Per últim permetre serialitzar les dades i sincronitzar amb la pàgina web a un servidor públic.

El nucli principal del Projecte consisteix en l'aplicació mòbil Corre. Aquesta permet a l'usuari les següents funcionalitats:

- Enregistrar un recorregut mitjançant el GPS[2] del seu dispositiu
- Emmagatzemar les dades enregistrades.
- Consultar les dades, tant estadístiques com el mapa amb la ruta realitzada.
- Modificar i esborrar recorreguts.
- Establir preferències d'enregistrament i sincronització amb la pàgina web.

Per altra banda es desenvolupa una pàgina web que permet:

- Sincronitzar els recorreguts enregistrats pel dispositiu.
- Visualitzar les dades dels recorreguts sincronitzades.
- Modificar i esborrar recorreguts.

1.2 Descripció del projecte

L'aplicació ha de reigistrar els recorreguts d'un usuari emprant un dispositiu mòbil i el *GPS* incorporat, donant les funcionalitats necessàries per gestionar-los. A més, es podrà sincronitzar amb una pàgina web que el mostre al navegador de l'usuari en un format major. El seu ús típic potser el següent.

L'usuari descarrega l'aplicació al seu dispositiu per poder-la emprar. Decideix realitzar un recorregut i enregistrar-lo perquè vol saber les distàncies que ha fet, temps emprat, altres dades d'interès i afegir alguna nota del seu interès.

Per començar obre l'aplicació i tria registrar un nou recorregut. Quan està preparat, polsa el botó d'inici i el dispositiu comença, mitjançant el seu *GPS* integrat, a enregistrar els punts pels que passa l'usuari. En finalitzar aquest polsa el botó de fi i el recorregut queda enregistrat.

L'usuari podrà consultar el traçat mostrat sobre un mapa amb les dades de temps emprat, dades segons un interval definit, altituds, etc. Si vol podrà modificar dades com el nom que li vol assignar al recorregut, afegir notes o altres dades d'interès.

---

[2] *Global Positioning System*: sistema de localització que mitjançant la triangulació de la senyal amb els satèl·lits pot establir la posició d'un objecte.

Per últim tindrà la possibilitat de sincronitzar els seus recorreguts amb una pàgina web on s'emmagatzemaran per poder consultar-los a una interfície major, el seu navegador, i més còmoda. També podrà editar dades que seran sincronitzades amb el dispositiu. L'accés a la pàgina serà mitjançant un usuari i contrasenya per protegir les seves dades.

1.2.1 Funcionalitats

En funció de l'ús descrit anteriorment les funcionalitats de les què disposarà el sistema són:

- Aplicació mòbil:
  - Enregistrament de rutes mitjançant el GPS, amb la possibilitat d'enregistrar punts al llarg de la ruta cada interval definit prèviament.
  - Presentació de les rutes a un mapa amb els punts ressaltats: inici, fi i punts intermedis.
  - Presentació de dades d'interès com el temps, altitud, distància, etc. i comparació d'aquestes amb recorreguts anteriors per la mateixa ruta.
  - Emmagatzemament i gestió dels recorreguts realitzats i les seves dades associades.
  - Registre i accés mitjançant usuari i contrasenya a l'aplicació web per sincronitzar les dades.
  - Sincronització amb l'aplicació web dels recorreguts.
- Aplicació web:
  - Enregistrament i accés a les dades de l'usuari.
  - Presentació dels recorreguts sincronitzats a un mapa i les dades associades.
  - Modificació de dades del recorregut.
  - Sincronització dels canvis amb l'aplicació mòbil.

1.2.2 Tecnologies

L'aplicació es desenvoluparà per l'iPhone[3] amb el seu sistema operatiu iOS[4]. La pàgina web, entre les múltiples opcions possibles, es desenvoluparà per ser desplegada al servei Google App Engine[5]. Aquest servei és gratuït fins un límit de consum de recursos i permet el desenvolupament de planes web amb el llenguatge Java[6] o Python[7]. A aquest cas

[3] http://www.apple.com/es/iphone/

[4] http://www.apple.com/es/iphone/ios4/

[5] http://code.google.com/intl/es-ES/appengine/docs/

[6] Llenguatge de programació orientat a objectes desenvolupat per Sun. La seva principal característica és la portabilitat donat que s'executa sobre una capa entre la màquina i el codi.

[7] Llenguatge d'alt nivell amb l'objectiu d'una sintaxi clara. El llenguatge suporta orientació a objectes, programació imperativa i inclús alguns elements de programació funcional.

s'emprarà el llenguatge Python i el *framework* Django[8], que facilita realitzar el desenvolupament més ràpid donat que evita desenvolupar parts bàsiques i reutilitzar-les.

1.3 Objectius del projecte

Amb el desenvolupament d'aquest projecte es volen assolir els següents objectius:

- Desenvolupar una aplicació per un dispositiu mòbil basat en iOS.
- Aprofundir al llenguatge de programació Objective-C[9], el *framework* Cocoa Touch[10], la seva arquitectura i els seus patrons de programació.
- Aprofundir als paradigmes de construcció d'aplicacions web amb sincronització als dispositius mòbils.

A més d'aquests objectius també es persegueix l'objectiu transversal d'aplicar els coneixements adquirits al llarg de la carrera als següents:

- Disseny amb l'aplicació de patrons, com per exemple el MVC[11].
- Protocol de comunicació entre dispositiu i web.
- Transformació de dades a documents XML.
- Planificació, seguiment i gestió d'un projecte informàtic d'inici a fi.
- Configuració d'un entorn de treball que permeta aplicar paràmetres de qualitat al desenvolupament del projecte (control de versions, seguiment del projecte, etc.).

1.4 Planificació

El projecte consisteix en desenvolupar una aplicació mòbil i una altra web. Els objectius i els lliurables estan clars d'inici. Donat el poc temps d'execució i les característiques del projecte s'aplicarà un cicle de vida en cascada.

El cicle de vida en cascada per a un projecte informàtic consta de les següents fases:

1. Definició de requisits.
2. Anàlisi i disseny del sistema.
3. Implementació.
4. Integració i proves.
5. Explotació i manteniment.

---

[8] http://www.djangoproject.com/ - Pàgina web del *framework* Django.
S'ha de resaltar que s'emprarà una modificació d'aquest per poder treballar amb base de dades no relacionals. *Google App Engine* empra una tecnologia de base de dades que anomenen *Big Table*. La modificació és *Django-nonrel* (http://www.allbuttonspressed.com/projects/django-nonrel).

[9] Llenguatge de programació que és un superconjunt del llenguatge C emprat com a llenguatge principal a Mac OS X i iOS.

[10] És un conjunt de llibreries que conté les funcionalitats per emprar les característiques dels dispositius mòbils d'Apple (http://developer.apple.com/technologies/ios/cocoa-touch.html).

[11] Model View Controller: patró de disseny que estableix tres capes que encapsulen les funcionalitats relacionades.

Amb l'objectiu de particularitzar aquest cicle general al nostre projecte s'afegeixen dues fases i es lleva la fase d'Explotació i manteniment. Per tant el projecte tindrà les següents fases:

1. Planificació del projecte.
2. Definició de requisits.
3. Anàlisi i disseny del sistema.
4. Implementació.
5. Integració i proves.
6. Documentació.

1.4.1 Fases del projecte

Planificació del projecte

A la fase de planificació de projecte es realitzaran les tasques per establir el pla d'execució del projecte així com la preparació de l'entorn de treball.

Tasca	Data inici	Data fi	Descripció
Definició projecte	07/03/2011	08/03/2011	Establir l'abast i objectius del projecte.
Preparació de l'entorn	09/03/2011	11/03/2011	Instal·lar les ferramentes necessàries per l'execució del projecte: IDE, gestor de projecte, SDK, etc.
Planificació	09/03/2011	11/03/2011	Establir les tasques i les dates de realització.
Preparació de documentació	12/03/2011	13/03/2011	Redacció del document per al lliurament.
LLiurament PAC1	14/03/2011	14/03/2011	Lliurament del pla de treball.

Definició de requisits

A la definició de requisits es realitzaran les tasques necessàries per establir les funcionalitats del sistema i com s'ha de comportar.

Tasca	Data inici	Data fi	Descripció
Definició de les funcionalitats	15/03/2011	16/03/2011	Establir les funcionalitats que el sistema aportarà a l'usuari.

Anàlisi i disseny del sistema

A aquesta fase es definirà el comportament de l'aplicació en base als requisits, així com el model de dades i la seua interfície.

Tasca	Data inici	Data fi	Descripció
Disseny Casos d'ús	18/03/2011	21/03/2011	Definició de les diferents accions que es poden realitzar partint dels requisits.

Tasca	Data inici	Data fi	Descripció
Disseny Arquitectura	22/03/2011	23/03/2011	Definició de l'arquitectura del sistema.
Disseny Model de dades	24/03/2011	26/03/2011	Definició del model de dades que emmagatzemarà les dades del sistema.
Disseny Interfície	27/03/2011	01/04/2011	Elaboració del diagrama d'estats de les aplicacions i maquetacions de la disposició de la interfície d'usuari.
Preparació de documentació	02/04/2011	03/04/2011	Redacció del document per al lliurament.
LLiurament PAC2	04/04/2011	04/04/2011	Lliurament del disseny del sistema.

Implementació

A la fase d'implementació es realitzarà la codificació de l'aplicació mòbil i la pàgina web per la seua sincronització.

Tasca	Data inici	Data fi	Descripció
Implementació de l'aplicació mòbil	05/04/2011	24/04/2011	Codificació de l'aplicació mòbil.
Implementació de la pàgina web	25/04/2011	04/05/2011	Codificació de la pàgina web.

Integració i proves

A la fase d'integració es realitzaran les tasques per acoblar les diferents parts del projecte i realitzar les proves necessàries per comprovar el seu funcionament correcte.

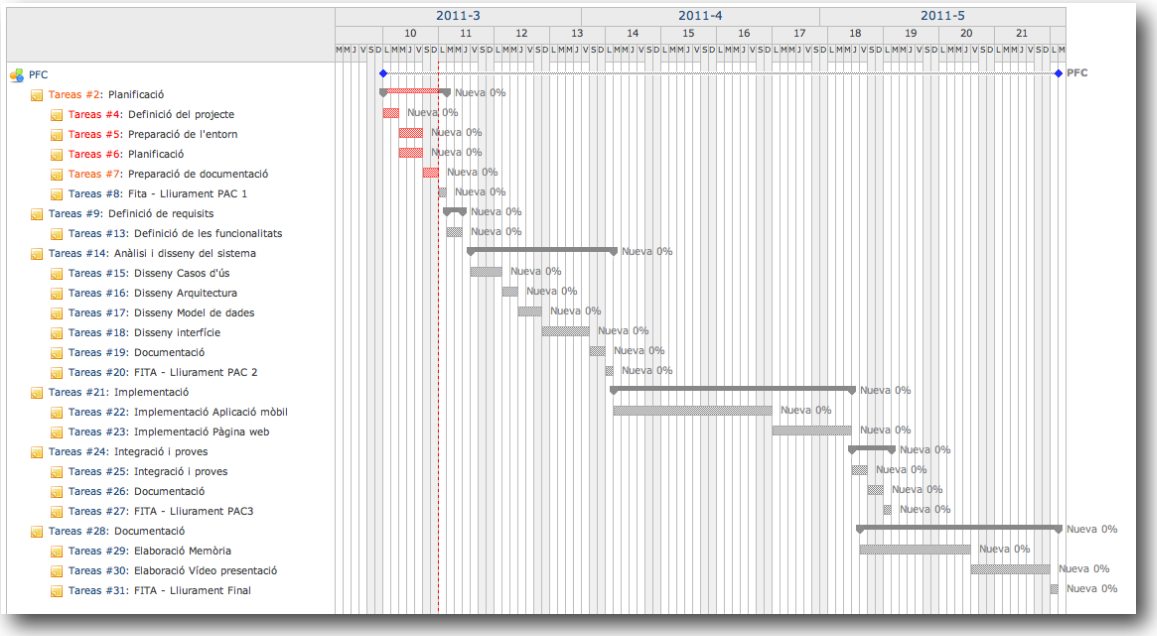
Tasca	Data inici	Data fi	Descripció
Integració i proves	05/05/2011	06/05/2011	Integració del sistema i proves.
Preparació de la documentació	07/05/2011	08/05/2011	Redacció del document per al lliurament.
LLiurament PAC3	09/05/2011	09/05/2011	Lliurament de la implementació del sistema.

Documentació

Es prepararà la memòria final i el vídeo presentació del projecte.

Tasca	Data inici	Data fi	Descripció
Elaboració Memòria	06/05/2011	19/05/2011	Recollida de la documentació i redacció de la memòria.
Elaboració Vídeo presentació	20/05/2011	29/05/2011	Gravació i maquetació del vídeo.
LLiurament Final	30/05/2011	30/05/2011	Lliurament final del projecte.

1.4.2 Diagrama de Gantt



1.5 Riscos

De la planificació del projecte no es desprenen gaires riscos. Encara que hi ha dependències entre parts del projecte, no hi ha grups de treball concurrents on es puguen desaprofitar recursos o que s'endarrerisquen entre grups de treball.

Risc	Impacte	Probabilitat	Descripció
R1 - Manca de coneixement de *iOS*.	Alt	Mitja	*Cocoa Touch* no es coneix en profunditat i pot endarrerir la marxa del projecte.
R2 - Manca de temps per la implementació per la data de lliurament.	Alt	Mitja	Donada la data de lliurament el temps per la implementació del sistema pot ser escàs.

Identificats els dos principals riscos del projecte, es proposen les següents accions correctives:

Risc	Acció correctiva	Descripció
R1	AC1 - Catàleg de recursos	Elaborar un catàleg de recursos i fonts d'informació on consultar les possibles dificultats i solucions que sorgisquen.
R2	AC2 - Reduir el temps dedicat a la fase de disseny	Si és possible, realitzar les tasques de disseny sense esgotar tot el temps planificat, podent així avançar la data d'inici de la fase d'implementació.
R2	AC3 - Reduir el tems dedicat a la fase d'integració	Ambdues aplicacions s'implementen per la mateixa persona. Amb el disseny detallat es pot reduir el temps necessari per integrar els sistema.

1.6 Entorn de treball

L'entorn de treball amb el què s'ha realitzat el projecte ha sigut el següent:

1.6.1 Equip

- **Ordinador iMac (Core 2 Duo 3,06GHz; 4GB RAM):** equip principal on s'han desenvolupat les tasques del projecte.
- **iPhone 4:** dispositiu mòbil per realitzar les proves.

1.6.2 Entorn de desenvolupament

- **XCode 4:** per al desenvolupament de l'aplicació mòbil.
- **TextMate[12]:** per al desenvolupament de la pàgina web.
- **iOS 4.3 SDK:** *kit* de desenvolupament per a l'aplicació mòbil.
- **Google App Engine SDK:** *kit* de desenvolupament per a la pàgina web que crea un servidor local.

1.6.3 Gestió de projecte

A banda de les ferramentes per desenvolupar la solució, s'ha procurat emprar ferramentes auxiliars pel desenvolupament del projecte:

- **GIT[13]:** sistema de control de versions. S'emprarà per poder seguir la traçabilitat dels canvis al codi.
- **Redmine[14]:** gestor de projectes basat a Ruby on Rails que permet:
  - Planificar les tasques i recollir les seues variacions.
  - Adjuntar documents al projecte.
  - Emmagatzemar documentació tècnica i d'usuari sobre el projecte.
  - Ús d'un *wiki* incorporat.

1.7 Resultats obtinguts

Com a resultat de la realització del projecte s'han obtingut els següents productes:

- Aplicació Corre: aplicació per a l'iPhone que permet l'enregistrament de recorreguts, el seu emmagatzemament, consulta, gestió i sincronització.
- Pàgina web Corre[15]: que permet mitjançant autenticació consultar i gestionar els recorreguts sincronitzats.

[12] http://macromates.com/

[13] http://git-scm.com/

[14] http://www.redmine.org/

[15] http://correwebapp.appspot.com/

A banda dels productes també s'ha generat la documentació de les diferents fases del projecte:

- Pla de treball: document amb l'abast del projecte i planificació.
- Disseny: document amb l'anàlisi i disseny del sistema a implementar.
- Manual d'instal·lació de la solució.
- Informe explicatiu de les implementacions desenvolupades.
- Memòria final del projecte.
- Vídeo amb la presentació del projecte.

2. Anàlisi

2.1 Requisits

2.1.1 Requisits funcionals

Els requisits funcionals del sistema es divideixen entre el dispositiu mòbil i el servei web.

Dispositiu mòbil

Les funcionalitats que prestarà l'aplicació mòbil es poden agrupar en funcionalitats relacionades amb els recorreguts i gestió d'usuaris.

Gestió de recorreguts

- Crear un nou recorregut: l'usuari podrà crear un nou recorregut iniciant l'enregistrament a través del GPS. L'usuari podrà:
  - Iniciar un nou recorregut.
  - Aturar l'enregistrament.
- Consultar els recorreguts enregistrats: es mostrarà a l'usuari un llistat amb tots els recorreguts enregistrats dels què pot triar un per consultar el detall. Les dades d'un recorregut es mostraran a diferents vistes:
  - Vista de mapa: on es mostra el recorregut amb indicadors de punts relevants: principi, fi i punts intermedis.
  - Vista amb les dades del recorregut.
  - Vista per comparar amb altres recorreguts pel mateix itinerari.
- Modificar dades d'un recorregut: l'usuari podrà triar un recorregut i modificar algunes dades, com poden ser un nom personalitzat, afegir una nota, o la freqüència cardiaca del recorregut.
- Esborrar recorreguts: l'usuari podrà esborrar els recorreguts que dessitge.
- Sincronitzar les dades amb l'aplicació web.

Gestió de registre

- Identificar-se al sistema: si l'usuari ja està donat d'alta es podrà identificar mitjançant el seu nom d'usuari i contrasenya.

Preferències

- Modificar les categories dels recorreguts.
- Modificar si es volen enregistrar els punts intermedis cada cert interval i definir la distància d'aquest.
- Modificar les dades d'usuari.

Servei web

Gestió de recorreguts

- Consultar els recorreguts sincronitzats: l'usuari tindrà un llistat dels recorreguts realitzats i podrà triar un per consultar el seu detall. Les dades que es mostraran seran:
  - Vista de mapa on es mostra el recorregut amb els indicadors dels punts rellevants.
  - Vista amb les dades del recorregut.
  - Vista amb la comparació d'altres recorreguts.
- Esborrar un recorregut.

Gestió de registre

- Registrar-se al sistema: l'usuari podrà donar-se d'alta al sistema omplint unes dades bàsiques.
- Modificar les dades d'usuari.
- Donar-se de baixa

Gestió del sistema

Es crearà una interfície d'administració de l'aplicació web per poder gestionar les dades emmagatzemades en cas necessari.

Aclaracions

Algunes de les funcionalitats descrites estan duplicades a les dues aplicacions, però evidentment la seua presentació serà diferent. Encara així s'han d'aclarir alguns detalls sobre el comportament de les funcionalitats:

- Sincronització de les dades:
  - Si l'usuari modifica les dades la sincronització actualitzarà el dispositiu que tinga la versió anterior.
  - Si s'han modificat les dades a ambdós llocs es demanarà la intervenció de l'usuari per triar la versió que desitja.
  - Si s'esborra un recorregut a la sincronització s'esborrarà a l'altre component del sistema.
- Si el dispositiu no pot rebre senyal GPS es mostrarà un missatge a l'usuari comunicant-li que no es pot registrar el seu recorregut.
- Comparació de recorreguts: el sistema establirà els recorreguts similars, si un usuari vol comparar es mostrarà un llistat amb recorreguts similars per escollir amb quin es vol comparar el que s'està veient actualment.

2.1.2 Requisits no funcionals

A banda dels requisits funcionals descrits anteriorment, es vol que el projecte tinga les següents característiques:

Interfície

- A l'aplicació mòbil es seguiran els elements d'interfície propis d'iOS i les pautes d'interacció amb l'usuari proposades per Apple[16]. L'objectiu és garantir una experiència d'usuari comuna a altres aplicacions i facilitar el seu aprenentatge.
- A la web es realitzarà una interfície senzilla per facilitar l'accés a les funcionalitats i el seu ús.

Dades d'usuari

- El sistema ha de garantitzar la privadesa de les dades de l'usuari.
- S'ha de facilitar el procés d'exportar les dades de forma que l'usuari puga utilitzar-les a altres aplicacions.
- S'ha de facilitar l'opció de donar-se de baixa del sistema i esborrar totes les dades de l'usuari.

2.1.3 Identificació d'usuaris

Segons la descripció del projecte el sistema donarà servei als usuaris de l'aplicació mòbil simplement amb la seua instal·lació. Però, per accedir a la funcionalitat de sincronització amb el servei web els usuaris s'hauran d'identificar al sistema.

