5690: Desenvolupament d'una aplicació Android per gestionar rutes de Bicing

Memòria del Projecte Fi de Carrera d'Enginyeria en Informàtica realitzat per
Daniel Salmerón Amselem
i dirigit per
Cristian Stefan Tanas.
Bellaterra, 15 de Setembre del 2014

El sotasignat, Cristian Stefan Tanas, professor de l'Escola d'Enginyeria de la UAB,

CERTIFICA:

Que el treball a què correspon aquesta memòria ha estat realitzat sota la seva direcció per en Daniel Salmerón Amselem

I per a que consti firma la present.

Signat: Cristian Stefan Tanas
Bellaterra, 15 de Setembre del 2014

Agraïments

Als meus pares per haver fet tants sacrificis en donar-me una bona educació
i a la meva dona per tot el suport rebut en aquest projecte.

Índex

Índex de figures

Índex de taules

Capítol 1

Introducció

L'ús de la bicicleta com a mitja de transport alternatiu per la ciutat s'està convertint en una pràctica cada vegada més habitual entre els habitants de grans ciutats, entre elles Barcelona. Darrerament s'ha vist una clara i important aposta per fer de les nostres grans ciutats un entorn sostenible i compromeses amb el medi ambient. Això fa que serveis com el Bicing [1], que ofereix als ciutadans la possibilitat de llogar bicicletes pels seus desplaçaments urbans, siguin cada vegada més populars. Els números avalen aquesta visió i és que el servei de Bicing ja compta amb més de 97.000 abonats, segons dades de la pròpia empresa gestora.

D'altra banda, gràcies al grau de penetració dels anomenats telèfons intel·ligents o *smartphones* en el mercat mòbil i en la vida dels ciutadans, cada vegada podem veure més aplicacions que assisteixen a l'usuari en l'ús del servei de Bicing (cerca de estacions o bicicletes, navegació per la ciutat, etc.).

Per tant, podem afirmar amb total convicció que serveis com el Bicing són iniciatives dignes de tenir en consideració i participar en la millora d'aquest servei és un dels objectius d'aquest projecte.

1.1 Motivació

En un món com el nostre on la població global no deixa de créixer i on les grans ciutats cada cop acollen a més habitants, els responsables de mobilitat d'aquests nuclis urbans es troben cada cop més sovint amb el repte de combatre la congestió viària i la contaminació urbana d'una forma eficient i sostenible. Un dels grans impactes que es poden apreciar notablement com a resultat d'aquest problema és la desacceleració del creixement econòmic, com es demostra en una gran varietat de fonts privades [2] i públiques [3].

Ciutats com Barcelona amb gairebé 8 milions [3] de desplaçaments diaris, lluiten des de fa anys en contra del vehicle privat amb iniciatives com la millora de la xarxa d'autobusos, l'ampliació de la xarxa de metro i la introducció del sistema de bicicletes públiques Bicing. L'objectiu és el de transformar de mica en mica la ciutat en un entorn on el vehicle privat no és necessari i retornar l'espai ocupat per aquests als ciutadans. Aquesta tendència però no és exclusiva de Barcelona. Les ciutats d'Amsterdam, París, Londres o Berlín són un gran exemple de que el problema de la mobilitat sostenible està en el punt de mira dels centres metropolitans més grans d'Europa.

Tot i que la solució més acceptada globalment a dia d'avui és la de proporcionar un transport públic accessible i de qualitat al ciutadà, encara hi ha molts dubtes sobre quina combinació de mitjans de transport adoptar. Mentre que algunes ciutats han apostat clarament per l'autobús i d'altres pel metro, s'ha observat que els ciutadans que segueixen fent servir el transport privat ho fan principalment per la comoditat que els proporciona.

És per aquest motiu que es pot observar com arreu del món [4] s'han realitzat grans inversions en els sistemes de bicicletes públiques, un sistema que sembla proporcionar l'equilibri perfecte entre ecologia, sostenibilitat i comoditat.

1.2 Objectius

L'objectiu d'aquest projecte és desenvolupar una aplicació mòbil que promogui l'ús de la bicicleta com a mitjà de transport. Específicament, es tracta de proporcionar informació acurada i de fàcil accés sobre l'estat dels serveis de bicicletes públiques, com per exemple el Bicing a la ciutat de Barcelona. Per assolir aquest objectiu, s'utilitzarà la informació que les empreses que gestionen els serveis públics d'aquest mitjà de transport posen a disposició dels seus ciutadans.

Entre les necessitats més destacables entre els usuaris, en trobem principalment dues de molt importants. D'una banda, la necessitat de conèixer la localització de les estacions més properes amb la finalitat de saber on recollir o deixar una bicicleta, i d'una altra la necessitat de conèixer com anar d'un punt a un altre de la ciutat d'una forma òptima. Tot i que com veurem més endavant, existeixen en l'actualitat un bon nombre d'aplicacions que proporcionen aquesta informació, cap d'elles ha dedicat suficients esforços en fer-ho d'una forma simple i intuïtiva per l'usuari.

Tot i que en una etapa inicial es marca com a objectiu proporcionar el servei només a la ciutat de Barcelona, es preveu estendre la funcionalitat de l'aplicació a altres ciutats d'arreu del món, gràcies al disseny modular i extensible del sistema. Aquesta funcionalitat permetrà als usuaris fer servir la mateixa aplicació amb la que es troben familiaritzats des de qualsevol indret del món.

Finalment, es proposa dissenyar i desenvolupar un mòdul de visualització de les dades que permeti dur a terme un anàlisi exhaustiu de la informació generada a partir de l'ús de l'aplicació. L'objectiu és comprendre millor l'utilització d'aquest tipus de servei per part dels usuaris i determinar patrons que permetin millorar, en un futur, el servei proporcionat.

1.3 Estructura de la memòria

L'ordre recomanat de lectura d'aquest document és el suggerit i permetrà al lector seguir en detall cadascuna de les etapes en les que el projecte s'ha dividit.

El capítol 2 introdueix l'estat de l'art en l'àmbit de les aplicacions mòbils pel transport públic en bicicleta. S'analitzen en detall les aplicacions més rellevants i es mostren algunes de les mancances més destacables. Això ens permetrà veure quins són els punts més importants i on es dedicaran la major part dels esforços en aquest projecte.

El capítol 3 presenta l'anàlisi previ al desenvolupament del projecte i els requeriments necessaris per assolir els objectius proposats. També mostra un estudi de viabilitat tant a nivell tècnic com operatiu, econòmic i legal conjuntament amb una planificació temporal de les tasques a realitzar.

En el capítol 4 es pot observar en detall el disseny global de la solució proposada, introduint de forma detallada els diferents components que integra l'aplicació. També analitzem en profunditat algunes de les decisions més rellevants.

El capítol 5 mostra quina ha estat la implementació del disseny presentat en el capítol anterior. Aquest capítol s'endinsa en els detalls d'implementació més rellevants per transformar el disseny presentat en un sistema funcional.

En el capítol 6 es presenten els resultats obtinguts, així com el seguit de proves que s'han portat a terme per assegurar que la implementació realitzada compleix amb els objectius i requisits establerts.

Finalment, el capítol 7 exposa les conclusions a les que s'han arribat després de finalitzar el projecte i es proposen algunes possibles línies de treball futur per estendre la feina realitzada.

Capítol 2

Estat de l'art

A dia d'avui existeix un ampli ventall d'aplicacions que proporcionen dades sobre l'estat dels diferents serveis de bicicletes. La gran majoria d'aquestes aplicacions estan focalitzades en una sola ciutat, mentre que d'altres han adoptat una visió més global i suporten una gran varietat de ciutats d'arreu del món. En aquest capítol analitzarem en detall algunes de les propostes més populars, centrant-nos en el servei de Bicing que s'ofereix a la ciutat de Barcelona.

2.1 Bicing: una forma alternativa de moure's per Barcelona

El Bicing és el servei de bicicletes públiques de la ciutat de Barcelona inaugurat l'any 2007. El seu funcionament es basa en un sistema d'abonament anual, que es pot realitzar a través de la seva web [1], i permet als seus usuaris fer-ne un ús gairebé il·limitat durant un any sencer. Les restriccions que existeixen es basen en fomentar la compartició de les bicicletes entre els usuaris, i penalitzar a aquells que en facin un ús indegut. Per aquest motiu,

tan sols els intervals continus de 30 minuts estan inclosos en l'abonament i per sobre d'aquests, existeixen penalitzacions econòmiques. Un cop l'usuari rep la targeta de soci, es pot dirigir a qualsevol de les estacions que es troben repartides al voltant de la ciutat, apropar-la al lector de targetes i agafar la bicicleta que el sistema ha desbloquejat. Un cop l'usuari arriba a la seva destinació, tan sols ha de col·locar la bicicleta en un dels llocs lliures de l'estació i el sistema automàticament reconeixerà a l'usuari que l'ha retornat. Algunes de les dades més rellevants dels servei les podem trobar a la Taula 2.1 que es troba a continuació.

Servei Bicing	
Nombre d'abonats	97.437
Nombre d'usos mensuals	1.214.947
Nombre d'usos acumulats any 2014	8.085.833
Temps mig de viatge	14.23 minuts
Bicicletes	
Nombre de bicicletes	6.000
Mitja d'usos per bicicleta i dia	6,5
Km. recorreguts per bicicleta al mes	544,05
Estacions	
Nombre d'estacions	420
Percentatge d'estacions sense avaria	96,84
Mitja de bicicletes reparades per dia	613
Internet	
Nombre de visites mensuals al web	57.011
Tarifes	
Abonament anual	47,16€
30 primers minuts amb abonament	Gratuït
Des del minut 30 fins a les 2 hores	0,74€/30 min.
Penalització per haver excedit les 2 hores	4,49€

Taula 2.1: Informació del sistema de Bicing - Juliol 2014 [5]

Gràcies als *smartphones* es pot donar un valor afegit als usuaris, ja que des de qualsevol lloc poden consultar la disponibilitat de les estacions sense necessitat de desplaçar-se.

2.2 Aplicacions existents pel servei de Bicing

A continuació es resumeixen les aplicacions més populars que ofereixen informació relacionada amb el servei de Bicing de Barcelona.

2.2.1 Aplicació oficial de Bicing

L'empresa gestora del servei públic de Bicing ha desenvolupat una aplicació mòbil disponible per les plataformes iOS, Android, Windows Phone i Blackberry. Entre les funcionalitats ofertes per aquesta aplicació podem trobar, per exemple, cerca de rutes, estat del servei o disponibilitat de cada estació.

Tot i estar present en una gran varietat de plataformes i disposar d'un clar avantatge respecte a la resta, l'aplicació presenta evidents mancances relacionades amb l'experiència d'usuari (UX -de l'anglès *User Experience*). Aquestes mancances es deriven del fet que s'intenta que l'aplicació sigui multi-plataforma alhora que ofereix la mateixa interfície d'usuari (UI - de l'anglès *User Interface*) i UX per totes les plataformes, obviant els mecanismes visuals propis de cada una d'elles. En la Figura 2.1 es pot observar un exemple de la UI per Android i iOS.