Per tant s'identifiquen els següents tipus d'usuari:

- **Usuari no registrat:** l'usuari no registrat és aquell que no ha realitzat el procés de registre del sistema i no s'identifica. Aquest usuari pot fer ús de les funcionalitats d'enregistrament d'activitats i gestió de recorreguts. Per últim tindrà la possibilitat d'enregistrar-se.
- **Usuari registrat:** aquest usuari podrà fer ús de la sincronització dels recorreguts a la web mitjançant la identificació al sistema. Com a usuari també podrà modificar les seues dades del sistema i donar-se de baixa.
- **Administrador:** aquest tipus d'usuari s'inclou per realitzar tasques d'administració dels usuaris registrats. Encara que aquests tindran les funcionalitats necessàries per gestionar-se a sí mateixos pot requerir-se de forma puntual necessitat de realitzar alguna gestió.

2.2 Casos d'ús

2.2.1 Diagrames

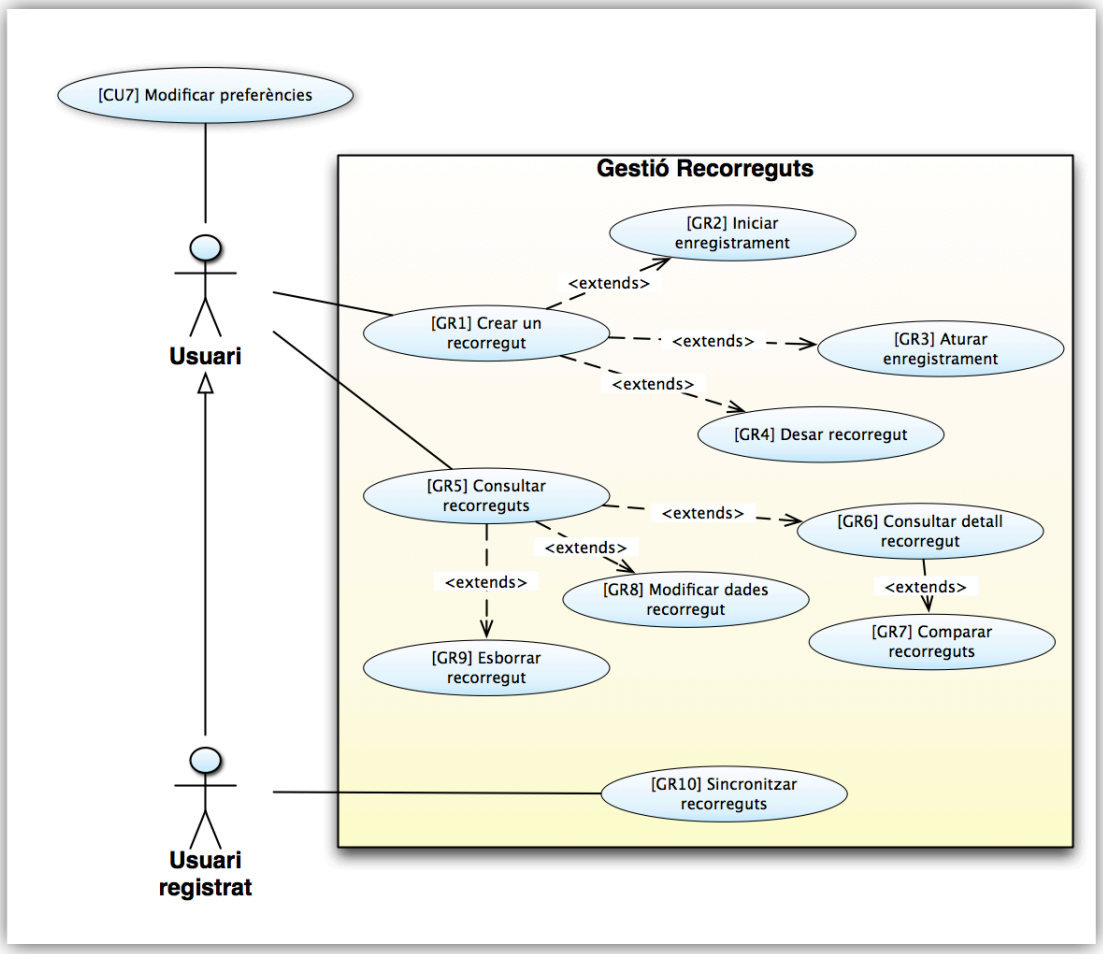
Donats els requisits del sistema s'han identificat els diferents actors i casos d'ús, els quals s'han agrupat segons el component del sistema.

Cal aclarir que la majoria de casos d'ús es donen als dos components i el seu comportament general és similar. Evidentment, a la posterior implementació s'haurà de particularitzar el seu comportament.

Dispositiu mòbil

Els casos d'ús identificats pel dispositiu mòbil són els següents:

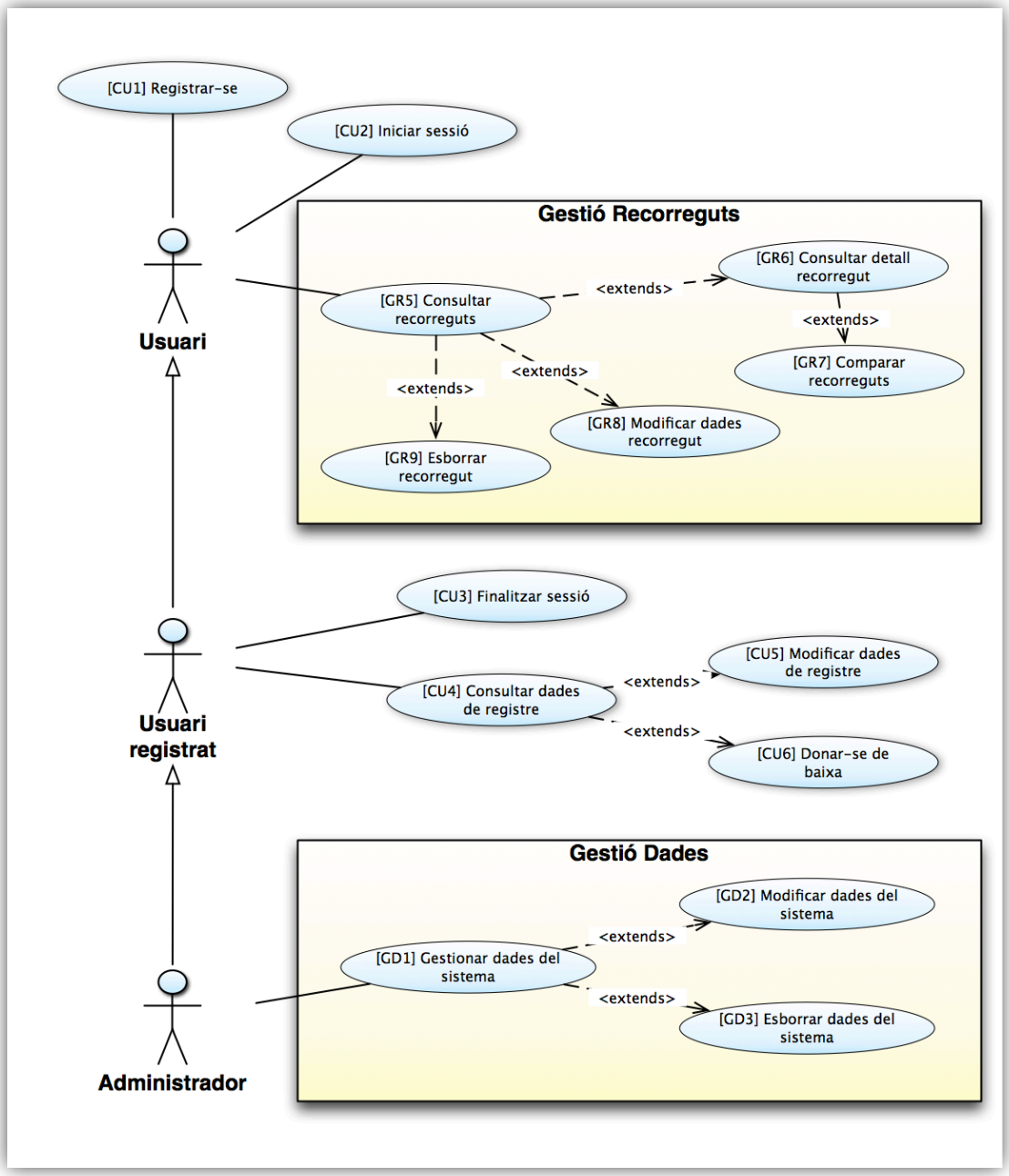
[16] iOS Human Interface Guidelines (http://developer.apple.com/library/ios/#documentation/UserExperience/Conceptual/MobileHIG/Introduction/Introduction.html)



Servei web

Els casos d'ús relatius al servei web són els següents:

(diagrama a la pàgina següent)



2.2.2 Descripció dels casos d'ús

Per realitzar la descripció detallada dels casos d'ús identificats s'han descrit els següents punts:

- **Resum:** un resum de la funcionalitat.
- **Actor(s):** actors que intervenen al cas d'ús.
- **Component(s):** component on aplica el cas d'ús.
- **Precondició:** condició que ha de complir el cas d'ús per poder dur-ho a terme.
- **Postcondició:** com queda el sistema després d'haver realitzat el cas d'ús.
- **Flux d'events:** successió de passes per realitzar la funcionalitat descrita.
- **Flux alternatiu:** en cas d'un esdeveniment inesperat què succeïx.
- **Extensions:** casos d'ús que amplien la funcionalitat del cas d'ús actual.

CU1 - Registrar-se

CU1 :: Registrar-se	
Resum	Registrar-se al sistema facilitant les dades necessàries.
Actor(s)	Usuari
Component(s)	Web
Precondició	L'usuari no està registrat prèviament.
Postcondició	Usuari enregistrat.
Flux d'events	1. L'usuari tria l'opció de registrar-se 2. L'usuari ompli les dades necessàries per registrar-se. 3. El sistema informa l'usuari que el procés ha sigut correcte.
Flux alternatiu	• Les dades no són correctes: el sistema informa i es torna al pas 2. • L'usuari cancel·la l'operació.
Extensions	No hi ha.

CU2 - Inciar sessió

CU2 :: Iniciar sessió	
Resum	Entrar al sistema com a usuari registrat.
Actor(s)	Usuari
Component(s)	Web
Precondició	Usuari sense sessió iniciada.
Postcondició	Usuari amb la sessió iniciada.
Flux d'events	1. L'usuari tria l'opció d'entrar al sistema. 2. L'usuari ompli les dades d'usuari i contrasenya. 3. El sistema informa l'usuari que el procés ha sigut correcte.
Flux alternatiu	• Les dades no són correctes: el sistema informa i es torna al pas 2. • L'usuari cancel·la l'operació.
Extensions	No hi ha.

CU3 - Finalitzar sessió

CU3 :: Finalitzar sessió	
Resum	Finalitzar la sessió iniciada al sistema.
Actor(s)	Usuari registrat
Component(s)	Web
Precondició	Usuari amb la sessió iniciada.
Postcondició	Usuari sense sessió iniciada.
Flux d'events	1. L'usuari tria l'opció d'eixir del sistema. 2. El sistema informa l'usuari que el procés ha sigut correcte.

CU3 :: Finalitzar sessió	
Flux alternatiu	• La sessió ja està finalitzada perquè el temps de sessió s'ha esgotat.
Extensions	No hi ha.

CU4 - Consultar dades de registre

CU4 :: Consultar dades de registre	
Resum	Consultar dades de registre de l'usuari emmagatzemades al sistema.
Actor(s)	Usuari registrat
Component(s)	Web
Precondició	Usuari amb la sessió iniciada.
Postcondició	Dades de registre mostrades a l'usuari.
Flux d'events	1. L'usuari tria l'opció de consultar les dades de registre. 2. El sistema mostra les dades desitjades.
Flux alternatiu	• No hi ha.
Extensions	• CU5 :: Modificar dades de registre • CU6 :: Donar-se de baixa

CU5 - Modificar dades de registre

CU5 :: Modificar dades de registre	
Resum	Modificar dades de registre de l'usuari emmagatzemades al sistema.
Actor(s)	Usuari registrat
Component(s)	Web
Precondició	Usuari amb la sessió iniciada.
Postcondició	Dades de registre de l'usuari modificades.
Flux d'events	1. L'usuari tria l'opció de modificar les dades de registre. 2. L'usuari modifica les dades desitjades. 3. El sistema informa l'usuari que el procés ha sigut correcte.
Flux alternatiu	• Les dades no són correctes: el sistema informa i es torna al pas 2. • L'usuari cancel·la l'operació.
Extensions	No hi ha.

CU6 - Donar-se de baixa

CU6 :: Donar-se de baixa	
Resum	S'esborra l'usuari i les seves dades del sistema.
Actor(s)	Usuari registrat

CU6 :: Donar-se de baixa	
Component(s)	Web
Precondició	Usuari amb la sessió iniciada.
Postcondició	Dades de registre i dels recorreguts de l'usuari esborrades del sistema.
Flux d'events	1. L'usuari tria l'opció de donar-se de baixa del sistema. 2. El sistema mostra un missatge de confirmació de l'operació. 3. S'esborren totes les dades relatives a l'usuari del sistema. 4. El sistema informa l'usuari que el procés ha sigut correcte.
Flux alternatiu	• L'usuari cancel·la l'operació.
Extensions	No hi ha.

CU7 - Modificar preferències

CU7 :: Modificar preferències	
Resum	Es mostren les preferències de l'aplicació per la seua modificació per part de l'usuari.
Actor(s)	Usuari
Component(s)	Dispositiu mòbil
Precondició	L'usuari ha triat un recorregut per modificar les dades.
Postcondició	Dades del recorregut modificades.
Flux d'events	1. L'usuari tria l'opció de modificar les dades d'un recorregut. 2. L'usuari modifica les dades. 3. L'aplicació mostra que la operació ha sigut correcta. 4. L'aplicació mostra les dades modificades.
Flux alternatiu	• L'aplicació s'atura per un esdeveniment extern: l'aplicació guarda el seu estat i les dades enregistrades fins eixe moment.
Extensions	No hi ha.

GD1 - Administrar dades d'usuari

GD1 :: Gestionar dades del sistema	
Resum	Gestió de les dades emmagatzemades al sistema.
Actor(s)	Administrador
Component(s)	Web
Precondició	Usuari amb la sessió iniciada i perfil d'administrador.
Postcondició	Listat de les dades demanades per l'administrador.
Flux d'events	1. L'administrador tria l'opció de gestionar les dades i el tipus de dades. 2. El sistema mostra les dades emmagatzemades al sistema.
Flux alternatiu	• No hi ha.

GD1 :: Gestionar dades del sistema	
Extensions	• GD2 :: Modificar dades del sistema • GD3 :: Esborrar dades del sistema

GD2 - Modificar dades del sistema

GD2 :: Modificar dades del sistema	
Resum	Modificació de dades emmagatzemades al sistema.
Actor(s)	Administrador
Component(s)	Web
Precondició	Usuari amb la sessió iniciada i perfil d'administrador.
Postcondició	Dades modificades.
Flux d'events	1. L'administrador tria les dades a modificar. 2. L'administrador modifica les dades. 3. Es desen les modificacions realitzades al sistema. 4. El sistema informa l'administrador que el procés ha sigut correcte.
Flux alternatiu	• Les dades introduïdes són incorrectes: el sistema informa i es torna al punt 2. • L'usuari cancel·la l'operació.
Extensions	No hi ha.

GD3 - Esborrar dades del sistema

GD3 :: Esborrar dades del sistema	
Resum	Eliminació de dades emmagatzemades al sistema.
Actor(s)	Administrador
Component(s)	Web
Precondició	Usuari amb la sessió iniciada i perfil d'administrador.
Postcondició	Dades del sistema triades esborrades.
Flux d'events	1. L'administrador tria les dades a esborrar. 2. El sistema mostra un missatge de confirmació de l'operació. 3. S'esborren totes les dades seleccionades. 4. El sistema informa l'administrador que el procés ha sigut correcte.
Flux alternatiu	• Les dades no es poden esborrar per dependències amb altres: el sistema informa i es retorna al cas d'ús GD1. • L'usuari cancel·la l'operació.
Extensions	No hi ha.

GR1 - Crear un recorregut

GU1 :: Crear un recorregut	
Resum	L'usuari enregistra un recorregut amb el GPS del dispositiu.
Actor(s)	Usuari
Component(s)	Dispositiu mòbil
Precondició	Cap.
Postcondició	Recorregut enregistrat.
Flux d'events	1. L'usuari tria l'opció d'enregistrar un recorregut. 2. L'aplicació enregistra el recorregut. 3. L'usuari tria si vol desar les dades o no. 4. L'aplicació desa les dades si el punt 3 és afirmatiu. 5. Es mostren les dades enregistrades a l'usuari.
Flux alternatiu	• No es rep senyal GPS: el sistema informa i es torna a l'inici. • L'aplicació s'atura per un esdeveniment extern: l'aplicació guarda el seu estat.
Extensions	• GR2 - Iniciar enregistrament • GR3 - Aturar enregistrament • GR4 - Desar recorregut

GR2 - Iniciar enregistrament

GR2 :: Iniciar enregistrament	
Resum	L'usuari indica el començament de l'enregistrament, el dispositiu mòbil comença la recepció i enregistrament de les dades de posició del GPS.
Actor(s)	Usuari
Component(s)	Dispositiu mòbil
Precondició	Dispositiu mòbil amb cobertura de senyal GPS.
Postcondició	Dispositiu mòbil rebent les dades de posició del GPS.
Flux d'events	1. L'usuari tria l'opció d'iniciar l'enregistrament. 2. L'aplicació comença a registrar les dades de posició del GPS.
Flux alternatiu	• El dispositiu perd la senyal GPS: el sistema informa i es passa al cas GR4 amb les dades que s'han pogut enregistrar. • L'aplicació s'atura per un esdeveniment extern: l'aplicació guarda el seu estat i les dades enregistrades fins eixe moment.
Extensions	No hi ha.

GR3 - Aturar enregistrament

GR3 :: Aturar enregistrament	
Resum	S'atura l'enregistrament de les dades de posició.
Actor(s)	Usuari

GR3 :: Aturar enregistrament	
Component(s)	Dispositiu mòbil
Precondició	Dispositiu mòbil registrant les dades de posició de GPS.
Postcondició	Dades rebudes i preparades per presentar a l'usuari.
Flux d'events	1. L'usuari tria l'opció d'aturar l'enregistrament. 2. L'aplicació atura la recepció de dades de posició. 3. L'aplicació mostra les dades enregistrades.
Flux alternatiu	• El dispositiu no rep senyal GPS: el sistema informa i es torna al cas d'ús GR1. • L'aplicació s'atura per un esdeveniment extern: l'aplicació guarda el seu estat i les dades enregistrades fins eixe moment.
Extensions	No hi ha.

GR4 – Guardar recorregut

GR4 :: Desar recorregut	
Resum	Es guarden les dades enregistrades al dispositiu.
Actor(s)	Usuari
Component(s)	Dispositiu mòbil
Precondició	Dispositiu mòbil amb dades registrades i no desades.
Postcondició	Dades desades al sistema.
Flux d'events	1. L'usuari tria l'opció de desar el recorregut. 2. L'aplicació desa les dades enregistrades. 3. L'aplicació informa que l'operació ha sigut correcta. 4. L'aplicació mostra les dades enregistrades.
Flux alternatiu	• El dispositiu no rep senyal GPS: el sistema informa i es torna al cas d'ús GR1. • L'aplicació s'atura per un esdeveniment extern: l'aplicació guarda el seu estat i les dades enregistrades fins eixe moment.
Extensions	No hi ha.

GR5 – Consultar recorreguts

GR5 :: Consultar recorreguts		
Resum	L'aplicació mostra un llistat dels recorreguts desats al sistema.	
Actor(s)	Usuari	Usuari registrat
Component(s)	Dispositiu mòbil	Web
Precondició	Cap.	
Postcondició	Llistat amb els recorreguts desats.	
Flux d'events	1. L'usuari tria l'opció de consultar els recorreguts. 2. L'aplicació mostra els recorreguts desats al sistema.	

GR5 :: Consultar recorreguts	
Flux alternatiu	• Dispositiu mòbil - L'aplicació s'atura per un esdeveniment extern: l'aplicació guarda el seu estat i les dades enregistrades fins eixe moment.
Extensions	• GR6 :: Consultar detall de recorregut • GR7 :: Modificar dades de recorregut • GR8 :: Esborrar recorregut

GR6 – Consultar detall de recorregut

GR6 :: Consultar detall de recorregut		
Resum	L'aplicació mostra el detall d'un recorregut seleccionat per l'usuari.	
Actor(s)	Usuari	Usuari registrat
Component(s)	Dispositiu mòbil	Web
Precondició	L'usuari ha triat un recorregut a consultar.	
Postcondició	Dades del recorregut mostrades a l'usuari.	
Flux d'events	1. L'usuari tria l'opció de consultar el detall d'un recorregut. 2. L'aplicació mostra les dades associades al recorregut.	
Flux alternatiu	• Dispositiu mòbil - L'aplicació s'atura per un esdeveniment extern: l'aplicació guarda el seu estat i les dades enregistrades fins eixe moment.	
Extensions	GR7 :: Comparar recorreguts	

GR7 – Comparar recorreguts

GR7 :: Comparar recorreguts		
Resum	L'aplicació mostra els detalls de dos recorreguts i els compara.	
Actor(s)	Usuari	Usuari registrat
Component(s)	Dispositiu mòbil	Web
Precondició	L'usuari ha triat un recorregut a consultar.	
Postcondició	Dades dels recorreguts comparades i mostrades a l'usuari.	
Flux d'events	1. L'usuari tria l'opció de comparar un altre recorregut al que està consultant. 2. L'aplicació mostra un llistat amb els recorreguts similars. 3. L'usuari tria un recorregut a comparar. 4. L'aplicació mostra les dades comparades.	
Flux alternatiu	• L'usuari cancel·la l'operació: es torna al punt 1. • Dispositiu mòbil - L'aplicació s'atura per un esdeveniment extern: l'aplicació guarda el seu estat i les dades enregistrades fins eixe moment.	
Extensions	No hi ha.	

GR8 - Modificar dades de recorregut

GR8 :: Modificar dades de recorregut		
Resum	L'aplicació permet la modificació de les dades d'un recorregut	
Actor(s)	Usuari	Usuari registrat
Component(s)	Dispositiu mòbil	Web
Precondició	L'usuari ha triat un recorregut per modificar les dades.	
Postcondició	Dades del recorregut modificades.	
Flux d'events	1. L'usuari tria l'opció de modificar les dades d'un recorregut. 2. L'usuari modifica les dades. 3. L'aplicació mostra que la operació ha sigut correcta. 4. L'aplicació mostra les dades modificades.	
Flux alternatiu	• Les dades introduïdes són incorrectes: l'aplicació informa i es torna al punt 2. • L'usuari cancel·la l'operació. • Dispositiu mòbil - L'aplicació s'atura per un esdeveniment extern: l'aplicació guarda el seu estat i les dades enregistrades fins eixe moment.	
Extensions	No hi ha.	

GR9 - Esborrar recorregut

GR9 :: Esborrar recorregut		
Resum	Esborrar un recorregut de l'usuari seleccionat.	
Actor(s)	Usuari	Usuari registrat
Component(s)	Dispositiu mòbil	Web
Precondició	L'usuari ha triat un recorregut per esborrar.	
Postcondició	Dades del recorregut esborrades del sistema.	
Flux d'events	1. L'usuari tria l'opció d'esborrar les dades d'un recorregut. 2. Es mostra un missatge de confirmació de l'operació. 3. L'aplicació esborra les dades. 4. L'aplicació mostra que la operació ha sigut correcta.	
Flux alternatiu	• Ocorre un error esborrant les dades: l'aplicació informa i es torna a l'inici del cas d'ús. • L'usuari cancel·la l'operació. • Dispositiu mòbil - L'aplicació s'atura per un esdeveniment extern: l'aplicació guarda el seu estat i les dades enregistrades fins eixe moment.	
Extensions	No hi ha.	

GR10 :: Sincronitzar recorreguts	
Resum	Sincronització dels recorreguts amb l'aplicació web per part de l'usuari.
Actor(s)	Usuari
Component(s)	Dispositiu mòbil
Precondició	• L'usuari està registrat al sistema. • Les dades d'accés (usuari i contrasenya) estan configurades al dispositiu mòbil.
Postcondició	Dades sincronitzades.
Flux d'events	1. L'usuari tria l'opció de sincronitzar les dades dels recorreguts. 2. L'usuari modifica les dades. 3. L'aplicació mòbil envia les dades a l'aplicació web. 4. L'aplicació informa que el procés ha sigut correcte.
Flux alternatiu	• No hi ha connexió a la xarxa: s'informa l'usuari que l'operació no es pot realitzar i es torna al punt 1. • Error a la sincronització: s'informa l'usuari i es torna al punt 1. • Conflictes de sincronització: es mostren a l'usuari per a que trie les dades correctes. • L'aplicació s'atura per un esdeveniment extern: l'aplicació guarda el seu estat i les dades enregistrades fins eixe moment.
Extensions	No hi ha.

2.3 Diagrames de seqüència

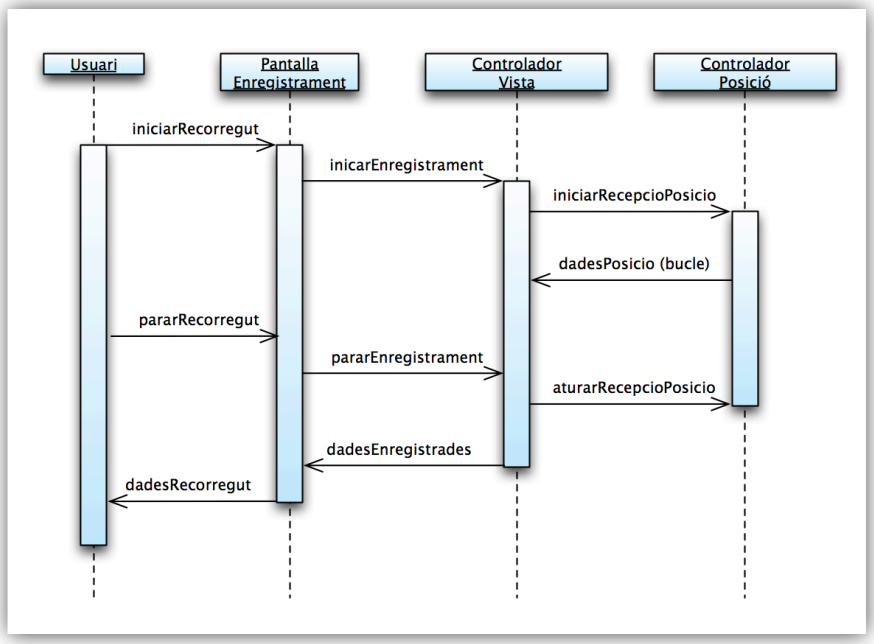
A la descripció dels casos d'ús realitzada al punt anterior s'ha detallat la seqüència de passes que es realitzen per a cada acció.

Així i tot es presenten els diagrames de seqüència de dues accions molt representatives del sistema, com són enregistrar els recorreguts i la sincronització entre el dispositiu mòbil i el servei web.

2.3.1 Registre d'un recorregut

Hi ha dues accions de l'usuari que provoquen resposta al sistema:

- **Inici de recorregut:** el sistema inicia l'enregistrament de la posició.
- **Parada de recorregut:** el sistema para l'enregistrament de la posició i mostra el recorregut resultant.



2.3.2 Sincronització

La tasca de sincronització es detalla donat que és l'operació amb més casuística del sistema. Es detalla les passes que es realitzen per sincronitzar les dades entre ambdós components:

1. El dispositiu mòbil revisa els recorreguts pendents de sincronitzar.

   1.1. Si hi ha conflictes pendent de resolució s'informa a l'usuari mostrant les opcions per a que trie les dades que vol conservar.

2. Prepara les dades per ser transmeses més les credencials necessàries per identificar-se al servidor.
3. Les envia a través de la xarxa.
4. El servidor rep les dades.
5. El servidor comprova les credencials:

   5.1. Si no són correctes retorna el resultat al dispositiu amb error, s'informa a l'usuari i s'atura el procés.

6. Processa els recorreguts rebuts:

   6.1. Si el recorregut no existeix crea un nou a la base de dades.

   6.2. Si existeix es comprova el seu estat:

      6.2.1. Defecte: s'actualitza amb les dades rebudes.

      6.2.2. Pendent de sincronitzar: es prepara per enviar al dispositiu per donar a triar a l'usuari quines dades vol que es desen.

      6.2.3. Pendent de resolució: havia un conflicte i el servidor estava a l'espera de la decisió de l'usuari:

         6.2.3.1. Si la resolució és Master-M el màster és el dispositiu mòbil, per tant les dades del dispositiu actualitzen les del servidor.

6.2.3.2. Si la resolució és Master-S: el màster és el servidor, per tant el dispositiu haurà actualitzat les dades i el servidor canvia les dades a l'estat defecte.

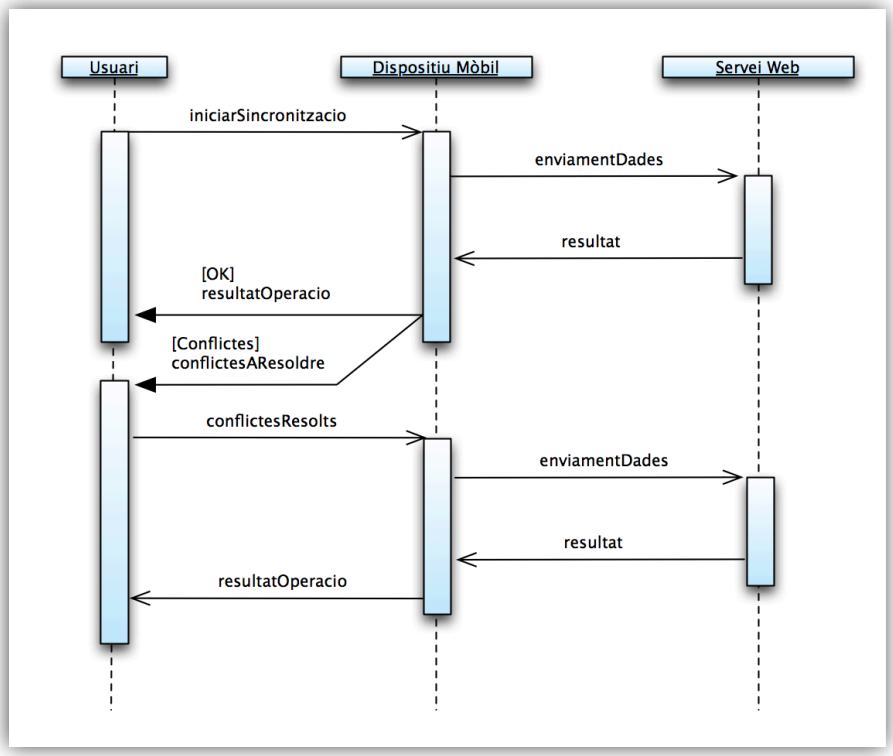
6.3. Si s'ha d'esborrar el servidor elimina les dades.

7. El servidor envia el resultat de l'operació amb les dades pendent de resolució.
8. El dispositiu mostra a l'usuari els conflictes per a que trie:

8.1. Si l'usuari resol els conflictes s'inicia el procés de sincronització.

8.2. Si l'usuari posposa la resolució s'emmagatzema i a la propera sincronització es tornarà a consultar.

A continuació es presenta una versió simplificada de les passes seguides a la sincronització:



3. Disseny

3.1 Decisions tecnològiques

El sistema està compost per dos components diferenciats: el dispositiu mòbil i el servidor web.

A banda d'aquests dos components també es descriu com s'implementarà la seva sincronització.

3.1.1 Dispositiu mòbil

El dispositiu per al què es desenvolupa l'aplicació és per l'iPhone i aquest funciona amb el sistema operatiu iOS. Aquest fet limita les tecnologies a emprar.

- **Objective-C:** el llenguatge de programació emprat per a iOS, és una especialització del llenguatge C.
- **Cocoa Touch:** és el conjunt de classes que proporciona l'entorn de desenvolupament per facilitar accions amb el dispositiu, presentació d'elements d'interfície, tasques, etc... Proporciona bastidors com el Core Data, UIKit, etc.

3.1.2 Servidor web

El conjunt d'alternatives de tecnologies a emprar per desenvolupar el component web del sistema és molt més ampla que al cas anterior.

El servei web es desenvoluparà al Google App Engine, que proveïx un entorn altament escalable de forma transparent, així com avantatges econòmics donat que fins un límit de consum de recursos els servei és gratuït.

Google App Engine permet el desplegament d'aplicacions web tant amb el llenguatge Java com Python. Per aquest component s'emprarà:

- **Python:** llenguatge que suporta tant l'orientació a objectes com imperatiu. Està concebut com un llenguatge que afavorisca la sintaxi neta i llegible. Aquest llenguatge és multiplataforma i permet el desenvolupament tant de aplicacions d'escriptori com processar CGI per servidors web.
- **Django:** és un marc de treball desenvolupat amb Python que permet el desenvolupament d'aplicacions web de forma ràpida.

3.1.3 Sincronització

Per realitzar la sincronització entre ambdós dispositius s'emprarà l'arquitectura REST codificant les dades a transmetre mitjançant SOAP. Aquest objecte encapsularà informació de la sincronització i les dades enviades estructurades de la següent forma:

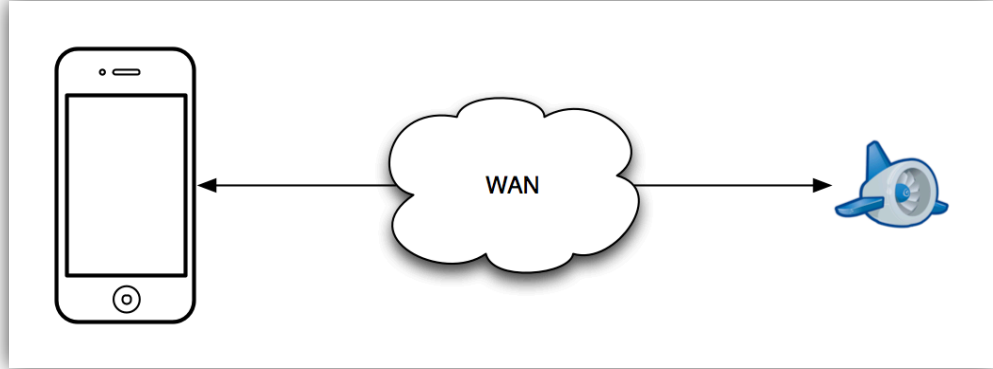
- **Capçalera:** amb informació sobre la sincronització.
- **Credencials:** informació sobre la identificació de l'usuari.
- **Resultat de les operacions:** informació sobre les operacions realitzades i si han sigut correctes.
- **Operacions a realitzar:** informació amb els següents passos a realitzar.
- **Dades a processar:** informació de les dades a sincronitzar entre el dispositiu i el servei web.

3.2 Arquitectura del sistema

El sistema proposat consisteix d'una aplicació mòbil que es sincronitza mitjançant WAN a un servidor, així les dades es mantindran a ambdós components en un estat consistent.

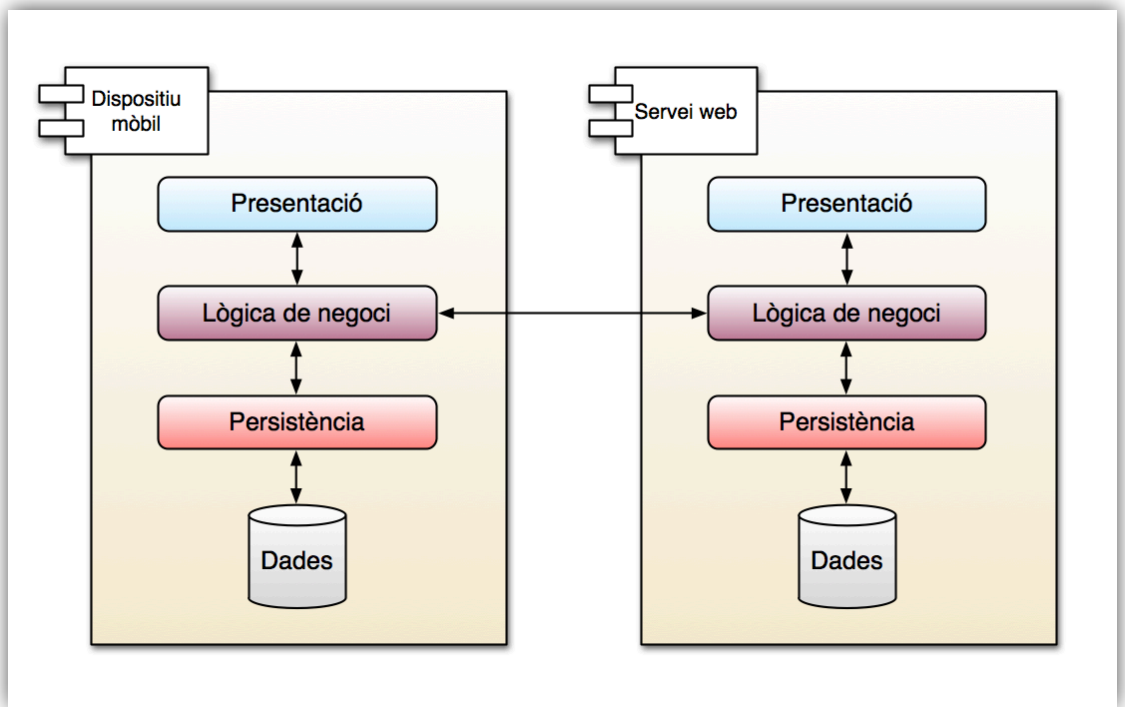
3.2.1 Vista física

L'arquitectura del sistema física del sistema correspon als dos components que es connectaran entre sí a través de la xarxa (WAN):



3.2.2 Vista lògica

La vista lògica del sistema es divideix de forma clara entre els dos components. Així i tot s'aplicarà la divisió en capes per encapsular les funcionalitats en grups funcionals de presentació, lògica de negoci i maneig de la persistència. Aquesta organització correspon al patró de disseny Model Vista Controlador que es detalla més endavant.



3.3 Patrons de disseny

3.3.1 Model Vista Controlador (MVC)

És el principal patró de disseny aplicat pel desenvolupament del sistema. Tant l'entorn de desenvolupament d'iOS com el marc de treball Django implementen aquest patró per obtindre codi més estructurat, encapsulat i reutilitzable.

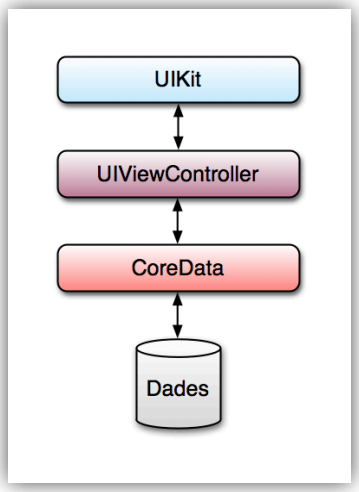
El patró separa en diferents capes que encapsulen les funcionalitats lògiques de l'aplicació:

- **Model:** encapsula les funcionalitats d'interacció amb el sistema de persistència de les dades.
- **Vista:** aporta les funcionalitats per presentar les dades i interactuar amb l'usuari.
- **Controlador:** conté la lògica de negoci de l'aplicació i fa de nexe entre la capa de Vista i el Model.

Aquest patró té característiques particulars segons el component.

Cocoa Touch (iOS)

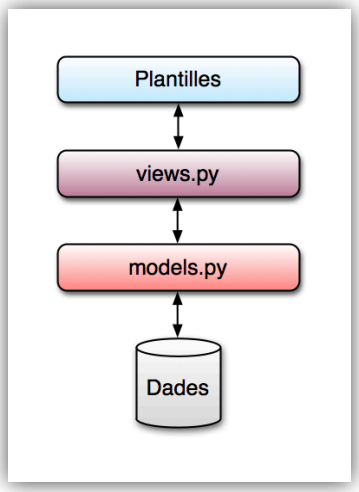
Cocoa Touch aporta una sèrie de marcs de treball que faciliten la crida d'operacions comuns del dispositiu segons la seva finalitat: interacció amb l'usuari, interacció amb elements multimèdia, ús de la localització, ús del sistema de persistència, etc.



Dels diferents marcs de treball les classes i conjunts de classes que aporten les funcionalitats de cada capa del sistema són:

- **Core Data:** aporta les classes que permeten la interacció amb la persistència de les dades.
- **UIKit:** té les classes que encapsulen la presentació dels elements de la interfície i les seves funcionalitats.
- **UIViewController:** aquesta classe incorpora la lògica de negoci de l'aplicació.

Django



Aquest marc de treball també aporta diferents mòduls per facilitar tasques comunes i poder reutilitzar el codi. Els elements del marc que aporten les funcionalitats són:

- **models.py:** a aquest arxiu es defineixen les classes que representen les dades i aporten les funcionalitats d'interacció amb el sistema de gestió de les dades.
- **Plantilles:** Django incorpora un sistema de plantilles que rep les dades i genera l'arxiu HTML per presentar a l'usuari.
- **views.py:** inclou les funcions que aporten la lògica de negoci a les peticions rebudes.

Aquesta nomenclatura pot prestar a confusió donat que la capa de controlador estiga continguda a l'arxiu *views* (vistes). De fet,

el projecte Django reanomena el patró com MVT (Model-View-Template) on la vista genera les dades a presentar i la capa *template* presenta les dades.