Entre els comentaris més populars escrits pels usuaris també destaca una UI molt poc intuïtiva, no orientada a proporcionar la informació més rellevant en el moment idoni. En el millor dels casos, l'aplicació hauria de ser lo suficientment intel·ligent com per saber quina acció es vol realitzar, però com això és un problema de per sí força complex, una aposta poc arriscada hauria

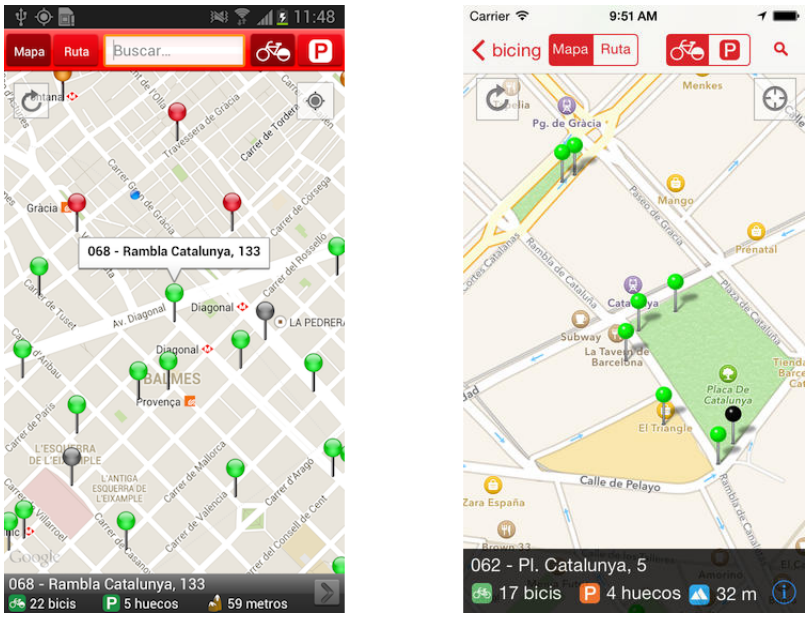

Figura 2.1: Aplicació oficial de Bicing per Android i iOS

sigut mostrar directament les estacions més properes i la seva disponibilitat.

2.2.2 Vicing

La segona aplicació més popular per a les plataformes iOS i Android és Vicing. Tot i disposar de millors valoracions dels usuaris en comparació amb l'aplicació oficial, manca de característiques tan importants com la cerca de rutes entre un punt i un altre de la ciutat. En canvi, inclou algunes altres menys útils com un llistat de totes les estacions del servei.

A nivell d'interfície, és probablement l'aplicació que disposa d'una UI més complexa i difícil d'entendre, en gran part degut a l'ús d'icones confuses que no revelen la informació que s'està buscant però també degut a l'absència de patrons de disseny propis de la plataforma Android (veure Figura 2.2). Malgrat aquestes deficiències, l'aplicació també disposa d'algunes funcionalitats força interessants com ara la visualització de carrils de bicicleta sobre

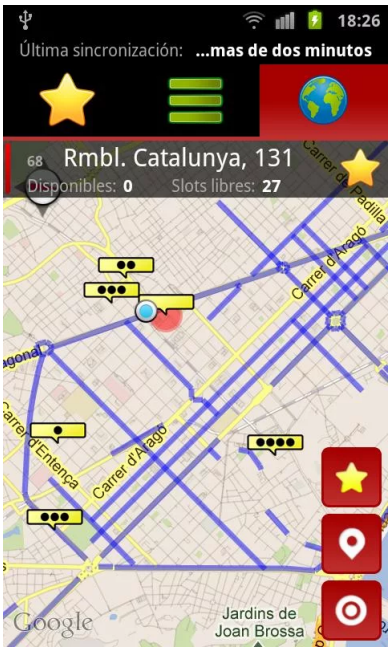

Figura 2.2: Aplicació Vicing per Android

el mapa i la possibilitat de guardar estacions a una llista de favorits.

2.2.3 Bicicleting

Aquesta aplicació és la més recent de totes les analitzades i només es troba disponible per la plataforma Android. En l'actualitat[1], disposa de la millor valoració dels usuaris, malgrat que només 100 persones l'han descarregat.

En general es tracta d'una aplicació molt atractiva a nivell visual (veure Figura 2.3) i disposa d'algunes de les funcionalitats més útils pel dia a dia. Entre elles, trobem la possibilitat de cercar rutes entre estacions i de guardar les estacions preferides. Tot i que són obvis els esforços que s'han dedicat en la creació d'una aplicació com aquesta, una característica que sembla indicar un cert grau de complexitat en l'ús d'aquesta és la incorporació d'una guia

[1]Setembre del 2014

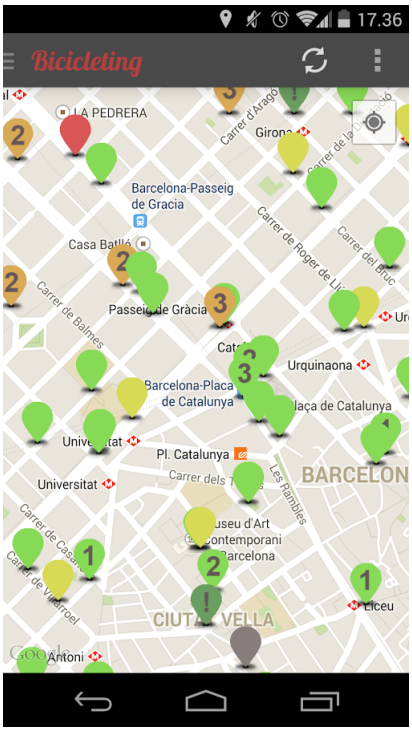

Figura 2.3: Bicicleting per Android

visual que explica com fer-la servir.

A nivell de rendiment i fiabilitat, s'ha pogut observar, durant les proves realitzades, que la aplicació deixa de funcionar i es tanca sobtadament de forma constant. Malauradament degut a la manca d'opinions a la Play Store, no s'ha pogut contrastar aquesta experiència amb la d'altres usuaris.

2.2.4 CityBikes

CityBikes és probablement la iniciativa que més punts té en comú amb els objectius d'aquest projecte. D'una banda, disposa d'una API pública que proporciona informació del servei de bicicletes de més de 170 ciutats del món, no només per Bicing. A més, disposa d'aplicacions oficials per iOS i Android de codi obert.

Tot i que al principi es valorava la possibilitat de contribuir en el projecte amb l'objectiu de millorar-lo, tant la API com el grau d'adaptabilitat de les aplicacions mòbils a les funcionalitats que es volen implementar van fer que es desestimes aquesta possibilitat. Un altre motiu que es va tenir en compte és que l'última actualització de l'aplicació d'Android data del 26 d'Octubre de 2013 i ens va donar indicis de que el projecte podia estar abandonat.

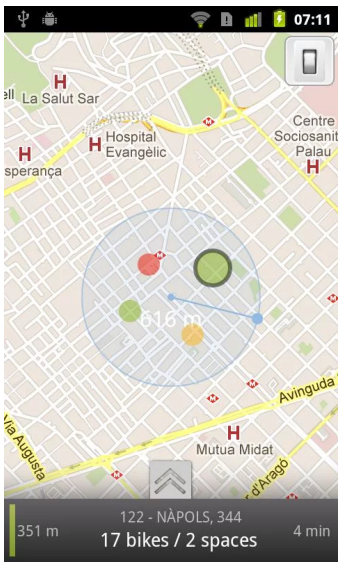

Figura 2.4: Aplicació CityBikes per Android

Una de les funcionalitats més interessants que proporciona, és la possibilitat de filtrar els resultats del mapa depenent de si estem buscant una estació on recollir o retornar una bicicleta. Això ens indica que l'autor ha pensat en els diferents casos d'ús i de l'existència d'usuaris amb necessitats diferents. Una altra funcionalitat a destacar és la selecció del servei a utilitzar depenent de la localització actual de l'usuari. Aquesta característica permet al usuari viatjar arreu del món i fer servir la mateixa aplicació amb la que es troba familiaritzat.

No obstant això, l'aplicació proporciona dades rarament fiables i són presentades a l'usuari d'una manera força complexa a través d'una interfície poc intuïtiva (veure Figura 2.4). A més a més, els problemes de rendiment que

es van observar amb l'aplicació anterior també hi són presents en CityBikes.

2.3 Conclusions

En aquest capítol hem revisat algunes de les aplicacions més populars pel servei de Bicing de Barcelona veient les seves virtuts així com els seus defectes. Un tret característic que es pot observar en la majoria de les aplicacions analitzades, a excepció de l'aplicació Bicicleting, és el baix nivell de SNR[6] (*Signal to Noise Ratio* per les seves sigles en anglès) que ens permet mesurar el que s'està mirant (*senyal*) en relació amb l'entorn en el que es troba (*soroll*). Si la quantitat de soroll és molt elevada, la senyal es perd. En el nostre cas, la visualització d'una quantitat molt elevada d'informació que l'usuari realment no necessita, pot donar pas a una pèssima experiència d'usuari. D'altra banda, la majoria d'aplicacions sembla tenir com a objectiu cobrir el cent per cent dels casos d'ús, comprometent la UX en cada un d'ells.

Per tant, tal i com veurem al llarg dels propers capítols, l'objectiu d'aquest projecte es desenvolupar una aplicació que reculli el millor de les solucions analitzades alhora que sigui ràpida, atractiva i sobretot funcional, orientada a proporcionar la informació més rellevant en el moment necessari.

Capítol 3

Anàlisi de requeriments

En aquest capítol veurem tots els requeriments, tant funcionals com no funcionals, necessaris per dur a terme aquest projecte. D'altra banda, presentarem un breu estudi de viabilitat i una planificació temporal que ens permetran entendre si es factible la realització d'aquest projecte.

3.1 Perfil d'usuaris

Abans de començar a exposar els requeriments funcionals i no funcionals del sistema que es vol desenvolupar, es necessari entendre el perfil dels interessats. En el nostre cas, es tracta de persones que viuen a la ciutat de Barcelona i que es mouen per la ciutat fent servir el servei públic de bicicletes Bicing. Com podem veure amb més detall a la Taula 3.1, ens trobem amb un nombre d'usuaris distribuïts de forma equitativa entre els dos sexes, majoritàriament joves i en conseqüència, molt familiaritzats amb les noves tecnologies, segons dades de la Viquipedia [7].

Degut a que la solució que es proposa desenvolupar és per un servei ja existent i degut a la competència ja existent en el sector, no ha sigut ne-

Usuaris	
Nombre d'abonats	97.437
Percentatge d'homes	57,7
Percentatge de dones	44,3
Mitjana d'edat	34,4 anys

Taula 3.1: Informació del usuaris de Bicing

cessari la creació d'entrevistes o enquestes per recollir els requisits a tenir de l'aplicació. En comptes, s'ha utilitzat la informació proporcionada per la pròpia empresa gestora del servei així com els comentaris i opinions dels usuaris de les aplicacions existents, accessibles a les plataformes de distribució d'aquestes.

Tot i que l'oferta existent podria indicar una saturació del mercat o donar indicis de la manca de demanda per una nova aplicació, els comentaris dels usuaris ens indiquen tot el contrari. Per aquest motiu, creiem que el desenvolupament d'una aplicació basada en l'experiència obtinguda de la resta d'alternatives i en els suggeriments del usuaris, ens pot ajudar a desenvolupar una alternativa més popular. A continuació, es presenten de forma detallada els requeriments tant funcionals com no funcionals de l'aplicació, els quals ens permetran més endavant acotar el conjunt de funcionalitats recomanades pel disseny d'un sistema software com el que es vol implementar.

3.2 Requeriments funcionals

Com hem vist al capítol anterior, per una banda, algunes aplicacions disposen de moltes funcionalitats i/o opcions, convertint en un autèntic repte el fet de proporcionar una UI/UX simple i funcional. Per altra banda, mantenir una UI/UX massa simple ens pot fer caure en l'altre extrem on els usuaris reclamen funcionalitats bàsiques que no es troben presents. A més a més

s'ha de tenir en compte que la majoria d'usuaris fan un ús puntual i quan ho necessiten d'aquestes aplicacions. És per aquest motiu que s'ha decidit limitar el nombre de característiques a implementar, enforcant-se en millorar aquelles que són realment importants pels usuaris i al mateix temps conservar els beneficis que proporciona disposar d'una UI/UX simple i intuïtiva.