3.3.2 Delegat

El patró Delegat (*Delegate)* consisteix en un mecanisme on un objecte nombra un delegat d'una tasca. D'aquesta forma el primer objecte envia un missatge (crida una funció) del delegat quan necessita alguna acció o dada.

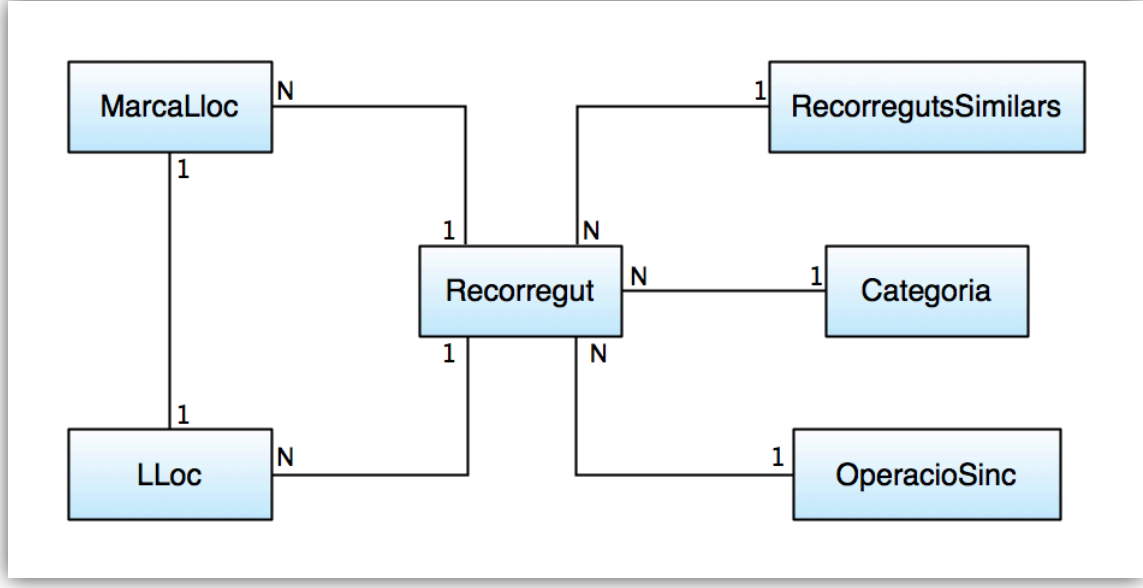
Per aplicar aquest patró normalment una classe ha d'implementar un protocol consistent a una sèrie de funcions necessàries que un objecte que el nombre delegat pot necessitar.

3.4 Persistència

Les dades del sistema, com s'ha exposat al punt de l'arquitectura, estaran emmagatzemades als dos gestors de dades dels components. Per tant, la major part del disseny de persistència coincideix a ambdós sistemes donat que emmagatzemen la mateixa informació.

3.4.1 General

A continuació es presenta un diagrama amb les entitats emprades pel sistema comunes a ambdós components:



Els camps de les entitats seran:

- **Recorregut:** id, nom, data, categoria, nota, ppm (pulsacions), distanciaTotal, tempsTotal, altitudMaxima, altitudMinima, ritme, velocitatMaxima, dadesKML.
- **MarcaLloc:** id, nom, tipus, tempsParcial, distanciaParcial, altitudMaxima, altitudMinima, velocitatMaxima, ritmeParcial, recorregut_id, lloc_id.
- **Lloc:** id, coordenadaX, coordenadaY, altitud, segellTemps.
- **Categoria:** id, nomCategoria.

- **RecorregutsSimilars:** id, recorregutsSimilars.
- **OperacionsSinc:** id, recorregut_id, accio, estat, dataActualitzacio.

Donat les característiques dels diferents sistemes de gestió de la base de dades hi ha un punt divergent referent al tractament de la codificació del recorregut en format KML. Al diagrama es pot observar el camp *kmlFile* que presentarà la següent particularitat segons el component on es tracte:

- **Dispositiu mòbil:** el camp emmagatzema una referència al sistema d'arxius on les dades estan emmagatzemades.
- **Servei web:** el camp serà de tipus *TextProperty*[17] i emmagatzemarà les dades en format KML del recorregut.

Per gestionar els recorreguts al servei web s'afegeix un camp que referencia l'usuari al què pertany el recorregut.

En quant a les dades relatives als usuaris el seu tractament dependrà del component.

3.4.2 Dispositiu mòbil

Al dispositiu mòbil s'emmagatzemaran les dades de registre de l'usuari (si està registrat) i també les preferències de les opcions que tindrà l'aplicació.

Per aquesta tasca s'emprarà la classe NSUserDefaults que permet gestionar els valors de preferències. Realment consisteix en un diccionari que permet emmagatzemar parells clau-valor que emmagatzema al sistema d'arxius a un arxiu tipus PLIST[18], a més d'implementar els mètodes necessaris per la seua gestió.

Preferències		
nomUsuari	Cadena	Nom d'usuari del sistema
Nom	Cadena	Nom de l'usuari
contrasenya	Cadena	Contrasenya de l'usuari
categories	Dictionary	Conjunt de categories per classificar els recorreguts
registrarMarques	booleà	Defineix si l'usuari vol enregistrar marques intermèdies del recorregut
distanciaMarques	Número	Distància entre marques
exactitud	Número	Exactitud de la senyal GPS, cada quants metres es registra una nova posició.

3.4.3 Servei web

Al servei web s'emmagatzemaran les dades de registre de l'usuari.

[17] Tipus de dades que permet emmagatzemar texts de més de 500 KB (http://code.google.com/intl/es/appengine/docs/python/datastore/typesandpropertyclasses.html#TextProperty).

[18] Els arxius PLIST són arxius XML que emmagatzemen valors de forma jeràrquica de tipus bàsics. Normalment s'empren per emmagatzemar preferències, diccionaris, etc. (http://developer.apple.com/library/mac/#documentation/Cocoa/Conceptual/PropertyLists/Introduction/Introduction.html).

El marc de treball Django proporciona un mòdul per facilitar les tasques de registre i identificació dels usuaris mitjançant nom d'usuari i contrasenya. Per realitzar aquesta funció defineix una classe User[19] que s'estima és suficient per emmagatzemar les dades de l'usuari.

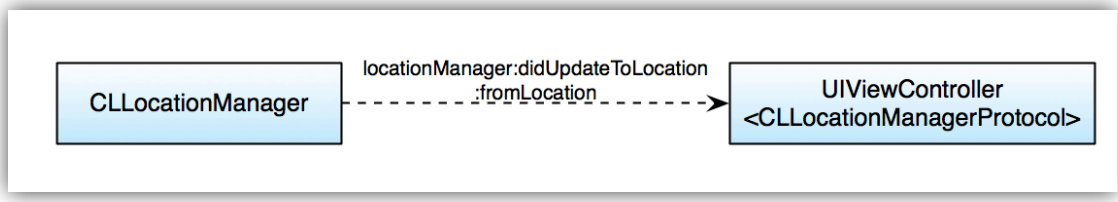
User		
nickname	Cadena	Nom d'usuari del sistema
first_name	Cadena	Nom de l'usuari
last_name	Cadena	Cognom de l'usuari
email	Cadena	Correu electrònic.
password	Cadena	Contrasenya de l'usuari
is_staff	Booleà	Indica si l'usuari forma part de l'equip de gestió del sistema. Pot accedir a determinades funcions d'administrador.
is_active	Booleà	Indica si l'usuari està actiu. Si no ho està no pot accedir al sistema.
is_superuser	Booleà	Indica si l'usuari té permisos de superusuari.
last_login	Data	Data i hora de la última connexió de l'usuari.
date_joined	Data	Data i hora quan l'usuari va donar-se d'alta.

3.5 Servei de localització

Una funcionalitat bàsica de l'aplicació és la localització de l'usuari per poder enregistrar el seu recorregut. Per realitzar-ho el marc de treball Cocoa Touch aporta un conjunt de classes anomenat Core Location[20].

El funcionament d'aquest marc de treball de forma resumida consisteix:

1. Crear una instància del controlador de localització amb les opcions d'exactitud i tipus de localització, etc. i el seu delegat a qui informarà.
2. S'indica que comence a enregistrar la posició.
3. El controlador de localització envia un missatge al seu delegat quan actualitza la posició amb les dades de la nova posició i l'antiga.



Les classes que s'empraran per a l'ús del sistema de posicionament són:

[19] Mòdul contribuït Auth (http://docs.djangoproject.com/en/dev/topics/auth/#api-reference).

[20] Referència Core Location (http://developer.apple.com/library/ios/#documentation/CoreLocation/Reference/CoreLocation_Framework/_index.html)

- **CLLocationManager:** la classe que controla l'ús del posicionament del dispositiu i envia la informació al delegat.
- **CLLocation:** aquesta classe encapsula la informació d'una determinada informació.
- **UIViewController:** serà la classe delegada que rebrà i tractarà la informació. Per realitzar aquest rol ha d'implementar el protocol CLLocationManagerDelegate, consistent en implementar les funcions de tractament de la informació enviada per CLLocationManager.

3.6 Interfície

A continuació es mostren uns esborranys que representen una primera aproximació de la interfície d'ambdós components.

3.6.1 Dispositiu mòbil

La interfície del dispositiu mòbil està concebuda per navegar entre les diferents vistes mitjançant una barra de pestanyes situada a la part inferior de la pantalla:

- Vista d'enregistrament: que permetrà iniciar i parar l'enregistrament dels recorreguts.
- Vista de recorreguts: mostra un llistat amb els recorreguts enregistrats i permet accedir al seu detall. Per controlar la navegació s'empra un Controlador de navegació situat a la part superior de la pantalla.
  - La vista de detall permet triar entre la visualització de les estadístiques o del mapa del recorregut.
- Vista de sincronització: mostra les opcions relatives a la sincronització de les dades.
- Vista de preferències: mostra les opcions de l'aplicació que pot personalitzar l'usuari.

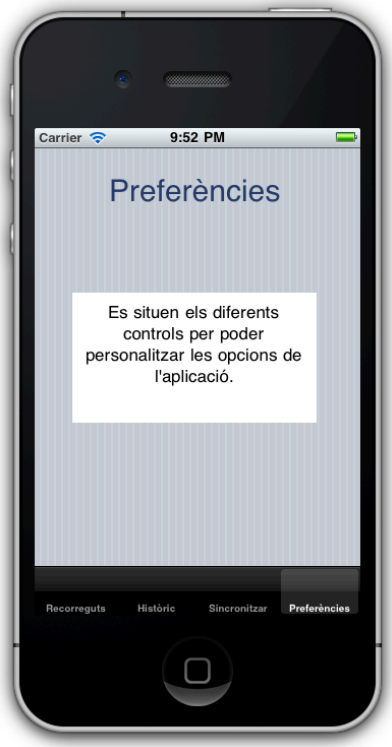
Captures de pantalla

Vista d'enregistrament

Vista de recorreguts

Vista de detall de recorregut

Vista de sincronització



Vista de preferències

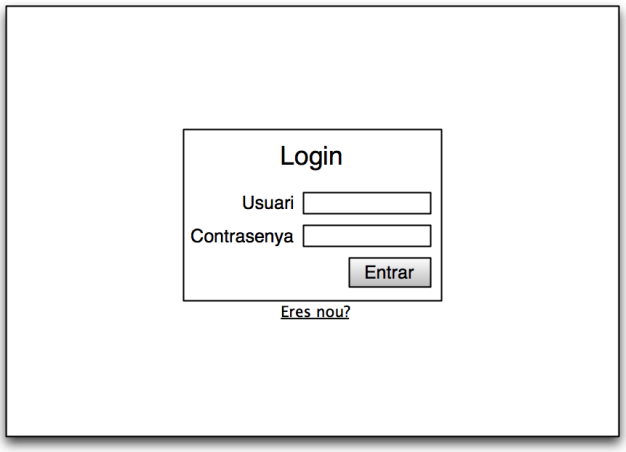
3.6.2 Servei web

La interfície del servei web constarà de dos vistes generals:

- Registre: on es mostrarà el formulari corresponent al registre de nous usuaris o identificació al sistema.
- Gestió: on es mostren els recorreguts i el seu detall.

A continuació es presenten els esborranys de les interfícies

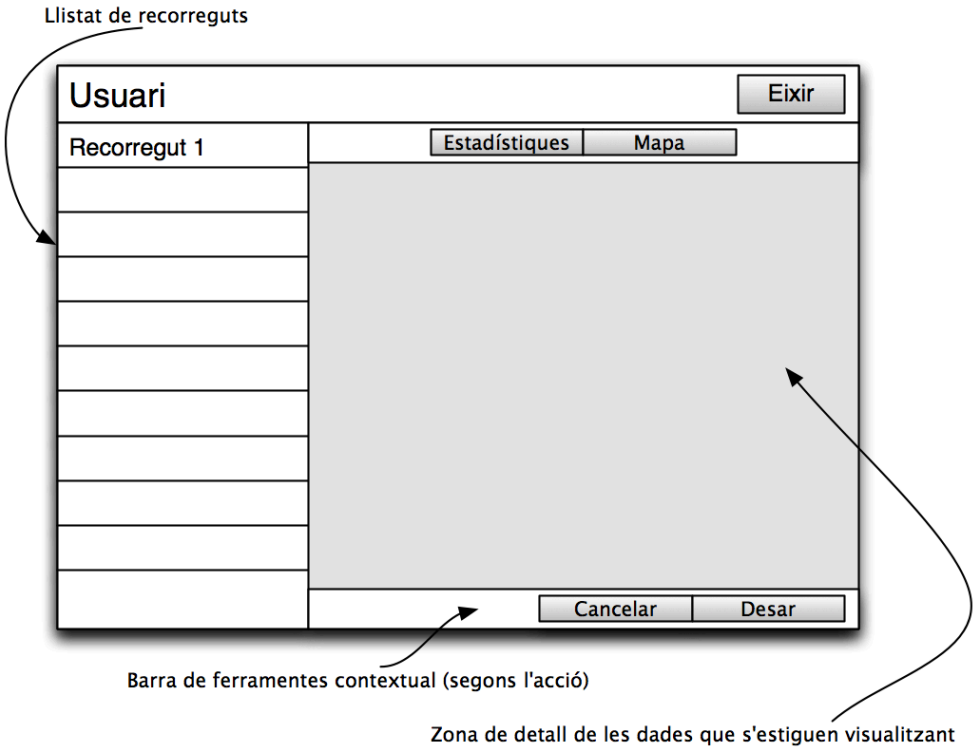
Registre



Gestió

La pantalla de gestió té vàries zones diferenciades:

- La part superior on es visualitza el nom d'usuari i l'opció d'eixir del sistema.
- La barra lateral esquerre on es situarà el llistat de recorreguts.
- La part central on es mostrarà la informació desitjada:
  - Estadístiques del recorregut seleccionat.
  - Vista de mapa del recorregut.
  - Vista de modificació de preferències.



4. Implementació

4.1 Dispositiu mòbil

4.1.1 Conceptes previs

El desenvolupament de l'aplicació mòbil amb l'entorn de programació Cocoa Touch ha precisat del coneixement d'alguns conceptes que s'han aplicat a diferents llocs.

A continuació es realitza una breu descripció d'aquests.

Classes Singleton

Són classes de les què únicament hi ha una instància. D'aquesta forma les seues propietats i mètodes són accessibles des de tota l'aplicació.

L'exemple principal és la classe UIApplication que permet l'accés a la instància en execució de l'aplicació. A l'aplicació implementada aquesta classe conté instàncies del context d'objectes gestionats i el gestor de permanència, així des de qualsevol lloc de l'aplicació es poden recuperar i emprar.

Aquests tipus de classe s'ha emprat per a la gestió de tasques que necessiten ser accedides a tota l'aplicació:

- Gestió de la recepció de les dades de localització.
- Gestió del registre de tasques a sincronitzar.

Protocols

Els protocols permeten la comunicació entre ambdues classes quan una realitza una acció o executa un mètode.

Una classe A que defineix un protocol declara quins mètodes (obligatoris o opcionals) ha d'implementar una classe B que vol emprar el protocol declarat. A banda dels mètodes també emmagatzema una referència a la instància delegada (de B) que emprarà per cridar als mètodes implementats per B.

Un exemple és per la gestió de les vistes de taules. Per poder mostrar les dades i gestionar les accions sobre una taula, la classe que la gestiona ha d'implementar els mètodes definits als protocols UITableViewDelegate i UITableViewDataSource. Aquests protocols permeten:

- UITableViewDelegate: s'ha d'implementar els mètodes que responen a la interacció dels usuaris.
- UITableViewDataSource: s'ha d'implementar els mètodes que retornen dades com el número de seccions d'una taula, el número de cel·les per secció, les dades corresponents a les cel·les, etc.

Els protocols s'han emprat a l'aplicació sobre tot en dos aspectes:

- Per retornar informació introduïda o modificada a una vista a la vista que l'invocava.
- Per comunicar la finalització d'accions o mètodes d'una instància a una altra.

Exemple de definició del protocol SyncManagerProtocol:

```
@protocol SyncManagerProtocol <NSObject>
```

```
// Mètodes que s'han d'implementar obligatòriament al protocol
// No hi ha

@optional
// Mètodes opcionals que es poden implementar
- (void)syncManagerDidEndSyncWithConflitcs:(BOOL)thereAreConflitcs;

@end

// A la capçalera de la classe s'emmagatzema una instància del delegat
@interface SyncManager : NSObject {
   id<SyncManagerProtocol> delegate;
   ...
}
```

Core Data

Core Data[21] es el *framework* que permet la gestió de la permanència de les dades, tant amb una base de dades com l'emmagatzemament a arxius.

Per la gestió de la base de dades hi ha tres classes fonamentals:

- **NSPersistentStoreCoordinator:** aquesta classe s'encarrega de les tasques de comunicació entre els objectes que representen les dades i el sistema de base de dades i d'arxius.
- **NSManagedObjectContext:** el context d'objectes gestionats consisteix en una instància que observa els canvis realitzats al conjunt d'objectes gestionats. Una vegada es volen emmagatzemar els canvis es crida a aquesta instància que realitza les tasques corresponents. També per realitzar consultes s'envien peticions al context.
- **NSManagedObject:** és la classe que representa un tipus de dades a la base de dades. D'aquesta forma la gestió a l'aplicació és amb les instàncies d'aquesta classe evitant la manipulació directa a la base de dades.

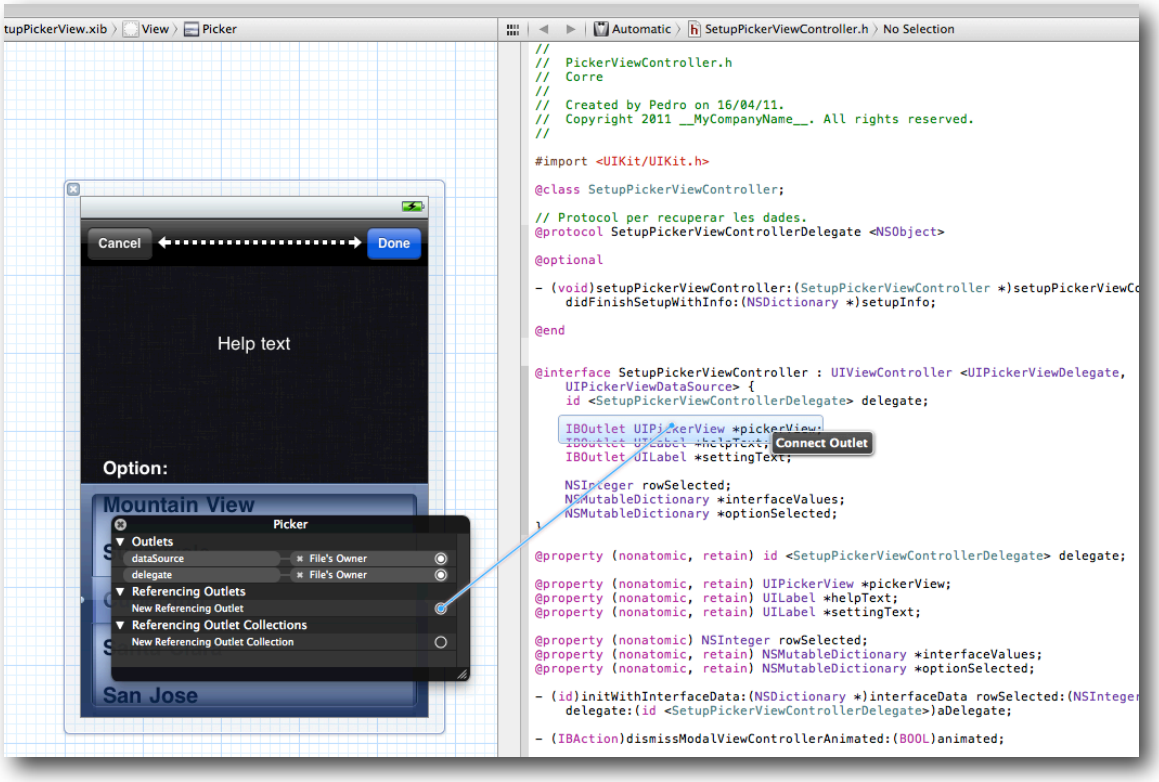
IBAction i IBOutlet

La creació d'interfícies i vistes pot ser mitjançant codi directament, personalitzant els constructors de la classe i afegint els elements o bé amb la ferramenta Interface Builder. Aquesta ferramenta facilita la inclusió dels elements a les diferents vistes i la connexió entre aquests elements i el codi.

Les interfícies generades es representen mitjançant un arxiu amb extensió XIB que té un format XML on especifica tots els elements inclosos a la interfície.

Els elements de la interfície poden passar o rebre informació, com per exemple un camp de text o respondre a una interacció de l'usuari, provocant un esdeveniment.

[21] Core Data Programming Guide (http://developer.apple.com/library/mac/#documentation/Cocoa/Conceptual/CoreData/cdProgrammingGuide.html....)



Per indicar al codi quines classes poden ser connectades a elements definits a un arxiu XIB s'empren els modificadors IBAction i IBOutlet:

- IBAction: es declara com a valor que retorna un mètode. Realment no retorna cap valor i és equivalent a *void*, però serveix per a poder connectar esdeveniments de la interfície amb el mètode.
- IBOutlet: es declara junt a la classe i, com al cas anterior, indica a la ferramenta Interface Builder que es pot assignar un element de la interfície a aquesta declaració de classe.

4.1.2 Model

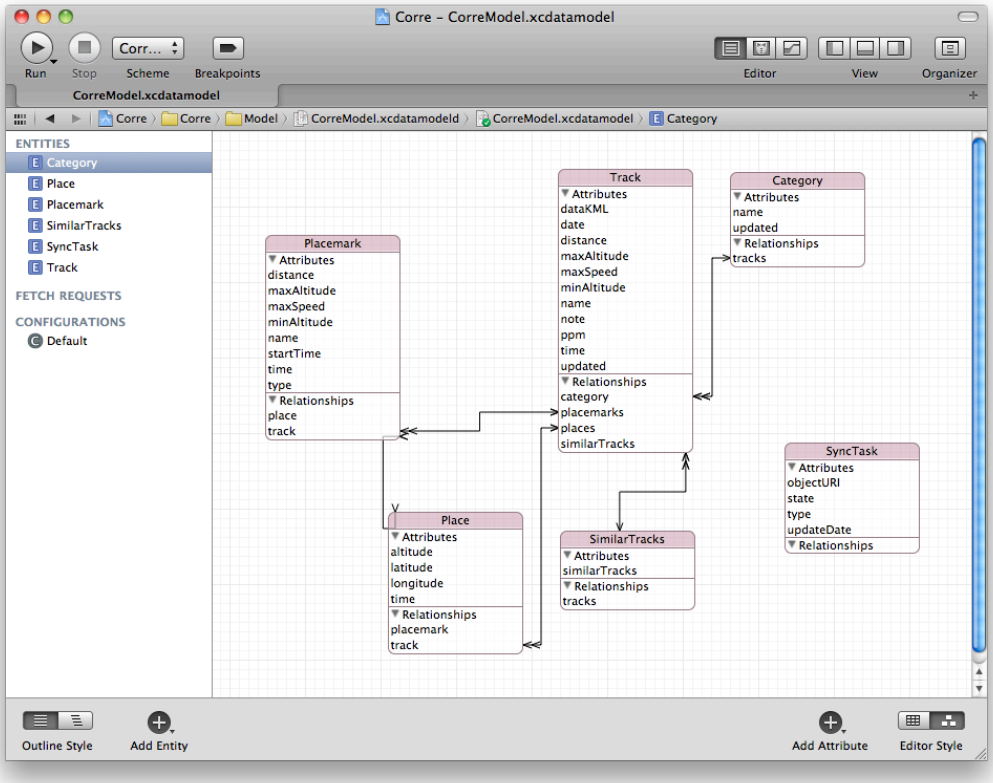
Per a la gestió de les dades generades dels recorreguts s'ha emprat:

- **Base de dades:** les dades dels recorreguts, categories, punts de localització, etc. s'han emmagatzemat a una base de dades, emprant les ferramentes de Core Data.
- **Sistema d'arxius:** quan es genera un recorregut es crea un arxiu KML associat a ell amb el què, posteriorment a la pàgina web, es mostraran les dades al mapa de Google Maps[22].
- **Preferències d'usuari:** el sistema aporta una classe del tipus diccionari que permet l'emmagatzemament de valors numèrics, text i booleans. Aquesta classe té el fi d'emmagatzemar preferències d'usuari o valors que no requereixen una gestió més complexa (com una base de dades).

[22] Aquest arxiu es pot obrir amb el programa Google Earth.

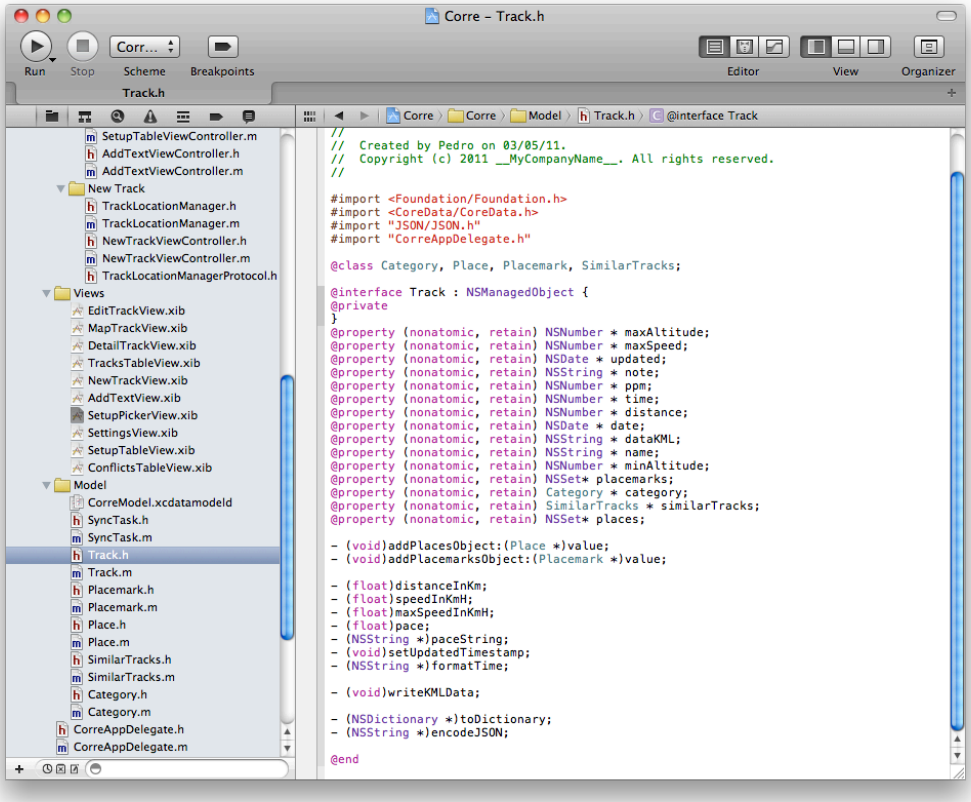
Base de dades

El model de dades es defineix amb l'entorn de desenvolupament que genera una arxiu XML amb el seu disseny (CorreModel.xcdatamodel).



A partir d'aquesta definició l'entorn permet generar les classes de cada entitat. Aquestes classes[23] són subclasses de NSManagedObject, cosa que ens permet gestionar les dades amb les instàncies que posteriorment la instància de l'objecte NSManagedObjectContext s'encarregarà de gestionar.

[23] Les classes estan al grup Model de l'entorn de desenvolupament.



Les entitats definides emmagatzemen una sèrie de valors, però hi ha d'altres que es poden deduir des d'aquests. Per aquest motiu hi ha classes del model als què s'han afegit mètodes que retornen aquests valors deduïts.

Com a exemple aprofundim amb l'entitat Track.

Aquesta entitat té definides com a propietats els diferents valors que s'emmagatzemen, entre ells la distància i el temps emprat per realitzar el recorregut:

```
@property (nonatomic, retain) NSNumber * time;
@property (nonatomic, retain) NSNumber * distance;
```

A partir d'aquestes propietats es pot calcular la velocitat o també, calcular la distància però en kilòmetres i no en metres com s'emmagatzemen. Així s'han definit els següents mètodes que calculen els valors derivats[24]:

```
- (float)distanceInKm;          // Distància en kilòmetres
- (float)speedInKmH;            // Velocitat en kilòmetres per hora
- (float)maxSpeedInKmH;         // Velocitat màxima en kilòmetres hora (està en m/s)
- (float)pace;                  // Ritme, expressat en minuts per kilòmetre
- (NSString *)paceString;       // Retorna el ritme però en format mm:ss
- (NSString *)formatTime;       // Retorna el temps emprat però en format hh:mm:ss
```

A banda dels mètodes per calcular valors derivats també s'han definit mètodes auxiliars per realitzar tasques relatives a la serialització de les dades (s'implementa a totes les classes del model on es necessita enviar la informació per sincronitzar) i per generar l'arxiu KML amb el recorregut:

[24] Els comentaris estan afegits a aquest document.

```
- (void)writeKMLData;              // Genera l'arxiu KML

- (NSDictionary *)toDictionary;    // Crea un diccionari amb els valors de la classe
- (NSString *)encodeJSON;          // Retorna una cadena amb format JSON del diccionari
```

Sistema d'arxius

El sistema d'arxius s'empra per emmagatzemar els arxius KML dels recorreguts generats per la classe Track.

Per realitzar aquesta tasca es fa ús de la classe *singleton* NSFileManager, recuperant la seva instància per poder generar un nou arxiu al sistema d'arxius:

```
// Instància de la classe NSFileManager
NSFileManager *fileManager = [NSFileManager defaultManager];
// Ruta on es creen els arxius KML.
NSURL *KMLDirectory = [NSURL fileURLWithPath:[[NSSearchPathForDirectoriesInDomains
(NSDocumentDirectory, NSUserDomainMask, YES) lastObject]
stringByAppendingPathComponent:@"KMLFiles"]];
...

// Creació de l'arxiu amb el nom emmagatzemat a la propietat dataKML
[fileManager createFileAtPath:[[KMLDirectory path] stringByAppendingPathComponent:[self
dataKML]] contents:nil attributes:nil];
```

Es genera una cadena de caràcters amb el codi XML que s'escriu a l'arxiu generat:

```
[fileContent writeToFile:[[KMLDirectory path] stringByAppendingPathComponent:[self
dataKML]] atomically:YES encoding:NSUTF8StringEncoding error:&error]
```

Per recuperar la cadena la classe NSString facilita una funció que inicialitza la instància amb els continguts d'un arxiu de text. Aquest mètode s'empra quan es genera el diccionari amb les dades d'un recorregut:

```
NSString *fileContents = [[NSString alloc] initWithContentsOfFile:path
encoding:NSUTF8StringEncoding error:&error];
```

Preferències d'usuari

La classe *singleton* que gestiona aquest tipus de valors és NSUserDefaults amb la seua instància standardUserDefaults. Aquesta classe emmascara un diccionari on es poden afegir valors associats a una clau.

A la primera execució de l'aplicació es comprova si hi ha preferències definides, si no es defineixen uns primers valors (així com quatre categories predefinides).

Les preferències corresponen als valors que empra el gestor de localització per definir els seus paràmetres:

```
[[NSUserDefaults standardUserDefaults] setBool:YES forKey:kPlacemarksKey];
[[NSUserDefaults standardUserDefaults] setInteger:1000 forKey:kMarksDistanceKey];
[[NSUserDefaults standardUserDefaults] setInteger:5 forKey:kAccuracyKey];
```

També s'emmagatzemen els valors d'accés a la pàgina web de l'usuari quan els defineix a la vista de preferències.

4.1.3 Vistes

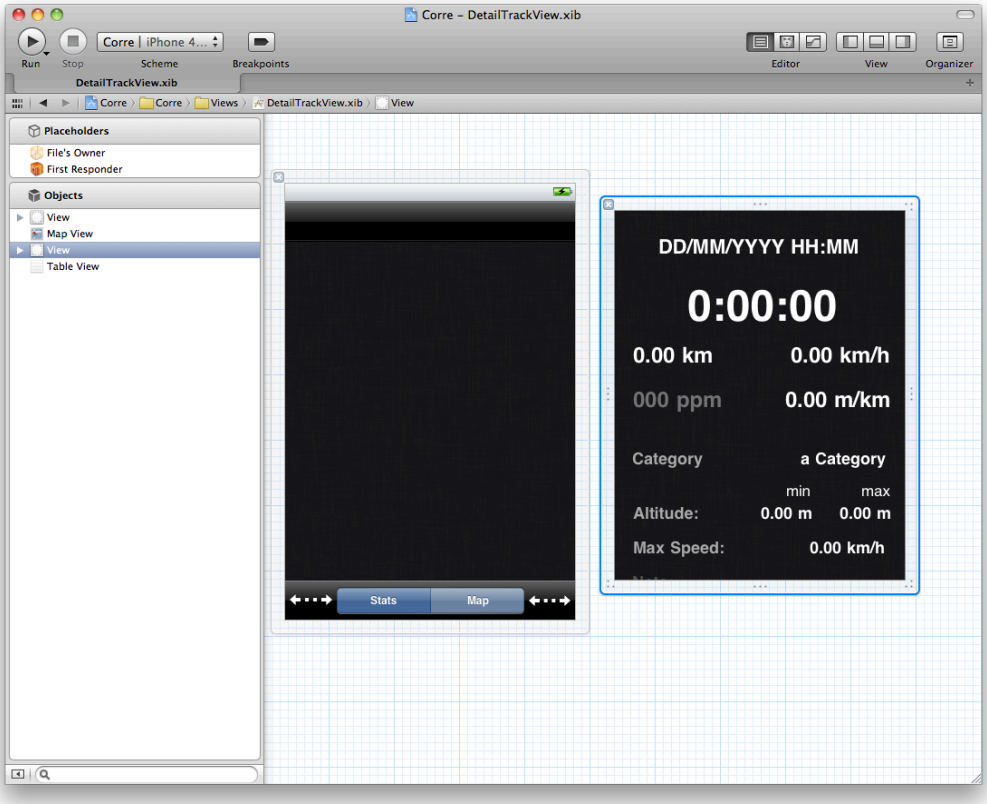
Les vistes estan realitzades mitjançant la ferramenta Interface Builder, consistent en els arxius XIB del grup *Views*.

Interface Builder permet de forma gràfica situar els diferents elements de les vistes i connectar-los a les propietats dels controladors que siguen necessàries. Aquesta informació dels elements de la vista i les seves propietats s'emmagatzemen a un arxiu amb extensió XIB en format XML.

Les vistes i les seues funcions són les següents:

- **MainWindow:** conté la barra de pestanyes inferior principal, definint els elements que conté així com les vistes que ha de carregar cada element.
- **NewTrackView:** a aquesta vista es disposen els elements de cronòmetre i botó per iniciar i aturar un enregistrament.
- **TracksTableView:** conté la vista de tabla on es mostren els recorreguts enregistrats. Cal dir que la barra de navegació la carrega el controlador de la vista de pestanyes, i és aquesta barra qui carrega la vista de tabla. Això permet la navegació entre la vista dels recorreguts i el seu detall intercanviant les vistes TracksTableView i DetailTrackView.
- **DetailTrackView:** aquesta interfície conté les vistes d'estadístiques, de taula per a les marques i de mapa. És mitjançant el seu controlador (DetailTrackViewController) com es situen les diferents vistes al seu contenidor i s'intercanvien responent a la interacció de l'usuari.
- **SettingsView:** aquesta interfície conté la vista de taula que presenta les preferències així com els elements per recollir els valors de nom d'usuari, contrasenya (UITextField) i l'enregistrament de marques (UISwitch). Per recollir els valors de distàncies entre marques o exactitud es fa ús de la interfície auxiliar SetupPickerView.
- **SetupTableView:** presenta la taula amb les categories. Si es vol afegir una nova categoria es fa ús de la interfície auxiliar AddTextView.

(figura a la pàgina següent)



Les interfícies descrites responen a les funcionalitats principals de l'aplicació, però com s'ha indicat hi ha interfícies desenvolupades per poder realitzar funcions auxiliars de l'aplicació. Aquestes interfícies són:

- **EditTrackView:** que presenta els camps que es poden editar d'un recorregut.
- **AddTextView:** presenta un camp de text per poder introduir una nova categoria.
- **SetupPickerView:** presenta la interfície amb un un camp de text d'ajuda i del nom de l'element a escollir genèric, que es pot personalitzar segons el tipus de valors que s'assignen. També es presenta un element per seleccionar entre els diferents valors.
- **ConflictsTableView:** presenta una vista de taula amb els conflictes a resoldre per la sincronització.

Decisions d'implementació

A les interfícies aproximades esmentades a la fase de disseny es mostrava una vista per a l'acció de sincronitzar les dades. Aquesta vista tenia únicament la funció de mostrar el progrés de la sincronització.

A la implementació s'ha cregut més convenient llevar-la de forma que només es presenten tres pestanyes principals i l'acció de sincronitzar es situa a la vista de llistat dels recorreguts.

Per mostrar a l'usuari que s'està realitzant l'operació de sincronització es mostra l'indicador d'activitat a la part superior de la pantalla.

4.1.4 Controladors

El conjunt de controladors contenen la lògica de l'aplicació, aportant les funcionalitats necessàries a les vistes així com la interacció amb el model.

A continuació descrivim els controladors agrupats pels diferents grans blocs de funcionalitat de l'aplicació.

New Track

La carpeta conté les classes NewTrackViewController que encapsula la lògica d'enregistrament d'un nou recorregut i la classe *singleton* TrackLocationManager. Aquesta última es descriurà més endavant.

NewTrackViewController

La funció de la classe consisteix a iniciar un temporitzador que actualitza el cronòmetre i l'enregistrament de les noves posicions que retorna el gestor de localització. La classe mostra la informació de temps i rep les accions de la vista NewTrackView.

Per realitzar aquestes tasques necessita per una banda d'unes propietats que aportaran la funcionalitat de captura de localització i temporitzador i per altra els mètodes que realitzen les accions.

Les propietats principals són:

- **managedObjectContext:** instància de la classe NSManagedObjectContext per poder crear i emmagatzemar nous recorreguts.
- **sharedTrackLocationManager:** instància de la classe *singleton* TrackLocationManager que gestiona l'inici i atur de la captura de localitzacions. S'ha d'implementar el seu protocol TrackLocationManagerProtocol per rebre les actualitzacions de posició.
- **timer:** instància de la classe NSTimer que permet configurar un disparador cada cert temps definit que cride una funció.

Per una banda s'implementen el mètode del protocol TrackLocationManagerProtocol:

- **trackLocationManager:didUpdateToLocation:fromLocation :** aquest mètode comprova si s'està enregistrant (tracking = YES) i actualitza els valors de posició cridant els mètodes d'actualització adhients.

Els mètodes principals de la classe són:

- **startTracking**: es marca que s'està enregistrant amb el valor *tracking*, inicialitza els valors per a un nou recorregut i marca (si cal) i inicia el temporitzador que crida cada segon la funció updateTimer (únicament actualitza les etiquetes del cronòmetre).
- **stopTracking:** atura l'enregistrament del recorregut aturant el temporitzador, establint el valor *tracking* com a que no s'està enregistrant i emmagatzema els valors a la base de dades.
- Mètodes auxiliars (Location helpers): els mètodes actualitzen els valors del recorregut i l'interval que s'està enregistrant en eixe moment.

Tracks

El grup Tracks situat a l'entorn de desenvolupament engloba els controladors que implementen les funcions de gestió dels recorreguts a l'aplicació: llistar-los, mostrar el seu detall, modificar dades, esborrar-los i iniciar la sincronització.

TracksTableViewController

El controlador és una sublcasse de UITableViewController, per tant per tant no s'han de declarar els protocols que implementa per a la gestió de la taula, però sí s'han d'implementar els mètodes necessaris per carregar i gestionar les dades.

La classe té les següents funcionalitats:

- Implementa el protocol per gestionar l'edició dels elements de la taula així com de la presentació de les dades.
- Implementa la funcionalitat per sincronitzar les dades amb el servidor. Per aquesta tasca ha d'implementar el protocol SyncManagerProtocol que permet cridar a la classe que encapsula les funcionalitats de sincronització.
- Implementa el protocol ConflictsTableViewController que permet la comunicació amb la vista ConflictsTableView i quan l'usuari acaba de realitzar aquesta tasca.