Cerca de les estacions més properes: La funcionalitat que segons s'ha observat és la més rellevant pels usuaris, és poder trobar les estacions més properes a on es troben. Aquest fet es dona ens els casos on els usuaris o bé volen recollir o bé retornar una bicicleta a l'estació més propera del punt on estan. Per aquest motiu i degut a la importància que aquesta funcionalitat té, la major part del disseny del sistema proposat gira al voltant de proporcionar una experiència òptima en la cerca de les estacions més properes. En conseqüència, es descarta mostrar totes les estacions de la ciutat tant en un mapa com en un llistat.

Cerca de rutes: La següent característica més popular és la possibilitat de buscar rutes des d'un lloc a un altre. Com hem vist al capítol anterior, no totes les aplicacions proporcionen aquesta funcionalitat i les que ho fan no sembla que hagin aconseguit satisfer als seus usuaris. D'una banda trobem que l'intent de proporcionar una UX satisfactòria passa a través de formularis complexos i poc intuïtius, mentre que d'altra banda trobem implementacions que es queden a mitges i només donen rutes entre estacions. Per tant, la nova proposat haurà d'integrar un mecanisme que proporcioni suggeriments a mesura que l'usuari introdueix l'adreça de destí, alhora que mostra la ruta des del punt origen fins al punt de destí mostrant quines són les estacions de Bicing més properes a cada un d'aquests punts.

Disponibilitat de les estacions: De poc serviria disposar d'una aplicació d'aquesta mena, sense la possibilitat de veure la disponibilitat de les estacions mostrades. En altres aplicacions és possible filtrar estacions depenent de si es vol deixar o agafar una bicicleta, mostrant d'aquesta manera només la

dada que interessa a l'usuari, el nombre de bicicletes o el nombre de llocs lliures. Malauradament, les diverses implementacions que s'han observat són en general força confuses i per aquest motiu, s'ha decidit eliminar aquesta funcionalitat i simplement mostrar ambdós dades.

Indicacions en un mapa: Donat que la majoria de gent està acostumada a interpretar indicacions sobre un mapa i les distàncies a realitzar són generalment curtes, s'ha decidit prescindir de descripcions escrites com les que es poden trobar en aplicacions com Google Maps. En comptes, es mostrarà una línia sobre el mapa amb el trajecte més curt fins al destí.

Durada del trajecte: Visualitzar una ruta sobre un mapa és molt convenient, però quan l'usuari no està familiaritzat amb l'entorn, és difícil percebre les distàncies entre els objectes. Per aquesta raó, s'ha pensat en afegir un indicador que permeti saber a l'usuari quant lluny es troba d'una estació. Malgrat que existeixen dues possibilitats clares, s'ha decidit fer servir unitats de temps en comptes de distància, doncs és un indicador universal que no requereix de conversió entre sistemes de mesura i proporciona el mateix grau d'informació.

Global: Com s'ha comentat en els objectius d'aquest projecte, es preveu expandir el servei a altres ciutats del món. Per aconseguir aquesta fita, és necessari dissenyar un sistema que o bé requereixi de la interacció de l'usuari per seleccionar un servei determinat, o bé d'un mecanisme que permeti fer aquesta selecció de forma automàtica (e.g. mitjançant l'ús de la seva localització). Mantenint l'estratègia de simplificar la UI/UX, s'ha decidit implementar la darrera opció. Com veurem en els propers capítols, és una de les funcionalitats més interessants i complexes d'aquesta aplicació.

3.3 Requeriments no funcionals

El nivell d'impacte que els anomenats *smartphones* o dispositius intel·ligents han tingut en la les activitats quotidianes de les persones ha provocat que tant empreses grans, petites com programadors independents, hagin buscant l'oportunitat d'oferir solucions adaptades a aquests terminals. Tot i així, l'impuls de les empreses per incorporar les noves tecnologies juntament amb el nivell baix de maduresa dels desenvolupadors en plataformes mòbils ha provocat que moltes de les solucions proporcionades per aquests dispositius no hagin estat adaptades de forma òptima a les restriccions que existeixen en aquests terminals. Tot i que aquesta tendència ha començat a desaparèixer amb el temps, és encara força comú trobar moltes aplicacions amb grans deficiències a nivell d'usabilitat.

Tenint en compte aquesta situació, s'ha pres especial atenció en el desenvolupament d'una aplicació adaptada completament a aquest tipus de dispositius. A continuació es mostren alguns dels punts més rellevants.

Simple: Donat l'espai de pantalla limitat dels *smartphones* és molt important determinar les funcionalitats que es volen incloure per evitar sobrecarregar la interfície d'usuari amb multituds d'opcions. Per aquesta raó i donada la familiaritat de les persones amb els mapes, s'ha decidit donar una especial rellevància al mapa de la ciutat i la localització de les estacions.

Ràpida: Un dels requisits més importants, sinó el més important de tots és la rapidesa amb que l'aplicació respon entre les diferents interaccions de l'usuari. Com bé Jakob Nielsen (expert en usabilitat) comenta en l'article "Website Response Times"[8], una interfície d'usuari ràpida és molt millor que una d'atractiva. L'estudi demostra la importància de proporcionar una resposta immediata a l'usuari, entre 0.1 i 1 segon, amb l'objectiu de proporcionar-li una sensació de control i alhora mantenir la seva concentració.

Intuïtiva: Relacionada estretament amb la simplicitat, és important fer

servir elements amb els que l'usuari ja es troba familiaritzat. Per aquest motiu, i en el cas de les aplicacions per Android, és molt recomanat utilitzar la guia d'estil [9] proposada per Google.

3.4 Estudi de viabilitat

A continuació es tracta en detall la viabilitat del projecte des d'un punt de vista tècnic, operatiu, econòmic i legal.

3.4.1 Viabilitat tècnica

En aquest projecte s'ha treballat sobre un entorn de desenvolupament web i mòbil utilitzant els llenguatges de programació Go i Java. Degut a que s'han desenvolupat dos components totalment diferents, s'ha fet servir Go per la part de servidor i Java, conjuntament amb les eines de desenvolupament d'aplicacions mòbils per la plataforma Android, per la part del client. El motiu pel qual s'han escollit aquestes dues tecnologies és degut a l'experiència prèvia de la que es disposa i que no requereix de cap aprenentatge previ.

En l'apartat de maquinari, s'ha disposat d'un ordinador portàtil MacBook Pro de 13 polzades amb processador *Intel Core i5* de 2.6GHz i 16GB de memòria RAM i d'un Mac Mini amb processador *Intel Core i7* de 2.3GHz i 4GB de memòria RAM. Pel desenvolupament de l'aplicació Android, s'ha fet servir un dispositiu *Nexus 5* de 32GB i el simulador proporcionat per l'entorn de treball Android Studio.

En quant a programari, s'ha fet servir VIM 7.4 com editor de text, Android Studio 0.8.6 pel desenvolupament de l'aplicació d'Android i PostgreSQL 9.3 com a base de dades relacional.

3.4.2 Viabilitat operativa

Com hem esmentat amb anterioritat, es desenvoluparan dos components: un sistema servidor que donarà servei a clients mòbils amb el sistema operatiu Android a través d'una API. Tant Go com la plataforma Android ofereixen una gran varietat d'eines per desenvolupar noves aplicacions, des de les més simples fins a les més complexes. A més a més, l'existència d'aplicacions similars ens ha permès entendre que el desenvolupament i posterior obtenció de dades no suposa cap risc en l'execució del projecte, ni en el cas del Bicing ni en el cas d'altres serveis similars oferts arreu del món.

D'altra banda, les altes prestacions dels terminals disponibles en el mercat dels *smartphones* podem eliminar també qualsevol limitació en quant a la instal·lació, execució i manteniment d'una aplicació d'aquesta mena.

3.4.3 Viabilitat econòmica

En l'àmbit econòmic, es disposa tant del *hardware* com de tot el programari necessari i per tant, no ha sigut necessari realitzar cap tipus de despesa. Malgrat que a l'inici el servidor s'allotjarà en el Mac Mini, més endavant serà imprescindible llogar un servidor virtual. A més, per publicar l'aplicació als canals oficials de distribució de Google és necessari adquirir una llicència de desenvolupador Android. Aquests costos, a càrrec de l'estudiant, es resumeixen a continuació.

- Lloguer del servidor: 14€/mes
- Llicència de desenvolupament Android: $25

En el cas de que els costos de producció sobrepassin els descrits anteriorment, existeix la possibilitat de buscar diverses formes de finançament

externes. Les dues opcions més simples i viables avui en dia són, o bé afegir anuncis a l'aplicació o bé vendre-la a través dels canals de distribució *online* d'Android.

3.4.4 Viabilitat legal

La font de dades sobre el servei de Bicing de Barcelona disposa d'una llicència que permet l'explotació del conjunt de dades per qualsevol finalitat, incloent-hi una finalitat comercial i la creació d'obres derivades. L'única condició requerida és la de citar l'autoria de l'Ajuntament de Barcelona.

La informació recollida sobre els usuaris es limitarà a la posició geogràfica en la que es troben. Aquesta informació no serà emmagatzemada enlloc i només s'utilitzarà amb el propòsit de trobar les estacions més properes. Donada la naturalesa d'aquestes dades, la LOPD [10] no suposa cap impediment per la realització del projecte.

En quant a les eines proposades pel desenvolupament d'aquest projecte, tot el programari que s'utilitzarà és programari lliure, les llicències del qual permeten la seva utilització sense cap restricció legal.

3.5 Planificació temporal

El desenvolupament d'aquest projecte es durà a terme seguint un model en cascada [11] i estarà dividit en 7 fases agrupades en una part tècnica i una altra part documental.

La part tècnica té una durada estimada de 300 hores repartides entre les següents fases i tasques:

- Anàlisi de requeriments: 20 hores.
- Disseny i implementació: 240 hores.
- Implementació del sistema basat en jocs: 26 hores.
- Llançament: 2 hores.
- Proves: 12 hores.

La part documental té una durada estimada de 40 hores i es compon només de dues tasques: redacció de la memòria i preparació de la presentació.

En conclusió, la durada estimada de realització del projecte és de 340 hores. La data d'inici del projecte és 10 d'Octubre del 2013 i s'estima que la data final d'entrega sigui el 15 de Juny del 2014, el que suposa una càrrega de treball de 10 hores/setmana. A la Figura 3.1 es pot veure en detall el diagrama de Gantt del projecte.

3.6 Conclusions

En aquest capítol hem vist amb detall quins són els requeriments del projecte, tant els funcionals com els no funcionals, i hem pogut observar que el projecte és viable des dels punts de vista tècnic, operatiu, econòmic i legal. Finalment, s'estima que es necessitaran 340 hores per completar el projecte seguint un model de desenvolupament software basat en cascada.

Figura 3.1: Planificació temporal

Capítol 4

Biqingo: sistema integrat per moure's per la ciutat amb bicicleta

En aquest capítol veurem amb detall el disseny d'una arquitectura client/servidor per la solució proposada, que ens garantirà que tots els requeriments descrits anteriorment són satisfets. Tot i que l'objectiu principal és desenvolupar una aplicació mòbil per la plataforma Android, gran part dels esforços i per tant del disseny s'han dedicat a la part del servidor tal i com veurem al llarg d'aquest capítol.

4.1 Arquitectura general

Com ja hem esmentat, el desenvolupament d'aquest projecte compren el desenvolupament d'una aplicació servidor encarregada de gestionar tota la informació relacionada amb les estacions de Bicing i la cerca de rutes entre diferents estacions, així com de proveir aquesta informació a un client

mòbil, la funció del qual és presentar aquesta informació d'una forma òptima i rellevant per l'usuari. D'aquesta forma podem delegar les operacions més costoses (e.g. l'algoritme de cerca de rutes) al servidor alhora que centrem els esforços per construir una UI/UX òptima en el client mòbil.

En l'actualitat existeixen un bon nombre d'arquitectures de disseny de software que ofereixen una forma senzilla i escalable per portar a bon port un projecte d'aquestes característiques. En el cas de Biqingo hem decidit utilitzar una arquitectura proposada per Robert Martin i anomenada *Clean Architecture* [12]. Aquesta proposta no introdueix conceptes revolucionaris, però recull i combina alguns dels patrons de disseny més coneguts en un sòl disseny. Com es pot observar a la Figura 4.1, Martin proposa un disseny per capes on cada capa dóna suport a un tipus diferent de funcionalitat. El concepte més destacat d'aquesta arquitectura, és l'anomenat *Dependency Rule*, el qual ens diu que la comunicació entre capes ha de ser en una sola direcció, des de fora cap a dins i mai a l'inversa. D'aquesta manera, aconseguim evitar que modificacions en elements a les capes superiors tinguin cap impacte en el disseny o implementació de les capes inferiors. És important destacar que les capes més externes són les responsables de definir els detalls més específics sobre el sistema, mentre que les capes més internes representen conceptes més abstractes sobre la lògica de negoci.

4.2 Disseny de l'aplicació servidor

Seguin el model d'arquitectura presentat a la secció anterior, el disseny del servidor es divideix en 4 capes: entitats, casos d'ús, adaptadors d'interfície i API que podem observar a la Figura 4.2 (tot i que disposades en forma de diagrama de mòduls).

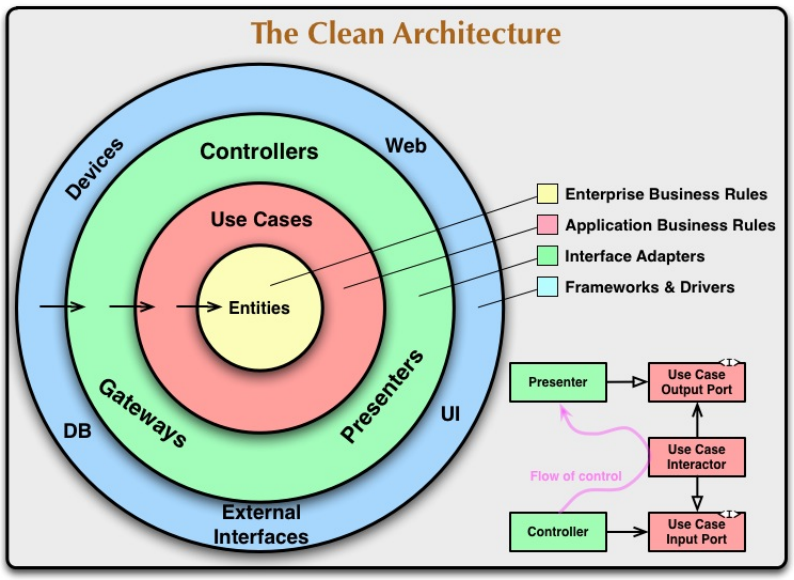

Figura 4.1: Clean Architecture per Robert Martin

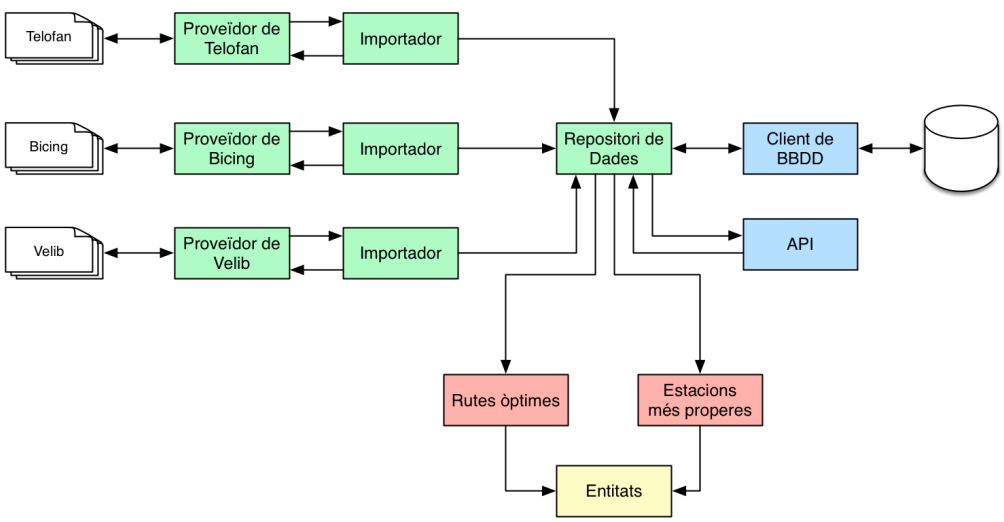

Figura 4.2: Disseny modular del servidor

4.2.1 Entitats

El mòdul d'entitats és l'encarregat de definir la lògica de negoci mitjançant la implementació d'objectes i les operacions sobre aquests. La separació

en un mòdul d'aquest tipus ens permet definir les regles que descriuen el nostre sistema, sense tenir en compte cap factor extern com per exemple, l'emmagatzemament d'aquests objectes. D'aquesta forma, les regles que es definiran en aquest mòdul seran vàlides tant si emmagatzemem els objectes en una base de dades com en un fitxer de text. Els objectes que conformen aquest mòdul són:

- **Estació:** Representa un estació de bicicletes genèrica que conté una adreça i la disponibilitat.
- **Adreça:** Conté informació sobre el carrer, número, ciutat, codi postal, país i posició geogràfica.
- **Posició:** Coordenades geogràfiques de latitud i longitud.
- **Disponibilitat:** Representa el nombre de bicicletes i espais lliures disponibles.
- **Ruta:** Composta per una parella representada per la posició geogràfica de l'origen i l'estació més propera a aquesta, i una altra parella per la destinació.

4.2.2 Casos d'ús

El mòdul de casos d'ús es l'encarregat de proporcionar els mètodes que implementen els requeriments que hem vist al capítol anterior, tant de cerca de les estacions més properes com de les rutes òptimes entre dos punts. Aquest mòdul es troba en una capa per sobre de l'anterior i ens permet evitar que canvis en els requeriments tinguin cap impacte sobre les regles de negoci. És a dir, donada una estació de la capa d'entitats i les regles que defineixen quan aquesta està lliure o no, un canvi en la manera amb que es calculen les rutes òptimes no ha de suposar cap canvi a la capa d'entitats.

Estacions més properes

La obtenció de les estacions més properes a un punt és un problema específic del conegut com *Nearest Neighbor Search*. En aquest tipus de problemes, s'utilitza una funció que indica el grau de semblança entre dos objectes. En el nostre cas, donat que es tracta d'objectes que representen posicions geogràfiques, aquesta funció estarà definida a partir de la distància mètrica entre dos punts.

En l'actualitat existeixen un bon nombre de solucions que permeten obtenir aquesta informació, però com veurem en el proper capítol d'implementació amb més detall, s'ha decidit utilitzar una estructura de dades *R-Tree* i un algorisme de cerca que permet trobar les k estacions més properes. Com el seu nom indica, el R-Tree és una estructura de dades de tipus arbre balancejat (veure Figura 4.3) i està especialment dissenyat per proporcionar mètodes d'accés espacial, és a dir, proporciona mètodes d'indexació d'objectes multidimensionals. Va ser proposat per Gutman [13] l'any 1988 degut a que una de les estructures de dades més populars de l'època, el *B-Tree*, només podia realitzar operacions de igualtat i de rang sobre dades que es podien ordenar d'alguna forma.

És important destacar que el R-Tree tan sols proporciona un mecanisme d'aproximació eficient, però un cop obtingut el node on es troba l'estació o estacions en qüestió, és necessari realitzar una cerca lineal per obtenir el resultat òptim. La manera amb la que això s'aconsegueix és mitjançant l'agrupació d'estacions propers, gràcies a la funció de semblança descrita anteriorment, en els anomenats *Minimum Bounding Rectangles* o *MBR*. Com ja hem esmentat anteriorment, els nodes fulla es troben tots a la mateixa alçada i són els únics que contenen les estacions. La resta de nodes, tenen la funció d'agrupar en aquests rectangles totes les estacions que contenen. És doncs, força intuïtiu comprendre que a mesura que recorrem l'arbre des de l'arrel fins a un node fulla, s'acota el rang de cerca. L'operació que ens permet

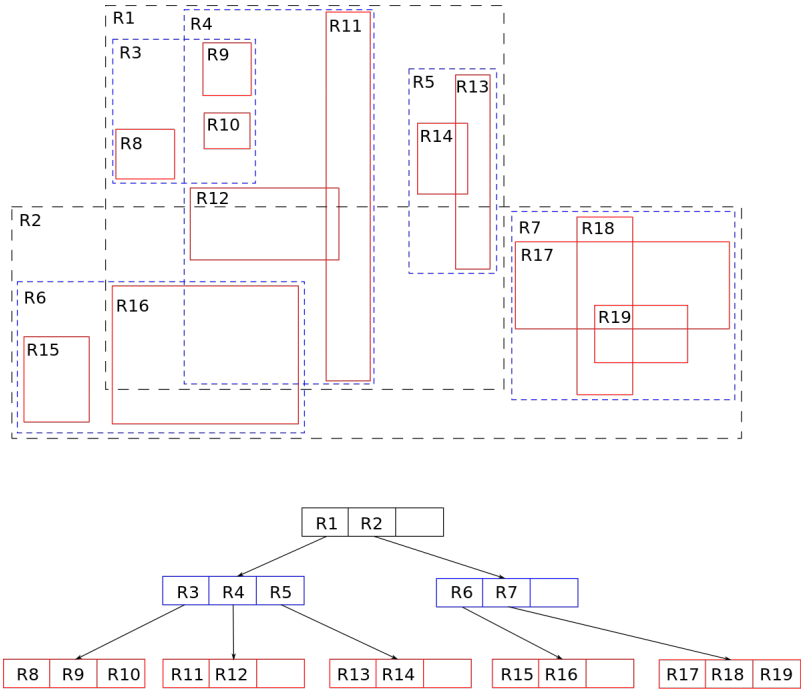

Figura 4.3: Representació d'un R-Tree i els MBR

decidir el node o nodes que s'han d'escollir a cada pas és molt simple, doncs només s'ha de realitzar la intersecció entre els MBR de cada node i l'element de consulta fins trobar aquells que sí s'intersequen. Aquesta operació es realitzarà repetidament a cada nivell fins arribar als nodes fulla on finalment es farà la cerca definitiva.

El major repte en el manteniment d'aquestes estructures de dades, és aconseguir mantenir la seva eficiència durant repetits cicles d'inserció i eliminació d'objectes. Per aconseguir-ho, es fan servir heurístiques que intenten minimitzar el solapament entre MBR per evitar la cerca en diferents subarbres. A la Figura 4.4 es pot observar l'impacte que diferents heurístiques tenen sobre el R-Tree.

Un cop es disposa d'una estructura de dades d'aquest tipus, la cerca dels k objectes més propers es pot realitzar aplicant l'algorisme presentat per N. Roussopoulos et. al. [14].