Els mètodes principals són:

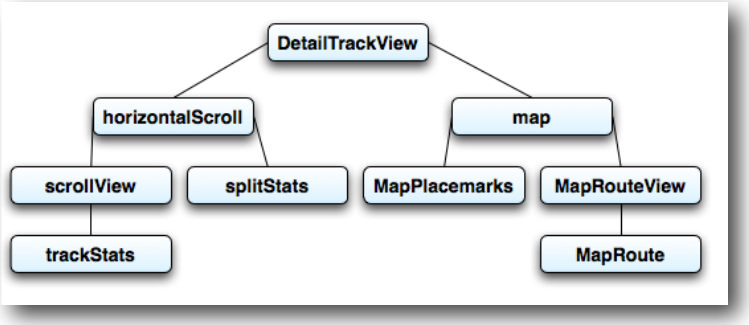
- **syncronizeTracks:** crida el mètode de sincronització a la instància del gestor de sincronització assignant-se com a delegat. D'aquesta forma quan el gestor acabe la tasca li ho comunicarà.
- **showConflictsAndSync:** el mètode recupera els possibles conflictes que puguen haver, encara que estiguen resolts, per mostrar-los a l'usuari mitjançant la vista de conflicts. Si no hi ha crida a la funció de sincronització de tasques.
- **checkConflicts:** funció que comprova si els conflictes estan resolts. Si ho estan es presenta una alerta a l'usuari per si vol tornar a sincronitzar.

Per gestionar les dades de la taula els mètodes a implementar es divideixen en tres grups per la seua funcionalitat:

- Dades: on s'implementen els mètodes que retornen el número de seccions de la taula, el número de files i la cel·la de cada fila.
- Interacció: El mètode que es crida quan es selecciona una cel·la.
- Edició: els mètodes que implementen la lògica quan la taula es mostra en mode d'edició.

DetailTrackViewController

La classe principalment gestiona les diferents vistes que presenten la informació d'un recorrregut: les seues estadístiques, marques i el mapa.



Aquestes són:

- **horizontalScroll:**
  - **scrollView:** que conté la vista amb les dades del recorregut.
  - **splitStats:** vista de taula que mostra les dades de les marques (si hi ha).
- **map:** que conté la vista de mapa que presenta els punts d'inici, fi i intervals i el recorregut.

L'intercanvi entre les vistes es realitza amb el control segmentat de la vista DetailTrackView i el mètode pageControlChanged.

Menció especial requereix la vista de mapa i les tasques necessàries per mostrar la informació.

Els passos a nivell general consisteixen en:

1. Centrar el mapa a la regió on es situa el recorregut consultat.
2. Situar els marcadors d'inici i fi. Si hi ha marcadors d'interval situar-los.
3. Situar la capa amb la ruta dibuixada sobre el mapa visualitzat.

Per centrar el mapa on es situen els punts de localització s'ha implementat el mètode getTrackCoordinateRegion que recorre totes les coordinades del recorregut i extrau el punt situat més a la part superior i esquerre així com l'inferior dret. D'aquesta forma es pot establir la regió amb afegint un poc d'espai que ha de mostrar el mapa.

Per situar els indicadors de posició s'ha implementat la classe MapPlacemark que segueix el protocol MKAnnotation. Aquest protocol defineix que la classe ha de tindre les següents propietats:

- **coordinate:** coordenada on situar la marca.
- **title:** títol per a la marca.
- **subtitle:** subtítol per a la marca.

També s'ha definit la propietat de tipus de marca per variar els colors:

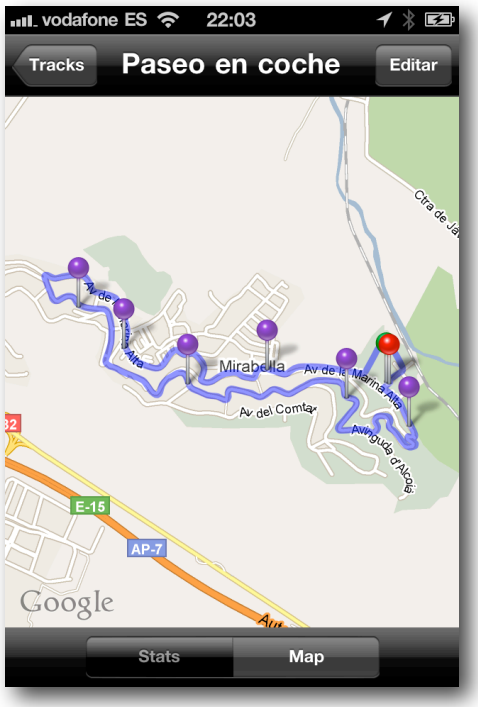
- Verd: marca d'inici.
- Roig: marca de fi.
- Violeta: marques d'interval.

Per últim, per dibuixar la ruta sobre el mapa calen dos elements:

- Una classe que implemente el protocol MKOverlay: MapRoute

- Una subclasse de MKOverlayPathView que implemente el mètode createPath: MapRouteView

La classe MapRoute realitza la principal funció al mètode d'inicialització:



- El mètode d'inicialització de la classe rep per paràmetre els diferents punts de la ruta en el sistema de coordenades terrestres.
- De cada punt realitza la conversió del sistema de coordenades terrestres a punts en dos dimensions que es situen a la vista de mapa (MKMapPoint).
- Aquests punts els emmagatzema a un vector i es defineix l'àrea on es situen els punts a la propietat boundingMapRect.

La classe MapRouteView implementa el protocol consistent en definir els mètodes necessaris per dibuixar mitjançant instruccions OpenGL la ruta al context. Aquests mètodes són:

- **createPath:** recupera els punts de la propietat *overlay* (corresponent a la instància de la classe MapRoute amb la ruta), crea una ruta CGPath on va afegint els punts del vector recuperat.
- **applyStrokePropertiesToContext:atZoomScale :** aquest mètode defineix les propietats de la línia amb la què es dibuixarà la ruta com són el color, l'amplària i si els cantons i extrems es volen redondejats.
- **strokePath: inContext :** conté les instruccions per dibuixar la ruta al context OpenGL.
- **drawMapRect: zoomScale: inContext :** mètode que es cridat per dibuixar la ruta al mapa. Fa ús dels mètodes anteriors amb els següents passos: crear la ruta, aplicar les propietats de la línia i dibuixar al context.

EditTrackViewController

El controlador implementa els mètodes necessaris per gestionar la vista d'edició d'un recorregut. Es crea amb una instància de recorregut d'on recupera les dades a mostrar a la vista per editar-les.

Defineix el protocol EditTrackViewControllerProtocol per poder comunicar-se amb la instància que li invoque quan acaba de modificar les dades. D'aquesta forma el controlador pot executar alguna acció quan s'acabe d'editar.

Per exemple, a l'aplicació és el controlador DetailTrackViewController qui crida al controlador per editar les dades del recorregut que s'està visualitzant. Quan rep l'avís que l'edició ha finalitzat pot refrescar les dades actualitzades a la pantalla.

Els mètodes principals són:

- **save:** emmagatzema els canvis a la base de dades, invoca el mètode del protocol i retira la vista del dispositiu.

- **cancel:** retira la vista sense aplicar els canvis.

SelectCategoryTableViewController

El controlador implementa els mètodes necessaris per presentar les diferents categories a una vista de taula amb la categoria actual marcada.

El controlador quan una cel·la es selecciona invoca el mètode del seu delegat per indicar que ha finalitzat l'edició i quin valor ha seleccionat l'usuari.

Aquest controlador no té una interfície de l'Interface Builder associada, sinó que defineix els elements mitjançant el seu constructor de classe initWithTrack: delegate.

ConflictsTableViewController

El controlador implementa la lògica per mostrar els conflictes recuperats del model de dades i emmagatzemar les eleccions de l'usuari mitjançant el mètode saveChanges.

Els conflictes es presenten en parelles per a que l'usuari trie quines dades vol que s'emmagatzemen.

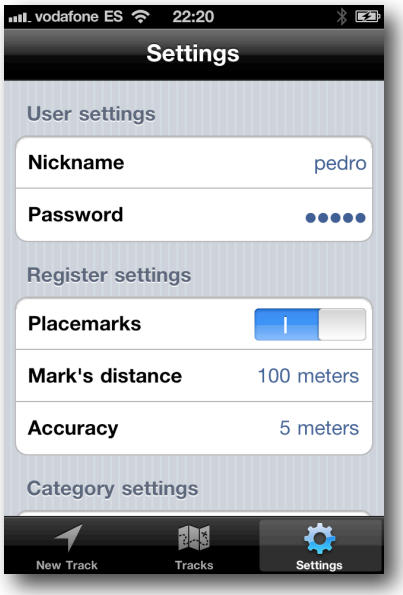
L'opció triada es senyala amb una marca. Si l'usuari torna a seleccionar pot llevar la marca i el conflicte quedaria sense resoldre.

Per comunicar la instància que ha invocat al controlador que ha finalitzat es defineix el protocol ConflictsTableViewControllerProtocol.

Settings

Al grup settings es situen els controladors que implementen la lògica per establir les preferències d'usuari.

SettingsViewController



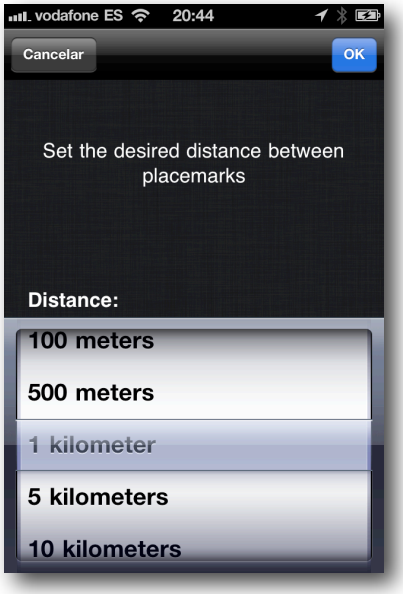
El controlador implementa la lògica per mostrar les preferències de l'usuari a una vista de taula.

Per mostrar-les aquest controlador personalitza les cel·les per a les dades d'usuari i registre de marques:

- Dades d'usuari: contenen el camp de text corresponent segons siga nom d'usuari (userNicknameTextField) o contrasenya (userPasswordTextField), des d'on es poden recuperar les dades introduïdes per l'usuari.
- Registre de marques: contenen l'interruptor togglePlacemarksSwith per gestionar si es vol registrar o no.

Per poder modificar les distàncies de filtre o entre marques a partir d'un conjunt de valors definit es fa ús de la vista SetupPickerView. Així s'implementa el seu protocol per rebre les dades triades quan aquesta finalitza.

SetupPickerViewController



Implementa la lògica per mostrar un conjunt de valors i retornar-los al seu delegat.

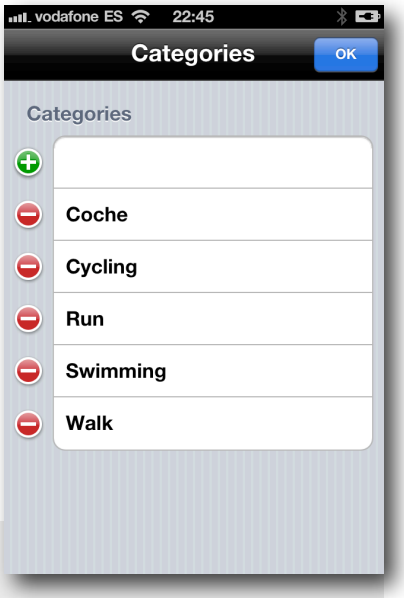
La vista consta de l'element UIPickerView que mostra un conjunt de dades simulant una roda. Per gestionar aquest element s'ha d'implementar els següents protocols:

- **UIPickerViewDelegate:** s'ha d'implementar el mètode que s'invoca quan la selecció ha canviat.
- **UIPickerViewDataSource:** s'ha d'implementar els mètodes que retornen el número de valors a mostrar així com els valors.

El controlador s'inicialitza amb un vector en que cada element és un diccionari amb dos claus: OptionName i Value. Aquestes contenen el nom de l'opció a mostrar per pantalla i el valor associat.

Quan l'usuari acaba la selecció el controlador retorna al seu delegat el valor seleccionat.

SetupTableViewController



Implementa la lògica per mostrar les dades a una vista de taula i editar-les.

Aquest controlador també és subclasse de UITableViewController, com el controlador TracksViewController. Per tant s'implementen els mètodes necessaris per gestionar la taula.

La particularitat d'aquest controlador es troba quan la taula es mostra en mode d'edició. S'afegeix una nova cel·la buida al principi de la taula per indicar que es poden afegir elements. Si l'usuari polsa a aquesta cel·la es mostrarà la vista AddTextViewController que retorna un valor de text quan finalitza l'edició. Aquest valor s'afegeix a la taula.

```
- (void)setEditing:(BOOL)editing animated:(BOOL)animated
{
    if (editing) {
        [[self navigationItem] setHidesBackButton:YES animated:YES];
                        NSManagedObject    *blankCategory    =    [NSEntityDescription
i n s e r t N e w O b j e c t F o r E n t i t y F o r N a m e : @ " C a t e g o r y "
inManagedObjectContext:self.managedObjectContext];
        [blankCategory setValue:@"" forKey:@"name"];
        [data insertObject:blankCategory atIndex:0];
          [[self tableView] insertRowsAtIndexPaths:[NSArray arrayWithObject:[NSIndexPath
indexPathForRow:0 inSection:0]] withRowAnimation:UITableViewCellEditingStyleInsert];
        NSLog(@"Estamos editando");


    } else {
        [[self navigationItem] setHidesBackButton:NO animated:YES];
        Category *blankCategory = [data objectAtIndex:0];
        [managedObjectContext deleteObject:blankCategory];
        [data removeObjectAtIndex:0];
          [[self tableView] deleteRowsAtIndexPaths:[NSArray arrayWithObject:[NSIndexPath
indexPathForRow:0 inSection:0]] withRowAnimation:UITableViewCellEditingStyleDelete];
        [managedObjectContext save:nil];
        NSLog(@"Dejamos de editar");
    }

    [super setEditing:editing animated:animated];
}
```

Quan s'acaba d'editar, aquesta primera cel·la buida s'elimina.

AddTextViewController

Implementa la lògica per recollir un valor del seu camp de text i retornar-los al seu delegat.

Defineix el protocol AddTextViewControllerProtocol. Aquest protocol permet que el valor introduït per l'usuari al quadre de text es retorne al delegat.

4.1.5 Classes singleton

Encara que aquestes classes s'emmarcarien amb els controladors s'ha cregut convenient separar-les donat que no aporten lògica a vistes concretes sinó que aporten funcionalitats a diferents parts de l'aplicació.

TrackLocationManager

TrackLocationManager defineix el protocol TrackLocationManagerProtocol que permet passar dades de localització al seu delegat quan aquests s'actualitzen. Per gestionar aquestes actualización implementa el protocol CLLocationManagerProtocol.

Aquesta classe també defineix els següents mètodes:

- **wakeUpLocationManager:** que s'encarrega d'inicialitzar el gestor de localització si no ho està ja i iniciar l'actualització de les posicions (startUpdatingLocation). Aquesta funció la crida el delegat de l'aplicació CorreAppDelegate quan l'aplicació torna a estar activa (applicationDidBecomeActive).
- **shutDownLocationManager:** si la instància del gestor de localització existeix atura l'actualització (stopUpdatingLocation) i el marca com a valor nul.

SyncManager

La classe SyncManager encapsula les funcions necessàries per enregistrar els canvis que s'han de sincronitzar i el procés de sincronització.

Els mètodes principals són:

- **registerSyncTaskType: forManagedObject :** crea un registre a la base de dades amb el tipus de tasca a realitzar i la referència[25] a l'objecte.
- **dataToSendJSON:** recupera les tasques a sincronitzar i les processa obtenint de cada objecte referenciat la seva representació JSON. Retorna totes les dades a enviar en format JSON.
- **syncRequest:** crea la petició HTTP per ser enviada al servidor amb les credencials de nom d'usuari i contrasenya i les dades enformat JSON a enviar.
- **connectionDidFinishLoading:** quan el dispositiu rep la resposta del servidor aquest mètode processa el seu resultat, si ha sigut de error alerta a l'usuari sinó crida la funció syncResponse per processar les dades.
- **syncResponse:** de les dades rebudes es fan les següents operacions:
  1. Es processen les tasques rebudes des de'l servidor fent les modificacions necessàries. Si detecta una tasca amb el valor d'estat 1 indica que hi ha un conflicte per tant ho modifica al dispositiu.
  2. Es recuperen les tasques del dispositiu i si no estan indicades com a que existeix un conflicte (estat 1) les esborra.
  3. Es comprova si al final de tot el procés hi ha conflictes.
  4. Invoca el mètode del delegat definit al protocol indicant la existència de conflictes.
- **clearAllSyncTasks:** esborra totes les tasques de sincronització enregistrades.
- **createAllNewSyncTasks:** processa tots els recorreguts i categories per registrar les seues tasques de creació. Aquest mètode s'empra quan l'usuari canvia de nom

[25] ObjectURI: és la ruta amb que es defineix de forma única un registre al sistema de base de dades. Amb aquesta referència es pot recuperar un NSManagedObjectID a través del coordinador de persistència i així recuperar l'objecte original.

d'usuari i contrasenya al sistema, per si vol tornar a carregar totes les dades en net a la pàgina web.

4.1.6 Comunicació dispositiu-servidor

Per l'enviament de les dades a través de la xarxa entre el dispositiu i el servidor finalment s'ha optat per serialitzar les dades en format JSON.

El format de la petició enviada des del dispositiu iniciant la sincronització es divideix en tres blocs:

- Identificació de l'usuari: dades necessàries per l'accés al sistema de l'usuari.
- Tasques de sincronització a realitzar: llistat amb les tasques a realitzar.
- Dades relacionades: dades necessàries per cada tasca.

```
JSONdata =  {  "credentials" : { "Nickname" : ... , "Password" : ... },
               "tasks" :   { 0 : { ... SyncTask en format JSON ... },
                           ... ,
                           },
               "tasksData":   {
                                 "objectURI" : { ... dades en format JSON ... },
                                 ...
                              }
            }
```

El servidor respon amb la següent estructura:

- Resultat de les operacions.
- Tasques de sincronització des del servidor.
- Dades de les tasques a realitzar.

```
JSONdata =  {  "resultCode" : codi de resultat,
               "log"  : cadena amb missatge,
               "tasks" :   { 0 : { ... SyncTask en format JSON ... },
                           ... ,
                           },
               "data":     {
                              "objectURI" : { ... dades en format JSON ... },
                              ...
                           }
            }
```

S'ha emprat per a la funció generar les representacions JSON a partir de diccionaris i a l'inrevés el *framework* JSON-framework[26] desenvolupat per Stig Brautaset.

4.1.7 Internacionalització

L'aplicació s'ha implementat amb l'anglès com a llenguatge base. Posteriorment s'ha traduït la interfície al català i castellà.

[26] JSON-framework (http://stig.github.com/json-framework/)

Per traduir una aplicació s'ha d'incloure la carpeta corresponent a cada idioma amb els arxius que contenen la traducció. Aquestes carpetes tenen el patró codi_llengua.lproj. Per exemple els arxius que contenen la traducció al català estan a la carpeta ca.lproj.

La internacionalització d'una aplicació es realitza per dos costats:

- Les interfícies definides a Interface Builder.
- Les cadenes de text incloses al codi.

Les interfícies s'inclouen als directoris de les diferents localitzacions i es tradueixen: si hi ha tres idiomes tindrem tres arxius MainWindow.xib per exemple.

Les cadenes de text incloses al codi es generen mitjançant la funció NSLocalizedString on s'indica la cadena a l'idioma base i un comentari d'ajuda.

```
NSLocalizedString(@"Connection error", @"Alert view title - Error"
```

Després es generen mitjançant la comanda genstrings un arxiu Localization.strings que contenen les traduccions. L'arxiu s'estructura de la següent forma: el comentari indicat, la cadena base i la corresponent traduïda:

```
/* Alert view title - Error */
"Connection error" = "Error de connexió"
```

L'aplicació estableix l'idioma a mostrar segons l'idioma definit al dispositiu.

4.1.8 Decisions d'implementació

Model de dades

A la fase de disseny es va definir una relació entre l'entitat Recorregut i OperacioSinc. Aquesta relació s'establí per permetre recuperar les dades a sincronitzar a partir de les tasques pendents.

Una de les característiques del sistema de permanència és que per a cada instància d'un objecte gestionat el gestor assigna un identificador únic al sistema de permanència: objectURI. Amb aquest identificador es pot recuperar la instància de l'objecte realitzant una consulta al gestor de permanència del sistema.

Així l'entitat OperacioSinc queda sense la relació, però amb el identificador únic de les dades a sincronitzar. Amb aquesta dada es pot recuperar les dades directament i podem emmagatzemar la referència a dades heteorgènies.

TrackLocationManager

En un principi la lògica per interactuar amb el gestor de localització es va associar a l controlador NewTrackViewController.

Per assegurar que l'enregistrament continuava encara que el dispositiu es posara en repòs o s'emprara una altra aplicació es va habilitar el mode de tasca en segon pla de localització.