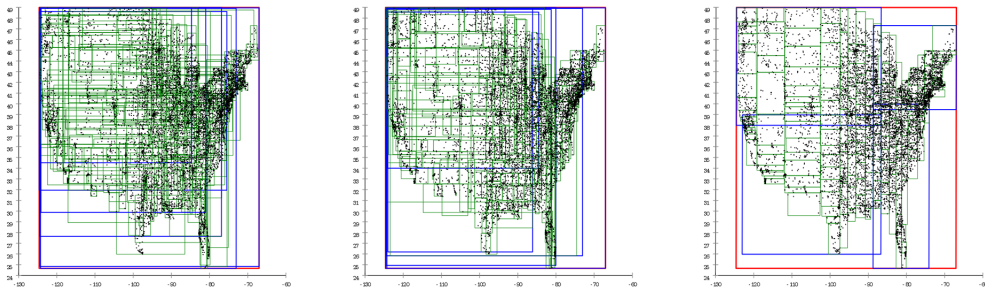

Figura 4.4: Impacte de les diferents heurístiques del R-Tree

Rutes òptimes

La obtenció de rutes òptimes entre dos punts utilitzant el servei de Bicing, és un problema que es pot solucionar fàcilment un cop es disposa del mecanisme vist al punt anterior. Donats dos punts, origen i destí, podem trobar les rutes òptimes entre els dos, mitjançant una permutació entre les k estacions més properes a l'origen i les k més properes al destí. Un cop obtinguda aquesta llista, s'ordenen en ordre creixent per distància del recorregut total i finalment es seleccionen les r primeres.

En el cas del nostre sistema, s'ha decidit escollir un valor de k i r igual a 3 doncs creiem que és un nombre d'opcions més que suficient per l'usuari.

4.2.3 Adaptadors d'interfície

Aquest mòdul és l'encarregat de proporcionar la interfície entre la lògica de negoci definida en els dos mòduls anteriors i els components més tècnics de la implementació del sistema. A continuació es mostren els tres components que formen aquest mòdul i veurem en detall la manera en que interactuen.

Importador

Donat que el sistema que es vol desenvolupar depèn completament de dades proporcionades pels serveis de bicicletes públiques, és necessari dissenyar un mecanisme per importar aquestes dades. Tenint en compte que es vol implementar una arquitectura que permeti estendre les funcionalitats a altres ciutats i serveis del món, és molt important entendre l'existència de restriccions específiques a cada ciutat. Mentre que alguns serveis proporcionen una API que es pot accedir remotament, altres simplement publiquen un fitxer estàtic en format XML o JSON. Alhora s'ha pogut observar que alguns serveis imposen restriccions en el nombre de consultes que es poden realitzar en un determinat interval de temps, mentre que d'altres no.

Amb l'objectiu de proporcionar un servei òptim per tots i cadascun d'aquests serveis, s'ha dissenyat el component d'importació de dades amb les següents dos característiques en ment: importació independent i aïllada per cada servei i intervals d'importació específics a les limitacions de cada servei.

Per aconseguir aquesta fita, l'importador es defineix com una interfície que abstrau els detall d'implementació per definir i mantenir un contracte entre aquest mateix i els seus clients. D'aquesta manera, el component d'importació no necessita conèixer des d'on s'importen les dades ni com es guarden. L'únic que el importador coneix és que donat un proveïdor d'un servei i un repositori de dades, ha de demanar al proveïdor de dades la informació sobre l'estat de les estacions i enviar aquestes dades al repositori de dades pel seu emmagatzemament. A la Figura 4.5 podem veure a grans trets el diagrama de funcionament d'aquest component.

Com veurem en el proper capítol, les decisions de disseny que s'han pres en aquest component afecten en la implementació de l'estructura R-Tree que hem vist prèviament.

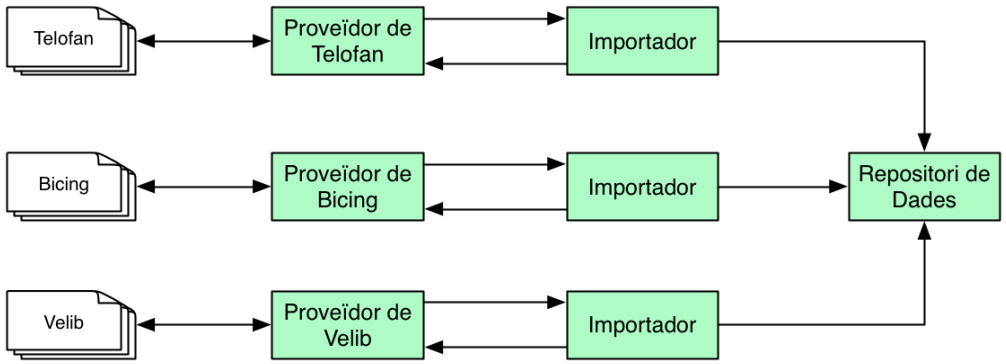

Figura 4.5: Arquitectura de l'importador

Repositoris de dades

El repositori de dades és el responsable d'emmagatzemar la informació i proporcionar una interfície d'accés per la resta de components. En el disseny del nostre sistema s'han implementat dos repositoris de dades diferents. El primer proporciona accés a una base de dades relacional i el segon a una estructura de dades en memòria. Donat que ambdós solucions implementen una mateixa interfície, podem intercanviar-les de forma transparent. Els detalls d'aquesta interfície es mostraran en el proper capítol.

Proveïdors

La funció del proveïdor és la d'obtenir les dades del servei de bicicletes d'una ciutat i transformar-les a un llenguatge comú definit pel mòdul d'entitats. Per tant, hi haurà tants proveïdors com serveis de bicicletes que es vulguin integrar amb l'aplicació. Tot i així, aquest component defineix una interfície que és independent de la ciutat del proveïdor i que facilita la comunicació amb el component d'importació.

Cal destacar que la funcionalitat d'obtenció de dades del proveïdor és altament reutilitzable, donat que la majoria de serveis publiquen les dades

mitjançat un servei web accessible a través dels protocols HTTP o FTP, mentre que la transformació i normalització de les dades obtingudes és una funció específica a cada ciutat.

4.2.4 API

Finalment, la comunicació entre el client mòbil i el servidor es realitzarà a través d'una REST API definida pel servidor. Les dues consultes que els clients mòbils poden realitzar al servidor són les que es descriuen a continuació.

1. **GET /stations**: Retorna les 3 estacions més properes a un punt donat.
    - Paràmetres:
        - latitude: Latitud del punt on es volen trobar les estacions.
        - longitude: Longitud del punt on es volen trobar les estacions.

1. **GET /routes**: Retorna les 3 millors rutes entre dos punts donats.
    - Paràmetres:
        - origin_latitude: Latitud del punt d'origen.
        - origin_longitude: Longitud del punt d'origen.
        - destination_latitude: Latitud del punt de destí.
        - destination_longitude: Longitud del punt de destí.

Com es pot observar, la API no implementa cap mecanisme d'autenticació, tot i que es preveu incorporar el protocol *OAuth 2.0* en un futur.

4.3 Disseny del client mòbil

El disseny del client és molt més simple que el disseny del servidor. No es tracta d'una coincidència sinó d'una decisió basada en el concepte de *Thin Clients* [15]. Aquesta idea proposa reduir al màxim la lògica a la part de client i donar la major part de la responsabilitat al servidor a diferència dels *Fat Clients* [16] que combinen les dues funcions en un únic dispositiu. A la Taula 4.1 podem veure una breu comparativa entre les dues tendències, clarament oposades, cada una amb les seves avantatges i inconvenients.

	Thin Client	Fat Client	Valor òptim
Manteniment	↓	↑	↓
Seguretat	↑	↓	↑
Escalabilitat	↓	↑	↑
Risc	↑	↓	↓

Taula 4.1: Comparativa entre clients *Thin* i *Fat*

En el cas del nostre sistema, el risc de que el servidor deixi de funcionar o els possibles problemes d'escalabilitat són qüestions que no tenen un gran impacte en el rendiment de l'aplicació en aquesta fase del desenvolupament i per tant, es poden enfrontar més endavant.

4.3.1 Descripció general

Com hem comentat anteriorment, el disseny del client està enfocat en oferir una experiència d'usuari òptima i per tant, la senzillesa de l'aplicació juntament amb l'organització de la informació són punts crítics.

Quan l'usuari obre l'aplicació, immediatament es mostra un mapa amb les 3 estacions més properes a la seva posició actual. L'estació més propera es senyala mitjançant un marcador de major grandària i amb un color més

accentuat. Sobre el mapa, es mostra mitjançant una línia de color negre el recorregut des d'on es troba l'usuari fins l'estació. A la part inferior de la pantalla, es presenten les dades més rellevants sobre aquesta estació. D'una banda s'indiquen el nombre de bicicletes i llocs lliures disponibles i d'una altra, el temps necessari per arribar a l'estació a peu (veure imatge de l'esquerra en la Figura 4.6. Si l'usuari vol conèixer les dades de les altres dues estacions, pot prémer sobre qualsevol dels marcadors o lliscar el dit d'un costat a l'altre sobre la barra inferior.

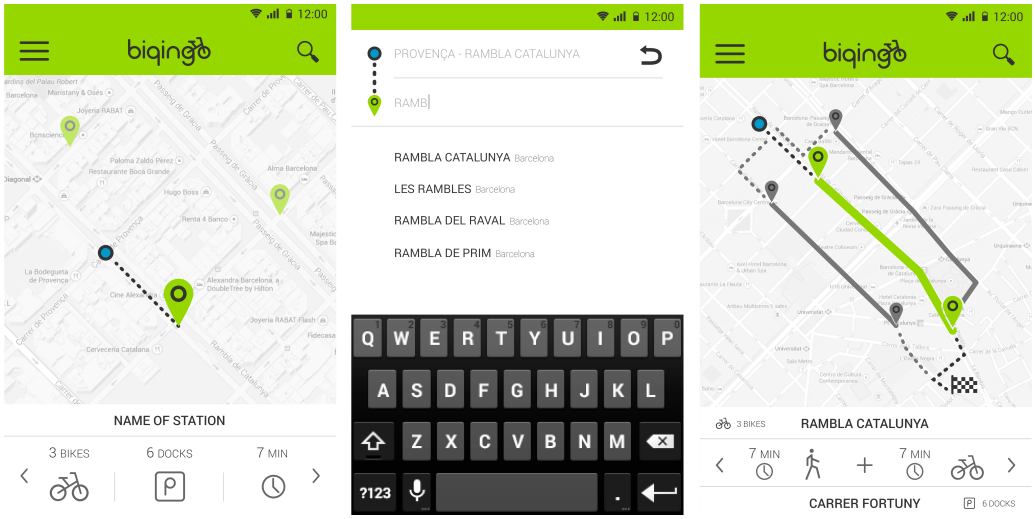

Figura 4.6: D'esquerra a dreta: estacions més properes, cerca i visualització de rutes.

En el cas que l'usuari estigui interessat en trobar la ruta entre dos punts de la ciutat, el botó situat a la part superior dreta de la pantalla ens portarà a una nova vista (veure imatge central en la Figura 4.6). A mesura que l'usuari va introduint les dades del seu destí, els resultats són actualitzats mitjançant els suggeriments proporcionats per Google. Un cop introduïda tota la informació, l'usuari pot prémer sobre el botó de cerca a a part superior dreta per tornar a la vista anterior, aquest cop podent visualitzar en el mapa les rutes disponibles.

Una vegada sabem a quina estació l'usuari vol agafar una bicicleta i en

quina deixar-la, només mostrem la informació necessària per cadascuna de les estacions. En quant a la visualització del temps de recorregut, es mostra tant el temps total de recorregut a peu com el temps de recorregut en bicicleta (veure imatge de la dreta en la Figura 4.6). Per veure les dades de les rutes alternatives, de nou l'usuari pot seleccionar qualsevol dels marcadors o lliscar el dit sobre la barra inferior.

4.3.2 Patró Model-Vista-Controlador (MVC)

En quant a l'arquitectura del client mòbil, seguirem el patró Model-Vista-Controlador (MVC), patró molt popular en l'àmbit d'aplicacions tant mòbils com web. Aquesta arquitectura està especialment enfocada en la implementació d'UIs i permet separar la representació interna de la lògica de negoci de la forma amb la que aquesta és presentada a l'usuari. A la Figura 4.7 es mostra un diagrama de la relació entre aquests tres components.

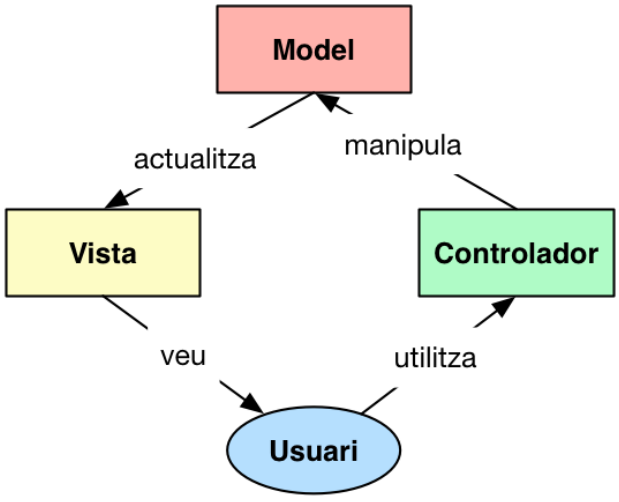

Figura 4.7: Arquitectura MVC

Models

Donat que els models són els responsables de representar les dades que es volen mostrar a l'usuari, totes les entitats que s'han vist a la part de servidor també es trobaran codificades a la part de client. D'aquesta manera, es podrà disposar d'objectes com estacions, adreces, rutes, etcètera. Malgrat que existeixen opcions per evitar la duplicació de codi entre els dos sistemes, generalment mitjançant l'ús de llenguatges de programació com C o C++, s'ha decidit evitar-ho degut al grau de complexitat afegit.

Vistes

Les vistes es corresponen amb la UI de l'aplicació i són les encarregades de definir com es presentarà la informació dels models a l'usuari. En el nostre cas, l'estructura i estil de les vistes estaran especificades en fitxers XML, permetent una completa separació amb els models.

Controladors

Els controladors tenen la responsabilitat de respondre a les interaccions amb l'usuari i modificar o actualitzar el model o la vista en conseqüència. En el nostre cas, utilitzarem un mecanisme anomenat *UI Data Binding* [17] per dur a terme la comunicació entre els models i les vistes a través de notificacions. Les dues versions més comunes d'aquest patró són conegudes com *one-way data binding* i *two-way data binding*. Mentre que la primera només suporta l'actualització de la vista respecte a canvis en els models, la segona permet al mateix temps actualitzar els models respecte a canvis realitzats sobre les vistes. A la Figura 4.8 es mostra amb més detall la diferència entre les dos.

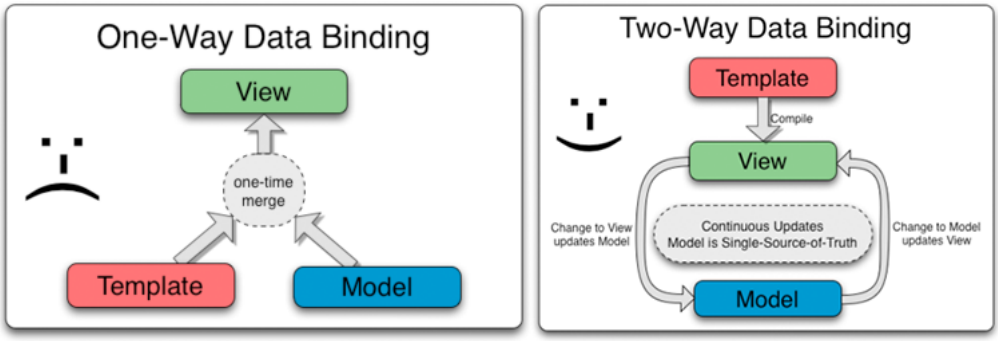

Figura 4.8: Gràfics sobre *Data Binding* de la documentació d'AngularJS

4.4 Conclusions

En aquest capítol hem introduït el disseny global de l'aplicació Biqingo. Hem pogut veure com el disseny segueix una arquitectura client/servidor on el servidor s'encarrega de proporcionar als clients informació relacionada amb les estacions de Bicing o les rutes entre dos punts de la ciutat, mentre que el disseny del client està centrat en la senzillesa i optimització de la informació presentada seguint un patró MVC.

Capítol 5

Detalls d'implementació

En aquest capítol veurem alguns dels detalls més importants d'implementació del disseny presentat en el capítol anterior. Veurem tant la implementació de l'aplicació servidor com la implementació del client mòbil, així com l'esquema de comunicació entre el client i el servidor.

5.1 Implementació del servidor

El servidor ha estat desenvolupat íntegrament utilitzant el llenguatge de programació Go [18], un llenguatge desenvolupat principalment a Google i especialment enfocat al disseny d'aplicacions concurrents.

5.1.1 Diagrama de classes d'entitats

Al diagrama de la Figura 5.1 podem observar el diagrama de classes de les entitats descrites en el capítol anterior, així com les relacions entre aquestes.

Tot i que els noms són força entenedors, a continuació expliquem amb més

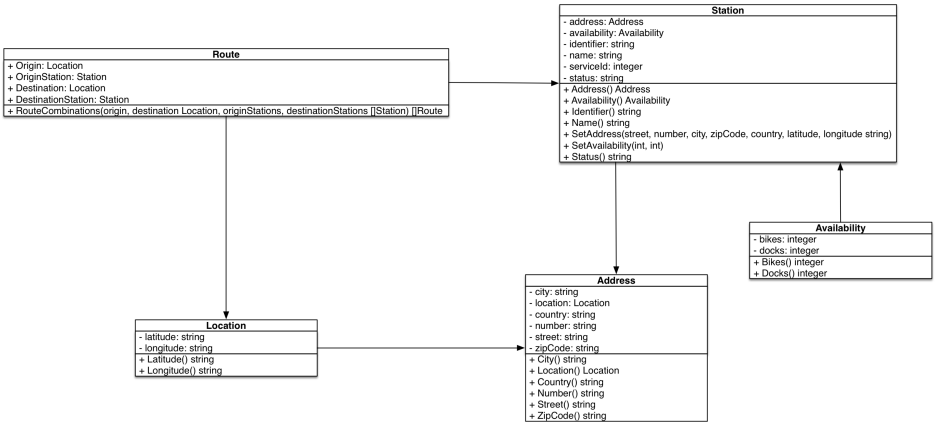

Figura 5.1: Diagrama de classes de les entitats

detall les responsabilitats de cada una d'aquestes classes. Les dues classes més importants són la classe **Station** i la classe **Route**. La primera ens permet representar una estació qualsevol i normalitzar les dades de les estacions de diferents serveis. La segona, ens permet representar la ruta entre dos punts, fent servir una estació de bicicletes tant a l'origen com al destí.

La classe **Location** es fa servir per indicar la posició geogràfica d'un objecte, representada per la latitud i la longitud en format String, i s'utilitza tant a la classe **Route** que acabem de veure, com a la classe **Address**. Aquesta última s'utilitza per emmagatzemar informació sobre la localització d'una estació, en un format més còmode i llegible per les persones i es mostra a la part del client.

Finalment disposem d'una classe **Availability** que indica el nombre de bicicletes i llocs disponibles hi ha a cada estació.

5.1.2 Intervals d'importació de dades

Com hem comentat en el capítol anterior, el component d'importació de dades de diferents serveis de bicicletes hauria de mostrar uns intervals d'im-

portació d'acord amb les limitacions de cada servei (e.g. nombre de consultes restringit). Per donar solució a aquest problema s'han avaluat dues solucions.

En una primera instància s'ha avaluat utilitzar un sistema similar al *Cron* [19], una eina present en la majoria de sistemes Unix-like i que permet la gestió i execució de tasques que s'han d'executar de forma periòdica. La implementació d'aquest mecanisme passa per dividir el servidor en dos parts diferenciades: gestió i publicació de la API i execució periòdica del component importador. Aquesta segona part proporciona un executable mitjançant el qual l'eina Cron inicia el procés d'importació de dades dels serveis especificats.

D'altra banda s'ha considerat aprofitar les eines que el propi llenguatge Go facilita alhora de desenvolupar aplicacions concurrents. Aquesta solució consisteix en fer ús de les anomenades *goroutines*, que descrites de forma general són funcions que s'executen de forma paral·lela amb altres *goroutines* en el mateix espai de memòria. El seu funcionament és semblant al dels *threads*, però amb la diferència que el sistema operatiu no té coneixement de la seva existència. El *runtime* de Go és l'encarregat de realitzar la planificació mitjançant un planificador cooperatiu [20] i al mateix temps permet distribuir les *goroutines* en diferents *threads*. En el cas de sistemes multi-core, el desenvolupament de software amb capacitats d'executar codi en paral·lel és molt més simple que en altres llenguatges. Tot i que fins ara només hem parlat de les *goroutines*, aquestes no són més que un tipus de thread molt més lleuger, però amb els mateixos riscs que aquests comporten degut a la compartició del mateix espai de memòria. Aquest problema però, és solucionat pels *channels*, que a grans trets canvien la forma amb la que dos *goroutines* es comuniquen. Es passa d'un model on es comunica mitjançant la compartició de memòria, com és el cas dels *threads*, a un model on es comparteix memòria mitjançant la comunicació [21].

Donat el suport del llenguatge Go per aquest tipus de software concurrent,

hem decidit integrar la segona solució en la versió final de l'aplicació servidor. Aquesta solució s'adapta millor a les nostres necessitats i gràcies a l'existència d'una llibreria [22] que implementa una interfície similar a Cron, no ha sigut necessària cap tasca de desenvolupament. A més a més, una de les avantatges més importants és que ens evita generar dependències externes i simplifica la tasca de publicació de noves versions al servidor.

5.1.3 Emmagatzemament d'informació geogràfica

La gestió eficaç de la informació de tipus geogràfic (e.g. latitud i longitud) és una de les tasques més crítiques en aquest tipus d'aplicacions. És per això que s'han estudiat dues opcions pel seu emmagatzemament i posterior manipulació.

La primera solució avaluada consisteix en utilitzar una base de dades relacional, com ara *PostgreSQL* amb l'extensió *PostGIS* per la indexació de dades geogràfiques. La implementació d'aquesta solució és força trivial i no requereix de gaires esforços. Malgrat l'existència d'una solució tan simple, s'ha decidit explorar la possibilitat d'implementar una estructura de dades en memòria que permeti la indexació d'aquest tipus de dades. A la Taula 5.1 podem veure una petita comparativa entre les dues aproximacions. Creiem que aproximació basada en memòria pot suposar una millora substancial de rendiment respecte a una aproximació més tradicional amb bases de dades.