Aquest fet provocava que quan l'aplicació es delega a un segon pla, però no s'està enregistrant continuara estant la instància del gestor de localització en marxa. Per solucionar-ho es va implementar a la funció del delegat de l'aplicació applicationDidEnterBackground la comprovació que si l'aplicació no està enregistrant deshabilitara el gestor de localització. Per deshabilitar-ho recupera la instància de NewTrackViewController i alliberava la propietat que conté el gestor.

Això generava un problema: si l'aplicació tornava del segon pla i la vista no era NewTrackView no permetia a NewTrackViewController tornar a inicialitzar el gestor de localització.

Per tant, es va creure convenient passar aquesta gestió a una classe *singleton* independent de cap vista que fóra accedida per l'aplicació i per la instància de NewTrackController.

Llibreria JSON

Per tractar les cadenes JSON s'ha inclòs la llibreria desenvolupada per Stig Brautaset[27].

Aquesta llibreria permet la transformació des d'estructures de dades amb parells clau-valor com són els diccionaris i vectors a una cadena en format JSON i a l'inrevés.

L'opció d'emprar una llibreria desenvolupada per tercers obeeix al risc R2 "Manca de temps per la implementació per la data de lliurament" identificat a la planificació. Encara que en un primer moment no es contempla com acció correctiva la possibilitat d'incorporar llibreries externes, s'adopta aquesta mesura per poder complir les dates de lliurament. D'aquesta forma es pot reduir el temps de codificació a parts auxiliars de l'aplicació.

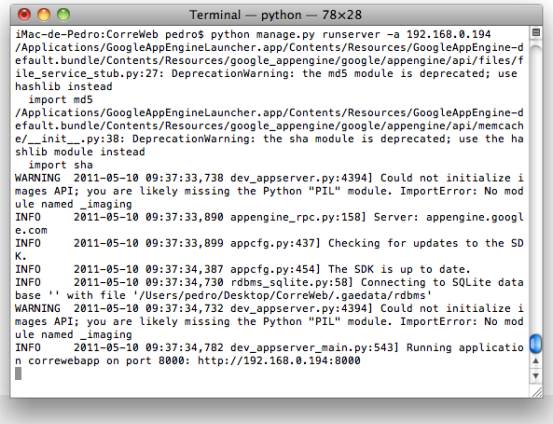
[27] http://stig.github.com/json-framework/

4.2 Pàgina web

4.2.1 Funcionament de l'entorn

El *framework* Django facilita totes les classes per generar pàgines web de forma fàcil. Per aquest motiu, directament, descomprimint el paquet i amb Python instal·lat es pot fer funcionar un servidor local.

Com aquesta pàgina web està implementada sobre l'entorn de Google App Engine s'ha fet servir una modificació de Django que permet la interacció amb la base de dades de Google App Engine.



Una volta descomprimits els paquets necessaris es pot arrancar el servidor de proves amb la següent instrucció[28]:

```
python manage.py runserver -a 192.168.0.194
```

La pàgina web es pot estructurar a mitjançant subprojectes continguts a carpetes i que són independents. A cada projecte l'estructura bàsica és:

- **urls.py:** arxiu on es defineixen les rutes web que es vinculen amb una funció que processa la petició.
- **views.py:** arxiu on es defineixen les funcions que processen peticions.

També es poden afegir altres arxius que amplien les funcionalitats

- **models.py:** arxiu on es defineixen les classes que representen el model de dades.
- **forms.py:** arxiu on es defineixen classes que representen a formularis per replegar dades.

L'aplicació web CorreWeb està situada a la carpeta "core".

A continuació es descriuen els seus components.

4.2.2 Model

El model està definit a l'arxiu models.py i conté les classes de les dades que emmagatzemarà la pàgina web:

- **Track:** dades dels recorreguts.
- **Placemark:** dades de les marques associades a un recorregut.
- **Category:** categories dels recorreguts.
- **SyncTask:** tasques de sincronització a realitzar.

Els camps continguts són similars al model definit a l'aplicació mòbil. L

[28] El paràmetre permet que el servidor siga accessible a la xarxa local a través de la direcció IP indicada. S'ha emprat per poder fer proves amb el dispositiu mòbil.
Si no s'indica cap paràmetre el servidor és accessible únicament al propi ordinador. Aquesta opció és perfectament vàlida per les proves amb l'emulador.

- **objectURI:** s'emmagatzema aquest valor del model de dades del dispositiu. Així, aquest camp permet establir la relació entre les mateixes dades a ambdues plataformes.
- **user:** s'emmagatzema la relació amb l'usuari que sincronitza les dades, permetent recuperar únicament les seues dades quan accedeix a la plana web.

Cada model es defineix com a subclasse de l'objecte Model proporcionat pel marc de treball i els camps es defineixen amb les classes pertinents per emmagatzemar els valors.

Com al dispositiu hi ha valors que es dedueixen dels valors emmagatzemats.

```
class Track(models.Model):
  user = models.ForeignKey(User)
  objectURI = models.CharField(max_length=255)
  category = models.ForeignKey(Category)
  name = models.CharField(max_length=100)
  date = models.DateTimeField()
  time = models.FloatField()
  distance = models.FloatField()
  minAltitude = models.FloatField()
  maxAltitude = models.FloatField()
  maxSpeed = models.FloatField()
  ppm = models.IntegerField(null=True, blank=True)
  note = models.TextField(null=True, blank=True)
  dataKML = models.TextField()
  updated = models.DateTimeField()

  def __unicode__(self):
    return u'%s (%s) - %s' % (self.objectURI, self.user.username, self.name)

  def distanceInKm(self):
    return self.distance / 1000

  def speedInKmH(self):
    return (self.distanceInKm() / (self.time / 3600))

  def maxSpeedInKmH(self):
    return (self.maxSpeed * 3.6)

  def pace(self):
    if (self.distance == 0):
      result = 0
    else:
      result = (self.time / 60) / (self.distanceInKm())
    return result

  # La funció formatTime() està definida a l'arxiu utils.py que conté funcions auxiliars
  def paceString(self):
    return formatTime(self.pace() * 60)

  def formattedTime(self):
    return formatTime(self.time)
```

Definició del model Track.

4.2.3 Configurador de URL

L'arxiu urls.py conté els patrons que responen a les peticions URL rebudes. Es comproven els patrons emmagatzemats al vector i si un patró encaixa amb la petició l'envia a la funció associada.

```
from django.conf.urls.defaults import *
from django.contrib.auth.views import login, logout

urlpatterns = patterns('core.views',
  (r'^$', 'index'),
  (r'^track/(?P<id>\d+)/?$', 'track'),
  (r'^track/edit/(?P<id>\d+)/?', 'edit_track'),
  (r'^track/delete/(?P<id>\d+)/?', 'delete_track'),
  (r'^track/dataKML/(?P<id>\d+)/data.kml$', 'getKML'),
  (r'^core/api$', 'sync'),
  (r'^accounts/login/$', login, {'template_name':'login/login.html'}),
  (r'^accounts/logout/$', 'logout'),
  (r'^accounts/register/?$', 'register'),
)
```

Codi de l'arxiu urls.py

Com es pot observar al codi hi ha peticions que es dirigeixen a funcions contingudes a l'arxiu de vistes del projecte core.views i també a funcions del component Auth de Django.

4.2.4 Formularis

L'arxiu forms.py conté un únic formulari de model que respon als camps que es poden editar dels recorreguts.

Els formularis es defineixen com a una subclasse de la classe Form inclosa al marc de treball Django que aporta les funcionalitats com poder transformar a codi HTML els camps, validació, gestió d'errors, etc.

```
from django import forms
from models import Track

class TrackForm(forms.ModelForm):
  ppm = forms.IntegerField(min_value=0, max_value=250, required=False)

  class Meta:
    model = Track
    fields = ('name', 'ppm', 'note')
```

Definició de la classe TrackForm corresponent al formulari d'edició dels recorreguts

4.2.5 Plantilles

A la carpeta "templates" es troben les plantilles HTML amb les què es presentaran les dades enviades des de les funcions de views.py.

El sistema de plantilles implementat per Django permet la generació de qualsevol tipus d'arxiu de text. Al cas concret de l'aplicació web es combina amb marques HTML per generar les planes on es mostra la informació.

Hi ha tres components bàsics al sistema de plantilles que s'indiquen amb els següents símbols:

- {{ variable }}: Els doble claudàtor serveixen per mostrar el contingut de la variable que tanquen.
- {% if variable %}: El claudàtor seguit del símbol del percentatge identifica a etiquetes que realitzen operacions com operadors de condició, iteradors o d'altres.

- {{ data|date:'d/m/Y' }}: l'etiqueta que segueix la barra vertical permet realitzar operacions sobre el contingut de la variable. L'exemple aplica el format indicat a una data.

```
{% if tracks %}
   {% for aTrack in tracks %}
         <a href="{% url core.views.track id=aTrack.id %}" {% ifequal aTrack.id track.id
%}class="active"{% endifequal %}>
            <p class="title">{{ aTrack.name }}</p>
            <p    class="subtitle">{{    aTrack.distanceInKm|floatformat:2    }}km    -
{{ aTrack.formattedTime }} - {% trans 'Pace' %}: {{ aTrack.paceString }}</p>
         </a>
   {% endfor %}
{% else %}
   <p>No tracks to show</p>
{% endif %}
```

Fragment de la plantilla track.html

El fragment comprova si la variable *tracks* està definida. Si ho està recorre el llistat i genera un enllaç al recorregut amb informació sobre ell.

4.2.6 Vistes

L'arxiu views.py conté la lògica que processa i retorna la informació.

L'esquema de les funcions que contenen la lògica és similar:

- Es rep la petició i els arguments associats.
- Es processa la informació, recuperen dades, efectuen operacions, etc.
- Es genera una resposta, normalment passant variables que contenen informació a les plantilles.

```
def register(request):
  if request.method == 'POST':
    form = UserCreationForm(request.POST)
    if form.is_valid():
      newUser = form.save()
      return HttpResponseRedirect('/')
  else:
    form = UserCreationForm()
    return  render_to_response('login/register.html',  {'form':  form},  RequestContext
(request))
```

Funció register definida a l'arxiu views.py

Gran part de les funcions tenen la cadena @login_required abans de la seua definició. Aquest decorador és part del component Auth de Django i indica que per poder executar aquesta funció s'ha d'estar autentificat, sinó es redirigeix a la pàgina d'accés.

```
@login_required
def index(request):
  tracks = Track.objects.filter(user=request.user).order_by('-date')
       return    render_to_response('track.html',    {'tracks':    tracks},
context_instance=RequestContext(request))
```

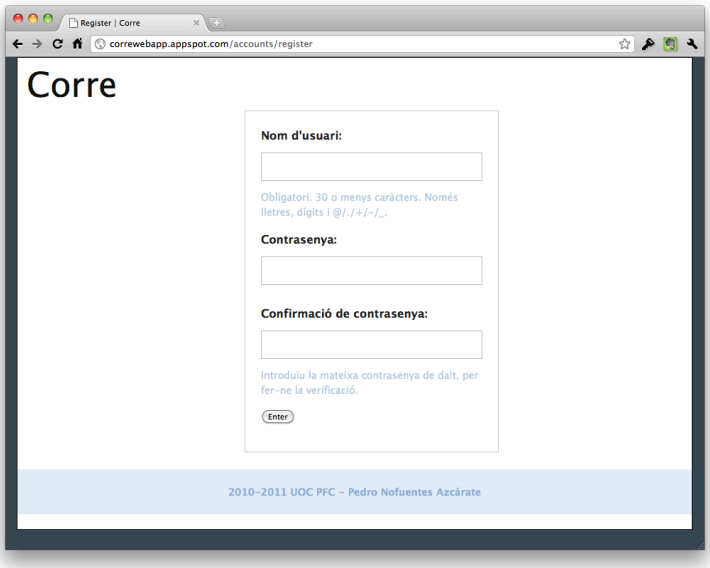
Funció index amb el decorador @login_required

A continuació es descriuen les seues funcions i les operacions que realitzen.

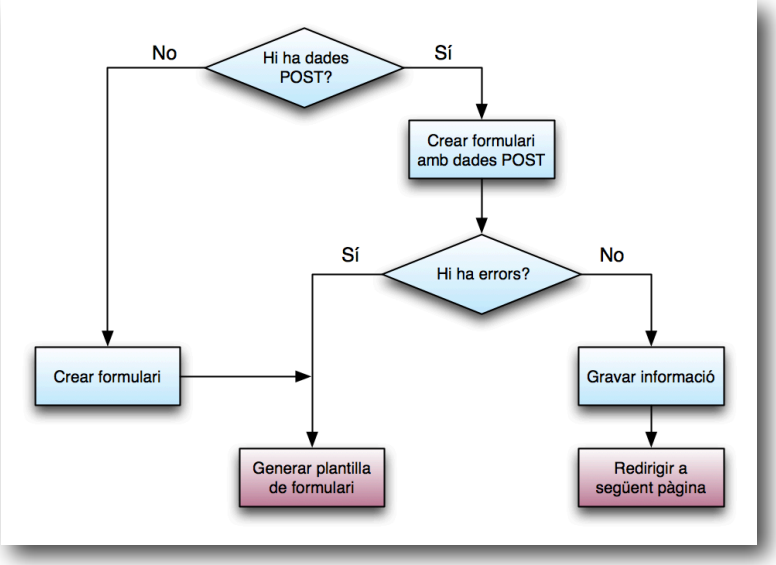
Register

La funció processa la petició i depenent de si rep la informació a través del mètode POST del protocol HTTP executa:

- Si hi ha dades: comprova que siguen vàlides. Si ho són les emmagatzema al model i redirigeix a la pàgina principal.
- Si no hi ha dades es crea una instància del formulari UserCreationForm del paquet Auth de Django i es genera la plantilla associada.



Aquest patró per processar les dades d'un formulari s'aplica sempre.



Com es veu al diagrama si la petició no conté dades es crea una nova instància del formulari (si és per editar es carreguen les dades recuperades del sistema de dades) i es genera la plana web.

Si la petició conté dades les apliquem a la variable que conté el formulari i amb els mètodes de validació d'aquest podem comprovar si són correctes.

Si ho són es realitzen les operacions necessàries i es redirigeix a una altra plana.

Si no ho són es torna a generar la plantilla, però com que no es crea de nou la instància del formulari, la variable conté els errors que s'han detectat, mostrant-se a la plana generada.

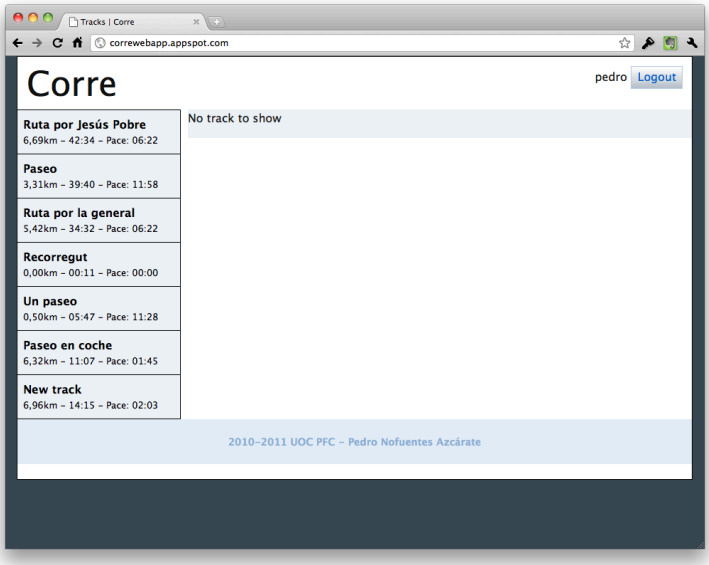
Logout

La funció logout esborra la informació d'autenticació donant per finalitzada la sessió.

Aquesta funció s'ha personalitzat donat que per defecte la funció que ve al paquet Auth redirigia a la plana de fi de sessió del panell de control.

Index

Aquesta funció executa una consulta per retornar tots els recorreguts emmagatzemats a la base de dades. Aquests recorreguts es passen a la plantilla corresponent per generar la pàgina web.



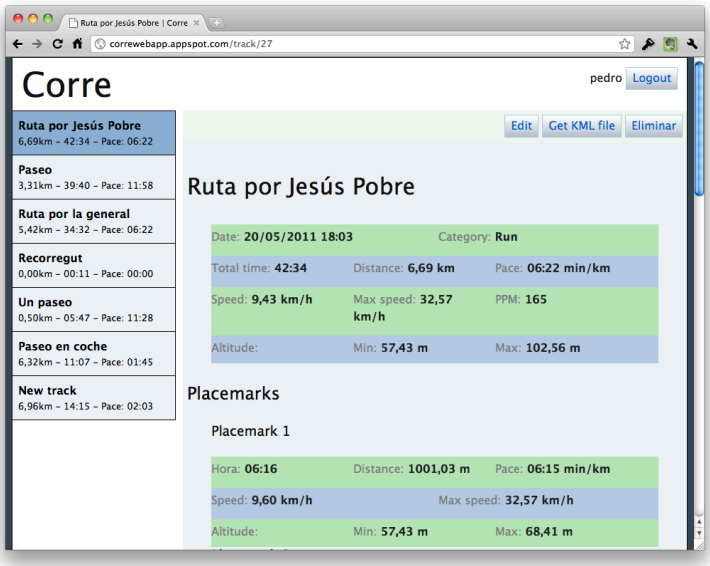
Track

La funció track rep per paràmetre un número amb l'identificador del recorregut que es vol consultar.

S'executa la consulta recuperant les dades del recorregut, si no existeix s'estableix un missatge que es mostrarà a l'usuari.

Les dades retornades per generar la plantilla són:

- Les dades del recorregut.
- Els recorreguts (per generar la columna esquerre).

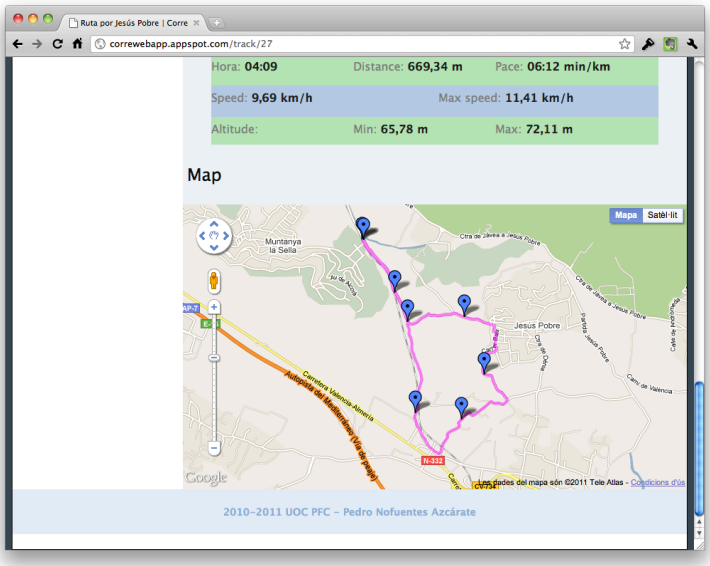


GetKML

Aquesta funció no requereix estar autenticat al sistema.

L'objectiu és retornar les dades KML associades a un recorregut per a què mitjançant l'API de Google Maps es puga superposar a un mapa. Per tant els passos són:

- Recupera les dades del recorregut identificat pel paràmetre "id" rebut.
- Crea una resposta HTTP amb les dades XML.



Aquesta petició HTTP únicament està encaminada per poder fer servir la càrrega de dades KML de l'API de Google Maps. L'API ha de poder recuperar des d'un servidor públic les dades [29].

[29] S'és conscient del problema de privadesa de dades associat a aquesta operació. Qualsevol que sabera l'identificador d'un recorregut podria recuperar els punts per on s'ha passat sense cap tipus d'autenticació.
Hi ha possibilitats per evitar-ho com poden ser generar codis que només serviren de forma temporal o altres solucions que no s'han explorat per anar més enllà de l'objectiu del projecte.

A la pantalla de detall es pot accedir a aquesta direcció web a través del botó “Get KML file” que descarrega un arxiu KML. Aquest arxiu pot obrir-se directament amb el programa Google Earth.

Edit Track

La funció permet l’actualització de les dades d’un recorregut. Les passes que realitza són les següents:

1. Recupera la informació del recorregut amb l’identificador rebut per paràmetre.
2. Si hi ha dades a la petició (POST):
   2.1. Es valida el formulari.
   2.2. S’assignen les noves dades al recorregut.
   2.3. S’emmagatzema el recorregut.
   2.4. Es comprova si ja hi ha una tasca de sincronització pendent.
      2.4.1. Si no hi ha es crea una nova.
3. Si no hi ha dades:
   3.1. Es crea una nova instància del formulari amb les dades del recorregut (les que estaven a la base de dades.
4. Es retorna el valor del recorregut (i tots els recorreguts per la columna de l’esquerra) a la plantilla corresponent.

El procés de tractament de dades de formulari és similar al comentat al de la funció register.