	PostgreSQL	Estructura de dades en memòria
Memòria	Persistent	Volàtil
Base de dades distribuïda	Simple	Complexe
Indexació	Simple	Complexe
Rendiment	Mitjà	Alt
Introdueix dependències	Sí	No

Taula 5.1: Comparativa entre les dues aproximacions considerades

Per implementar la segona solució hem utilitzat una llibreria [23] per Go que incorpora una implementació de l'algorisme original R-Tree proposat per Gutman, al que hem afegit petites variacions.

El problema de la implementació d'una solució d'aquesta mena sorgeix en el moment d'emmagatzemar les dades. Quan l'importador proporciona les dades de les estacions al repositori de dades, aquest ha de guardar-les d'alguna forma. Sense importar el tipus de format que s'utilitzi, aquesta operació presenta varis reptes que cal sobrepassar. En el cas de que ja hi hagi dades en el sistema, és necessari detectar aquelles estacions la informació de les quals ha estat actualitzada, ja sigui la seva disponibilitat com qualsevol altre camp. A més, s'han d'identificar aquelles estacions que s'hagin eliminat del servei o aquelles que s'hagin afegit. Donat que aquesta operació resulta molt costosa i complexa, es proposa una solució molt més simple que consisteix en eliminar totes les estacions emmagatzemades prèviament d'un servei específic, i guardar-les de nou. D'aquesta forma evitem haver de realitzar qualsevol tipus de comparacions entre objectes.

Malauradament, això comporta un problema per una implementació que es basa en un únic R-Tree pels diferents serveis, ja que les operacions d'inserció i eliminació d'objectes de forma repetida poden generar un empitjorament del seu rendiment. Això és degut a que cada cop que s'elimina o s'afegeix un objecte, l'arbre genera una estructura completament diferent basada en la reagrupació de nodes a partir dels MBR, i doncs pot causar un alt nivell de superposició entre aquests. Per evitar aquesta situació, existeixen dues possibilitats. La primera i més directa, és fer servir un algorisme basat en els R-Tree especialment orientat a dades estàtiques, és a dir, dades que no solen canviar amb freqüència i es coneixen en el moment de la inserció. Aquest algorisme, conegut com STR [24], permet generar una distribució òptima de les entrades dels nodes. La segona opció consisteix en disposar d'un arbre R-Tree per cada servei. Aquesta solució ens permet tenir varis arbres R-Tree que no es veuen afectats per les repetides operacions d'inserció i eliminació,

però introdueixen un de nou. Per tal d'obtenir les estacions més properes a un punt donat, s'ha de realitzar la cerca per cadascun dels arbres i buscar aquells que generen interseccions amb els MBR dels nodes.

Malauradament, per una implementació en memòria basada en un únic R-Tree pels diferents serveis, les operacions d'inserció i eliminació d'objectes de forma repetida, poden generar un empitjorament del seu rendiment. Això és degut a que cada cop que s'elimina o s'afegeix un objecte, l'arbre genera una estructura completament diferent basada en la reagrupació de nodes a partir dels MBR, i doncs pot causar un alt nivell de superposició entre aquests. Per evitar aquesta situació, existeixen dues possibilitats. La primera i més directa, és fer servir un algorisme basat en els R-Tree especialment orientat a dades estàtiques, és a dir, dades que no solen canviar amb freqüència i es coneixen en el moment de la inserció. Aquest algorisme, conegut com STR (*Sort Tile Recursive* per les seves sigles en anglès) [24], permet generar una distribució òptima de les entrades dels nodes. La segona opció consisteix en disposar d'un arbre R-Tree per cada servei. Aquesta solució ens permet tenir varis arbres R-Tree que no es veuen afectats per les repetides operacions d'inserció i eliminació, però introdueixen un nou problema: per obtenir les estacions més properes a un punt donat, s'ha de realitzar la cerca per cada un dels arbres i buscar aquells que generen interseccions amb els MBR dels nodes.

Tot i que s'ha realitzat una implementació experimental d'aquest mecanisme, finalment, i degut a restriccions de temps, s'ha decidit fer servir l'extensió PostGIS per PostgreSQL i deixar per més endavant una implementació en memòria basada en els R-Tree molt més completa.

5.2 Implementació del client

Tot i la existència de múltiples plataformes per desenvolupar un client mòbil, hem optat per desenvolupar aquesta primera versió del client per la plataforma Android de Google [25] pel seu caràcter open-source així com pels coneixements previs que es tenien sobre aquesta plataforma.

5.2.1 Desenvolupamet de les Activities

Les *Activities* en Android són components que proporcionen a l'usuari una finestra amb la que poden interactuar. Normalment ocupen tota la pantalla i generalment es disposa d'un *Activity* per cada tipus diferent de visualització. En el cas de la nostra aplicació, només tenim dos *Activities* diferents: la *MainActivity* que mostraria un mapa amb les estacions més properes i les rutes més òptimes i la *SearchActivity* que conté el formulari per la cerca de rutes. A la Figura 5.2 podem veure un diagrama de la relació entre aquestes dues.

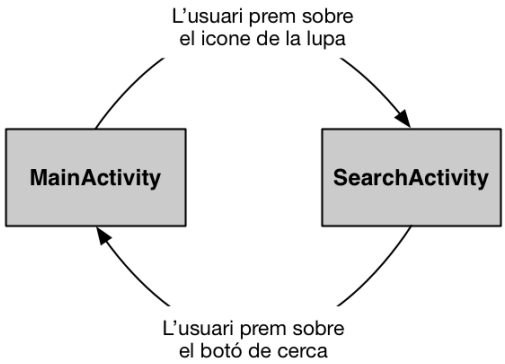

Figura 5.2: Disseny de les Activities en el client

Quan s'utilitza més d'una *Activity*, és molt important entendre correctament el seu funcionament i al mateix temps, implementar la correcta interac-

ció entre aquestes. Tot i que no entrarem en detall, Android només executa una Activity al mateix temps i la manera amb la que l'usuari es mou d'una Activity a una altra és mitjançant un tipus de pila en la que les Activities que passen a segon pla son guardades juntament amb el context en el que estaven sent executades. Cada Activity disposa d'una sèrie de *callbacks* que permeten proporcionar més control al programador sobre l'estat de la aplicació. Degut a que la nostra aplicació només disposa de dos Activities, aquesta coordinació és molt simple.

5.2.2 Carrega de dades asíncrona

El sistema operatiu Android assigna un únic procés a cada una de les aplicacions instal·lades en el dispositiu i tots els components d'una aplicació -a no ser que s'especifiqui el contrari- són executats en un *thread* que es coneix com *UI Thread*. Aquest thread és l'encarregat d'executar tots els processos de renderització d'elements d'UI (e.g. botons, vistes, etc.). La mala gestió d'aquest thread és el principal causant del baix rendiment que experimenten algunes aplicacions desenvolupades per aquesta plataforma. Això és degut a que si s'executa alguna altra operació costosa, com per exemple importar dades des d'un servidor remot, el *UI Thread* ja no pot donar resposta a les interaccions de l'usuari fins que la segona tasca hagi acabat. Això, finalment comporta un alentiment de l'aplicació i conseqüentment una insatisfacció general entre els usuaris.

Tot i així, Android proporciona un conjunt de mecanismes que permeten l'execució asíncrona de tasques i la posterior recollida de dades. Malgrat el bon nombre d'opcions, la classe *AsyncTask* de la llibreria estàndard d'Android sembla ser la més simple i popular.

Per tant, en el nostre cas utilitzarem una sèrie d'*AsyncTask* per dur a terme tasques en segon pla relacionades tant amb les consultes al servidor de

l'aplicació (e.g. calcular les rutes més òptimes entre dos punts de la ciutat) com consultes a serveis propis de Google (e.g. cerca d'adreces). Una vegada obtingudes les dades necessàries i després de la seva translació a objectes de la lògica de negoci, aquestes s'utilitzen per modificar les vistes que es mostren a l'usuari, ara sí, a través del *UI Thread*.

5.2.3 Adapters

En Android, us mecanisme molt popular que ens permet sincronitzar canvis de les dades entre els models i les vistes és l'ús del *Adapters*. Els *Adapters* proporcionen mètodes d'accés a les dades del model a la vegada que s'encarreguen de construir o actualitzar les vistes en base a les dades obtingudes.

En el nostre cas, donat que la major part de la vista es composa d'un mapa, hem implementat un Adapter, *FooterAdapter*, per crear un tipus d'interacció simple amb els continguts de l'aplicació. Aquest Adapter és responsable de sincronitzar una instància del *ViewPager* [26] amb un conjunt d'instàncies de tipus *Station* o *Route*.

5.3 Comunicació entre client i servidor

La comunicació i transferència de dades entre el client i el servidor es realitza mitjançant una JSON REST API. Aquest tipus de API són força populars degut a la simplicitat del format JSON i a l'enorme ventall de llibreries existents per satisfer els requisits de l'arquitectura REST.

En el nostre cas, el client mòbil, a través dels mecanismes de carrega asíncrona de dades que ja hem explicat, enviarà peticions HTTP al servidor d'acord amb la API vista en el capítol anterior. L'única acció suportada fins al moment és *GET*. Una vegada rebuda i processada la petició, el servidor

contestarà al client, també a través del protocol HTTP, amb un document en format JSON que el client, al seu torn, parsejarà i transformarà a entitats de la lògica de negoci.

A continuació es mostra un exemple de cada una de les crides disponibles a la API.

1. **GET /stations?latitude=41.403005&longitude=2.146658**
2. Resposta: Document en format JSON.


```
[
    {
        "address":{
            "city":"Barcelona",
            "country":"Spain",
            "location":{
                "latitude":"41.403384",
                "longitude":"2.145875"
            },
            "number":"21",
            "street":"Gleva",
            "zip_code":""
        },
        "availability":{
            "bikes":5,
            "docks":19
        },
        "identifier":"320",
        "name":"",
        "status":"OPEN"
    },
    ...
]
```


1. **GET /routes?origin_latitude=41.403005&origin_longitude=2.146658 &destination_latitude=41.385055&destination_longitude=2.137332**

2. Resposta: Document en format JSON.


```
[
    {
        "destination": {
            "latitude": "41.385055",
            "longitude": "2.137332"
        },
        "destination_station": {
            "address": {
                "city": "Barcelona",
                "country": "Spain",
                "location": {
                    "latitude": "41.385482",
                    "longitude": "2.138645"
                },
                "number": "96",
                "street": "Nicaragua",
                "zip_code": ""
            },
            "availability": {
                "bikes": 1,
                "docks": 29
            },
            "identifier": "367",
            "name": "",
            "status": "OPEN"
        },
        "origin": {
            "latitude": "41.403005",
            "longitude": "2.146658"
        },
        "origin_station": {
            "address": {
                "city": "Barcelona",
                "country": "Spain",
                "location": {
                    "latitude": "41.403384",
```

```
                "longitude": "2.145875"
            },
            "number": "21",
            "street": "Gleva",
            "zip_code": ""
        },
        "availability": {
            "bikes": 6,
            "docks": 18
        },
        "identifier": "320",
        "name": "",
        "status": "OPEN"
        }
    },
    ...
}
]
```


5.4 Conclusions

En aquest capítol hem vist els detalls d'implementació que ens han permès transformar el disseny presentat en el capítol anterior en un sistema funcional. D'una banda, hem vist com el servidor implementa les *goroutines*, pròpies del llenguatge Go, per implementar el component importador i com es guarden aquestes dades de caire geogràfic en una base de dades PostgreSQL amb l'extensió PostGIS. D'altra banda, hem vist com el client mòbil es composa de dues *Activities*, pròpies d'Android, i utilitza diversos components del framework proporcionat per Android per implementar funcionalitats com la carrega asíncrona de dades o la sincronització entre el model i les vistes.