Delete Track

La funció esborra un recorregut seguint les següents passes:

1. Recupera el recorregut de la base de dades.
2. Recupera les marques associades al recorregut.
3. Esborra les marques.
4. Esborra el recorregut.
5. Registra una tasca de sincronització del tipus 2 (esborrar).

Sync

La funció “sync” realitza les operacions de sincronització amb el dispositiu mòbil. Aquesta funció fa una primera tasca de descodificació i posterior processament de les tasques.

Per afavorir la senzillesa del codi (en el que siga possible) s’ha creat l’arxiu api.py on estan les funcions de processament d’una tasca.

Rebut la petició amb les dades en format JSON, les passes i operacions són les següents:

1. Es comprova si l’usuari i contrasenya són correctes, si no ho són es retorna el codi d’error. Si ho són continuem.
2. Es recuperen les dades per una banda de les tasques (tasks amb claus ordenades numèricament) i per altra les dades associades (tasksData, diccionari amb cadenes ObjectURI per clau).
3. Per cada tasca a “tasks” es recuperen mitjançant l’identificador objectURI les seves dades si hi ha i es crida la funció processTask (implementada a l’arxiu api.py).

4. Si hi ha errors s'assigna el codi corresponent.
5. Es recuperen totes les tasques pendents de sincronitzar.
6. Per cada tasca per sincronitzar es preparen les dades associades.
7. Per últim es codifica en format JSON totes les dades i es retorna el resultat.

API

L'arxiu api.py conté les funcions que processen cada tasca rebuda des de'l dispositiu mòbil. La successió de passes és la següent:

1. **processTaskFromUser:** aquesta funció processa el tipus de tasca a realitzar i al tipus de dades que s'ha d'aplicar.
   1.1. Si és del tipus 0 (crear): extrau el tipus de dada i crida la funció createCategory o createTrack segons pertoque.
   1.2. Si és del tipus 1 (modificar): crida la funció modifyTrack donat que les categories des de'l dispositiu no es poden modificar.
   1.3. Si és del tipus 2 (esborrar): crida la funció corresponent al tipus de dades.

Les funcions cridades realitzen les següents operacions:

- **createCategory:** crea una nova categoria amb les dades rebudes.
- **createTrack:** crea un nou recorregut amb les dades rebudes, per cada marca rebuda a les dades crea una nova associada al recorregut amb la funció createPlacemark.
- **createPlacemark:** crea una marca amb les dades rebudes per paràmetre.
- **modifyTrack:**
  1. Si l'estat rebut és 0 (no hi ha conflicte previ) comprova si hi ha una tasca pendent per sincronitzar des de'l servidor.
     1.1. Si hi ha es marca aquesta tasca amb l'estat 1 (conflicte detectat pendent de resoldre).
     1.2. Si no es marca que les dades es poden actualitzar.
  2. Si l'estat rebut és 2 (conflicte resolt, les dades del dispositiu predominen) es marca que les dades es poden actualitzar i s'esborra la còpia de la tasca de sincronització del servidor.
  3. Si l'estat rebut és 3 (conflicte resolt, les dades del servidor predominen) s'esborra la tasca de sincronització i no es fa necessària l'actualització de dades.
  4. Si la marca d'actualització de dades és vertadera s'actualitzen les dades.
- **deleteCategory:** esborra la categoria rebuda per paràmetre i si hi ha una tasca pendent de sincronitzar s'esborra (les operacions d'esborrat predominen sobre les altres).
- **deleteTrack:** com a la funció anterior s'esborra el recorregut i les marques associades, així com possible tasca de sincronització pendent.

4.1.7 Internacionalització

S'ha habilitat la internacionalització a la pàgina web.

Per traduir la interfície de la pàgina es marquen les cadenes a traduir amb una etiqueta o una funció depenent si està a la plantilla o als arxius de codi Python.

A la plantilla s'ha d'habilitar la internacionalització amb la corresponent etiqueta i després marcar les cadenes. Per exemple:

```
{% load i18n %}

...
<p>{% trans 'No tracks to show' %}</p>
...
```

Per fer ús de la traducció als arxius amb codi Python es fa ús de la funció ugettext. Les cadenes passades per argument es buscarà les seues traduccions als arxius corresponents.

```
# S'empra l'alias _ per facilitar l'ús repetit de la funció
from django.utils.translation import ugettext as _

...
      request.user.message_set.create(message=_('The data was saved successfuly'))
...
```

Per generar els arxius amb les traduccions s'empra la comanda:

```
manage.py createmessages -l ca
```

Aquest mètode genera l'arxiu django.po situtat a la carpeta locale/ca/LC_MESSAGES. Cal observar que 'ca' indica el codi d'idioma, en aquest cas el català.

La cadena a traduir a l'arxiu és:

```
...
#: views.py:87
msgid "The data was saved successfuly"
msgstr "Les dades s'han desat correctament"
...
#: templates/edit_track.html:15
msgid "No tracks to show"
msgstr "No hi ha recorreguts per mostrar"
...
```

Per últim s'han de compilar els arxius amb les traduccions. Per aquesta tasca emprarem la següent comanda:

```
manage.py compilemessages -l ca
```

Generant un arxiu django.mo a la mateixa carpeta que l'arxiu amb les traduccions.

El mòdul Locale de Django determinarà segons la configuració de l'idioma definit a les preferències del navegador de l'usuari quin idioma mostrar.

4.1.8 Auth

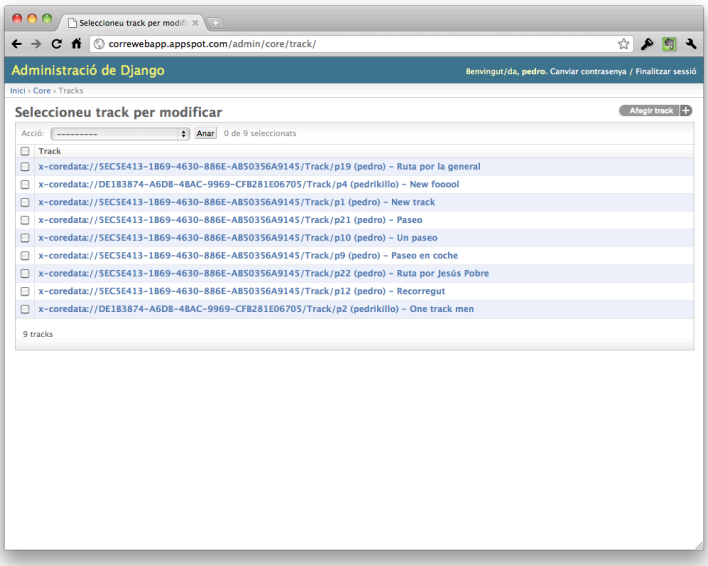
Django facilita el mòdul *Auth* per la gestió de l'autenticació al sistema. Aquest mòdul aporta funcionalitats per gestionar l'autenticació, control de sessions, emmagatzemament de dades dels usuaris, etc.

Per a la pàgina web s'han emprat principalment per a les següents funcions:

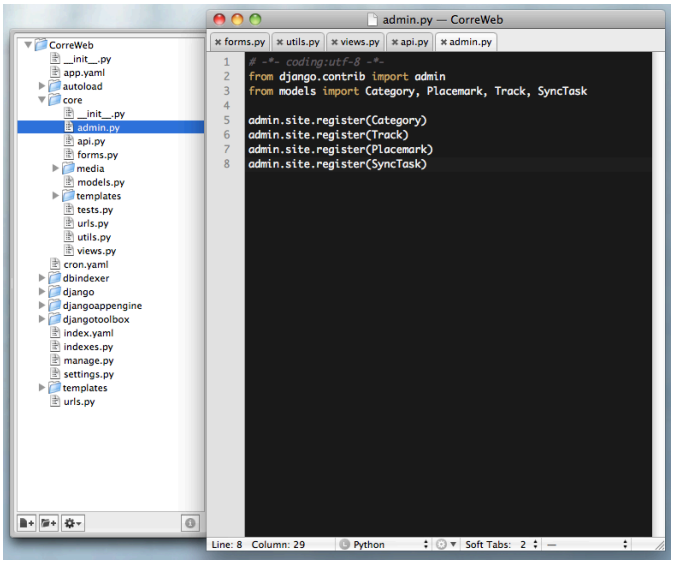
- Model User que conté informació dels usuaris de la pàgina web.
- Gestió de l'autenticació amb les funcions login i logout del mòdul.
- Formulari UserCreationForm del mòdul que conté els camps necessaris per al registre i la seua validació.
- Gestió de la sessió i missatges passats a l'usuari mitjançant aquest mètode.

4.1.9 Admin

El mòdul Admin de Django facilita una interfície per poder gestionar mitjançant formularis les dades contingudes a la base de dades.



Per definir les dades que es volen gestionar es crea l'arxiu admin.py on es registren els models a gestionar:



5. Línies obertes

5.1 Aplicació mòbil

Les línies de millora de l'aplicació mòbil les diferenciem entre millores de les funcionalitats implementades i les funcionalitats que es podrien afegir a les ja existents.

5.1.1 Aspectes a millorar

Lògica de l'aplicació

Autenticació amb el servidor

La solució implementada és molt insegura, i amb qualsevol ordinador amb un programa per rastrejar paquets pot obtindre la informació d'usuari i contrasenya del sistema.

Per evitar-ho es poden implementar dues solucions complementàries: establir connexió segura amb el servidor per a què les dades s'envien xifrades i implementar un protocol que evite l'enviament de la contrasenya i inclús l'emmagatzemament d'aquesta al dispositiu mòbil.

Per implementar aquesta segona solució es aplicar el protocol OAuth[30]. El funcionament d'aquest bàsicament consistiria:

2. L'aplicació mòbil envia una petició d'accés al servidor.
3. L'usuari és redirigit a la pàgina web del servidor on es demana que s'identifique i se li pregunta si permet l'accés de l'aplicació mòbil a les seues dades.
4. Si l'usuari ho ha permès, l'aplicació mòbil genera una petició amb una fitxa (*token*) generada amb informació d'identificació de l'aplicació i de l'usuari (sense necessitat de la seua contrasenya).

El servidor tindrà una identificació de l'aplicació mòbil. D'aquesta forma es pot implementar l'accés a més aplicacions de diferents tipus o desenvolupadors.

En qualsevol moment es pot implementar que l'usuari revoque el permís per a alguna aplicació.

Refinament de codi

Per codificar per als dispositius mòbils un dels elements que s'ha de tindre en compte és que els recursos de processador i memòria són més limitats que a un ordinador. Per aquest motiu a l'entorn d'iOS la gestió de memòria és responsabilitat del programador, per així aprofitar al màxim els recursos.

El millor aprofitament dels recursos permet que la resposta de l'aplicació siga més lleugera i que l'usuari no perceba retards.

El codi de l'aplicació mòbil és millorable amb els següents aspectes:

- Millora dels algorismes per evitar consultes o iteracions innecessàries.

[30] http://oauth.net/
Aquest protocol s'implementa a moltes aplicacions web. Per exemple Twitter ho empra per permetre a aplicacions de tercers accedir a operacions de publicació, mostra de dades d'un usuari, etc.

- Millor gestió de la memòria evitant reserves de memòria d'instàncies que es poden evitar.
- Alliberament de memòria d'objectes que no es necessiten o es poden recuperar més endavant.

També s'ha detectat l'enregistrament d'alguna dada errònia donada per la localització del dispositiu. S'hauria d'implementar algun tipus de filtratge d'aquestes dades errònies.

Interfície

La interfície és l'element amb per el què l'usuari rep la informació de l'aplicació i amb el què l'usuari interactua. Per tant és de vital importància per a que l'aplicació responga a les necessitats de l'usuari.

Presentació de la informació

Les vistes de l'aplicació mòbil que presenten informació són millorables per a que aquesta siga millor interpretada o trobada per l'usuari.

- **Vista de nou recorregut:** es pot afegir una imatge de fons que junt amb les etiquetes de temps simule un cronòmetre. Aquesta interfície és familiar per l'usuari i facilita la interpretació de la informació.
- **Vista de detall:** com a la vista anterior es pot simular amb una imatge de fons una taula on s'estructure la informació. A més es poden afegir icones, com per exemple fletxes al cas de valors màxims o mínims.
- **Vista de mapa:** les marques tenen la possibilitat d'afegir un botó per desplegar una vista de detall d'una marca on es presenten les dades detallades.
- **Sincronització:** afegir una barra de progrés que informe de forma més adient el progrés de la sincronització.

Interacció amb l'usuari

Quan un usuari interactua amb un dispositiu basat en iOS (o altre sistema operatiu) identifica elements comuns entre aplicacions que associa a diferents operacions. Es pot dir que la presentació de certs elements impliquen una metàfora que l'usuari interpreta amb certes operacions.

Si a una aplicació es respecten aquestes metàfores s'aconsegueix que la corba d'aprenentatge de l'usuari siga molt alta, ja que identifica moltes funcionalitats en base al seu aprenentatge posterior.

Encara que s'ha procurat seguir les indicacions de la guia iOS Human Interface Guidelines d'Apple hi ha vistes que es presenten que poden millorar-se per seguir aquestes convencions i facilitar l'ús de l'aplicació.

5.1.2 Funcionalitats a implementar

A banda de les característiques comentades anteriorment hi ha funcionalitats que es poden planificar per futures versions que completen l'aplicació:

- Pausar l'enregistrament d'un recorregut.
- Indicar al mapa amb un sistema de colors la major o menor velocitat de cada tram del recorregut.
- Connectar amb xarxes socials i compartir les dades d'un recorregut o enviament per correu electrònic.

- Reproducció de música durant l'enregistrament d'un recorregut.
- Poder establir un pla d'entrenament previ amb ritmes de carrera desitjats o blocs de temps a un determinat ritme.
- Avís per part de l'aplicació de les distàncies: per exemple una vibració quan s'enregistra un interval.
- Gràfica amb l'altitud i ritme del recorregut.

5.2 Pàgina web

Com a l'aplicació mòbil es diferencien entre aspectes del producte resultant millorables i funcionalitats que es poden afegir.

5.2.1 Aspectes a millorar

Lògica de l'aplicació

Gestió de la seguretat

Un dels aspectes a millorar a tot el producte és la gestió de la seguretat de l'aplicació web, sobretot a l'intercanvi a través de la xarxa de dades de l'usuari.

El marc de treball Django implementa sistemes de seguretat per evitar per exemple injeccions de codi[31] o *Cross Site Scripting* (XSS)[32], però no es pot evitar intercepció de dades a través de la xarxa.

Per evitar-ho es pot emprar les connexions segures amb el servidor amb les peticions que impliquen l'enviament de dades sensibles entre el client i el servidor.

Un punt que també s'ha esmentat és l'enviament de les dades KML als servidors de Google per generar el mapa associat a través d'una petició que no requereix autenticació.

Per evitar-ho es pot implementar un sistema de *tokens* que permeta donar una caducitat a la direcció web, d'aquesta forma es podria evitar l'ús de la direcció per aconseguir dades de l'usuari.

Refinament de codi

Com a l'aplicació mòbil hi ha parts de codi que es poden simplificar i optimitzar com per exemple el tractament de les dades a sincronitzar.

Interfície

El disseny de la interfície es pot millorar per presentar les dades de forma més estructurada i ordenada.

Donat que els navegadors actuals suporten Javascript es pot implementar una interfície aprofitant les funcionalitats que aporten imitant controls i forma de presentació de les dades com al dispositiu mòbill.

[31] http://en.wikipedia.org/wiki/Code_injection

[32] http://es.wikipedia.org/wiki/Cross-site_scripting

Una possibilitat és emprar el marc de treball Sproutcore[33]. Aquest marc de treball aporta elements d'interfície similars a les aplicacions d'escriptori emprant el llenguatge Javascript.

5.2.2 Funcionalitats a implementar

Possibles funcionalitats per millorar l'aplicació mòbil implementada són:

- Gràfiques de les dades dels recorreguts: altitud i ritme d'un recorregut, distàncies dels recorreguts, evolució dels temps, etc.
- Gestió de les categories des de la pàgina web.
- Gestió de les dades de l'usuari.
- Connexió amb xarxes socials per compartir informació.
- Implementació de fòrums o espais de comunicació entre els usuaris del sistema.

5.3 Producte

Les aplicacions desenvolupades formen part d'un producte consistent en una xarxa d'usuaris que enregistren els seus recorreguts i els consulten.

Per tant les línies de continuïtat del sistema passaran per facilitar l'usuari funcionalitats del seu interès. Aquest fet té l'objectiu d'augmentar el número d'usuaris donat que és el que aporta valor al producte.

Per donar viabilitat al projecte es poden estudiar vies com per exemple:

- Model *freemium*: on es donen funcionalitats bàsiques de forma gratuïta però es cobra per un servei més avançat.
- Publicitat.
- Cobrament a empreses per oferir serveis a través de la plataforma.

[33] http://www.sproutcore.com/

6. Conclusions

Amb la realització del projecte s'han assolit l'objectiu principal del projecte: implementar un sistema per enregistrar i gestionar recorreguts.

Encara que no s'han pogut complir tots els requisits esmentats a la planificació, s'han implementat pràcticament quasi tots.

Amb aquesta implementació s'ha treballat amb l'entorn de desenvolupament iOS SDK aprenent el seu funcionament i treballant amb els seus components:

- Principals vistes de les aplicacions per iOS: vista de taules, de desplaçament i de mapa. Aquesta última aprofundint per mostrar les rutes.
- Navegació entre les vistes amb el controlador de pestanyes i els controladors de navegació representats per les barres inferiors i superiors respectivament.
- Persistència de les dades, per una banda emmagatzemant els arxius KML al sistema d'arxius i per altra emprant l'entorn de manipulació de la base de dades amb els objectes gestionats i el seu gestor de context.
- Ús de les peticions de connexió amb servidors per transmetre, rebre i processar dades.
- Gestió de la localització amb el gestor CoreLocationManager.
- Gestió del funcionament de l'aplicació en segon pla per no interrompre l'enregistrament de la posició.
- Ús de les ferramentes d'internacionalització del sistema per facilitar-lo en tres idiomes: anglès, català i castellà.

També s'ha pogut treballar amb la plataforma Google App Engine i sobretot amb el *framework* Django, on s'ha assolit:

- Gestió d'usuaris.
- Gestió de les dades emmagatzemades i les dades enviades a través de la xarxa.
- Internacionalització de la pàgina en tres idiomes: anglès, català i castellà.
- Ús de l'API de Google Maps per generar un mapa i mostrar les dades des d'un arxiu KML.

A banda dels productes també s'ha treballat tot el procés d'implementació d'una aplicació amb les fases de planificació, anàlisi, disseny i implementació. Aquest fet ha permès prendre consciència del procés de forma global i trobar-se amb les dificultats pròpies d'un projecte real.

Aquest procés ha permès també aplicar de forma conjunta els coneixements adquirits a diferents àrees de la carrera com enginyeria del programari, comunicacions sense fils, sistemes distribuïts, compiladors, gestió de projectes, etc. podent conjugar-los per assolir l'objectiu del projecte.

Per últim, a nivell personal, el projecte ha suposat l'aprenentatge i presa de contacte amb el desenvolupament d'aplicacions per a l'entorn iOS. Considere que amb la implementació de l'aplicació amb els seus components he pogut analitzar i comprendre el funcionament de l'entorn. D'aquesta forma m'ha suposat obtindre una base per poder desenvolupar per aquesta plataforma i continuar l'aprenentatge.

7. Referències

7.1 Objective-C i Cocoa Touch

- John Ray, Sean Johnson - iPhone Application Development (Desarrollo de aplicaciones para iPhone - Ed. Anaya Multimedia).
- iOS Developer Library (http://developer.apple.com/library/ios/navigation/).

7.1.1 Codi

Entre els recursos que posa a disposició dels programadors Apple hi ha petites aplicacions enfocades a una tasca en concret que s'han consultat per aprendre a realitzar diferents accions (com pot ser la localització o el dibuix de la ruta als mapes).

- Breadcrumb (http://developer.apple.com/library/ios/#samplecode/Breadcrumb/Introduction/Intro.html#//apple_ref/doc/uid/DTS40010048)
- LocateMe (http://developer.apple.com/library/ios/#samplecode/LocateMe/Introduction/Intro.html#//apple_ref/doc/uid/DTS40007801-Intro-DontLinkElementID_2)
- TableViewSuite (http://developer.apple.com/library/ios/#samplecode/TableViewSuite/Introduction/Intro.html)
- Route-me (https://github.com/route-me/route-me). Aquest codi no és d'Apple.

7.1.2 Resolució de problemes

A l'hora de resoldre els problemes la font de consulta i resolució d ha sigut principalment la pàgina web StackOverflow (http://www.stackoverflow.com).

7.2 Django

- Adrian Holovaty, Jacob Kaplan-Moss - The Definitive Guide to Django Web Development Done Right (Apress) - (La guía definitiva de Django - Anaya Multimedia).

7.3 Google Maps

- Referència d'API Versió 3 de Google Maps (http://code.google.com/intl/es/apis/maps/documentation/javascript/reference.html).