Capítol 6

Proves i resultats

En aquest capítol veure algunes de les proves que s'han dut a terme per comprovar el correcte funcionament de l'aplicació i el grau de compliment dels objectius i requeriments, així com una breu presentació dels resultats obtinguts.

6.1 Proves funcionals

Durant tot el procés d'implementació, s'han realitzat un gran nombre de probes per garantir el correcte funcionament del sistema. Degut a que la part del servidor és l'encarregada de proporcionar la major part de la lògica, s'ha dedicat especial atenció a aquest.

S'ha comprovat que les dades de les estacions són actualitzades correctament a la base de dades, mitjançant observacions manuals.

S'han examinat els resultats obtinguts de la cerca de les estacions més properes i comparat amb les dades proporcionades pel servei de Bicing. S'ha pogut observar que les dades proporcionades per Bicing sobre les estacions

més properes i les que proporciona la nostra aplicació, varien lleugerament però gràcies a la comprovació de la distància amb Google Maps i Bing Maps, s'ha verificat la validesa dels nostres resultats en front dels de Bicing.

6.2 Proves de rendiment

Per avaluar el rendiment de la API i del funcionament global de l'aplicació, s'han realitzat proves mitjançant l'ús de *wrk* [27], una eina que ens permet fer *benchmarks* sobre el protocol HTTP, i permet fer ús de diferents *cores* del processador. Degut a la naturalesa de Go, podem executar el servidor sobre el nombre de *cores* disponibles en el nostre processador. Les proves s'han realitzat sobre el Macbook Pro de 13 polzades, que disposa d'un processador Intel Core i5 amb 2 cores i 16GB de RAM. El que això implica, es que en el millor dels casos podem respondre a dues peticions a la vegada (en paral·lel) i evidentment, acceptar moltes altres de forma concurrent. A la Taula 6.1 podem observar les diferències que s'han observat en l'execució del servidor amb diferents configuracions.

	PostgreSQL	Estructura de dades memòria
Durada	5 minuts	5 minuts
Threads	2	2
Connexions	100	100
Latència	222,01ms	63,53ms
Peticions/seg. per thread	225,10	800,03
Nombre de peticions	135.205	478.563
Peticions/seg.	450,65	1595,50
Transferència/seg.	2,60MB	9,61MB

Taula 6.1: Resultats de rendiment del servidor

Com es pot apreciar, el rendiment del servidor amb l'estructura de dades en memòria és 3.5 vegades superior i mostra el potencial que té la implemen-

tació d'una solució com aquesta.

Durant les proves, també s'ha analitzat l'impacte que aquesta solució té sobre els recursos del servidor (veure Figura 6.1). Com es pot observar, l'impacte en memòria és gairebé inapreciable, mentre que l'ús del processador es mou en un rang entre 45 i 70 per cent, en el cas d'ambdues implementacions presentades.

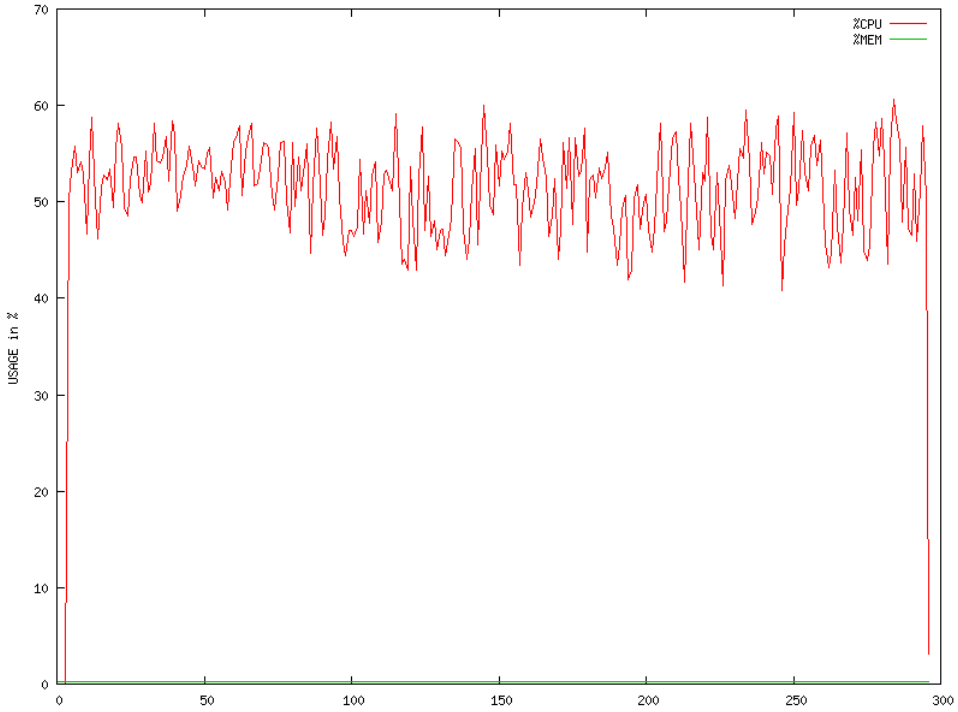

Figura 6.1: Percentatge d'ús en memòria i processador

6.3 Resultats obtinguts

L'objectiu final del projecte és desenvolupar una aplicació mòbil que proporcioni informació acurada, d'una forma molt intuïtiva a l'usuari a la vegada que proporciona una UI/UX simple i funcional. A la Figura 6.2 es pot observar la pantalla principal que Biqingo mostra als seus usuaris.

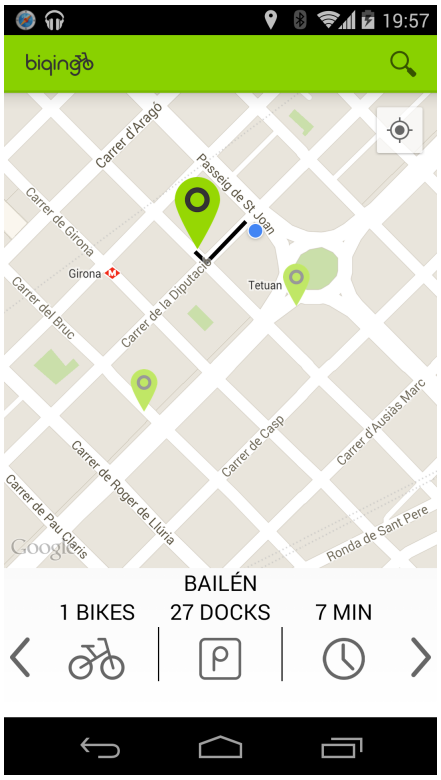

Figura 6.2: Pantalla principal amb les estacions més properes

Com es pot veure, en obrir l'aplicació, l'usuari pot veure directament les 3 estacions de Bicing més properes i la ruta (dibuixada amb negre sobre el mapa) fins a l'estació més "òptima". També es pot observar com l'aplicació mostra una instància del *ViewPager* amb informació rellevant sobre l'estació.

Una de les principals característiques de l'aplicació és que permet a l'usuari cercar les millors rutes entre dos punts qualsevol de la ciutat. Aquesta funcionalitat la podem observar a la Figura 6.3.

Com podem observar, en primer lloc se li mostra a l'usuari dos camps de text per introduir les adreces d'origen i destí amb la funció de suggeriments de Google. Una vegada introduïda aquesta informació, la ruta més òptima es mostra dibuixada a sobre del mapa. Igual que abans, una instància del *ViewPager* ens permet visualitzar informació rellevant sobre la ruta que estem

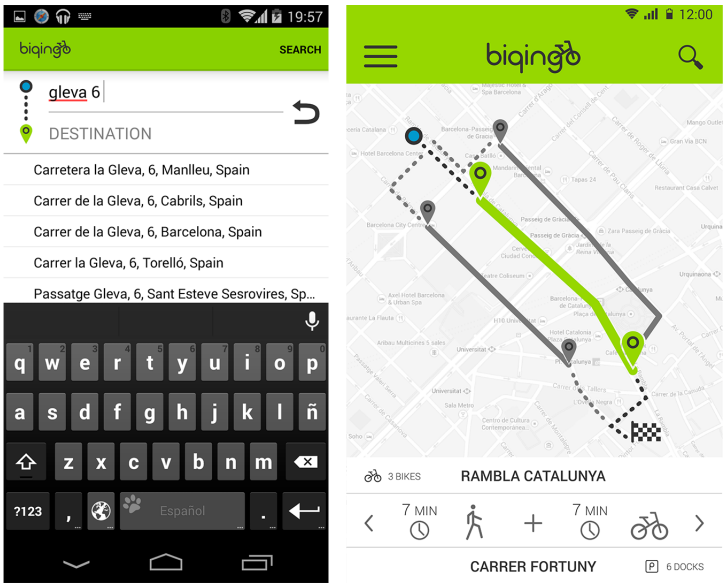

Figura 6.3: Cerca i visualitzacio de rutes òptimes

a punt de seguir. Lliscant el dit a l'esquerra o dreta sobre aquesta informació podem visualitzar les dades de les rutes alternatives.

Capítol 7

Conclusions

En aquest capítol veurem un recull dels objectius acomplerts, un resum de la feina realitzada per acomplir aquests objectius, la planificació temporal resultant i algunes línies de treball futur.

7.1 Objectius acomplerts

Si recordem, el principal objectiu d'aquest projecte és el de desenvolupar una aplicació mòbil per facilitar i incentivar l'ús dels sistemes de bicicletes públiques. Arribats al final del projecte podem dir que hem assolit amb èxit aquest objectiu.

Durant el transcurs del projecte hem volgut donar especial èmfasi en l'experiència d'usuari, la simplicitat i la rapidesa en satisfer les consultes dels usuaris de l'aplicació. És per això que una gran part dels esforços s'han dedicat a desenvolupar una aplicació servidor que integri les dades proporcionades pels proveïdors del servei de bicicletes públiques i assisteixi a l'aplicació mòbil en la cerca d'estacions i rutes entre dos punts qualsevol de la ciutat. L'aplicació servidor proporciona una API als seus clients i la comunicació té lloc a

través del protocol HTTP.

Al principi del projecte també ens marcàvem com a objectiu desenvolupar un mòdul de visualització de les dades generades a partir de l'ús de l'aplicació, amb la finalitat de comprendre millor l'utilització d'aquest tipus de servei. No obstant, els canvis no previstos en la planificació temporal del projecte com el volum de feina requerit per desenvolupar tots els components del sistema van fer que no es pogués arribar a complir amb aquest objectiu, passant aquest a ser una línea de treball futur.

Malgrat tot, estem segurs que la solució desenvolupada al llarg d'aquest projecte és superior tant a nivell tècnic com a nivell d'usabilitat respecte a l'oferta existent a dia d'avui[1].

7.2 Planificació temporal

A l'inici del projecte es va presentar una planificació temporal amb totes les tasques que s'havien de realitzar en el projecte. Com podem observar a la Figura 7.1, tot i que a les primeres etapes es va aconseguir mantenir aquesta planificació, algunes de les tasques, especialment la d'implementació, es van veure afectades per detalls que no s'havien previst amb anterioritat. Entre els imprevistos més destacables, el començament d'una nova feina que no estava prevista, i la implementació d'una estructura de dades específica pel tractament de dades geogràfiques.

Malgrat que la data d'entrega s'ha allargat en gairebé 3 mesos, de Juny a Setembre de 2014, és possible afirmar a dia d'avui que ha sigut la decisió correcta. No només s'han pogut completar gairebé tots els objectius proposats al principi sinó que s'han pogut realitzar els esforços necessaris per incrementar el valor aportat en relació a les expectatives inicials.

[1] Setembre del 2014

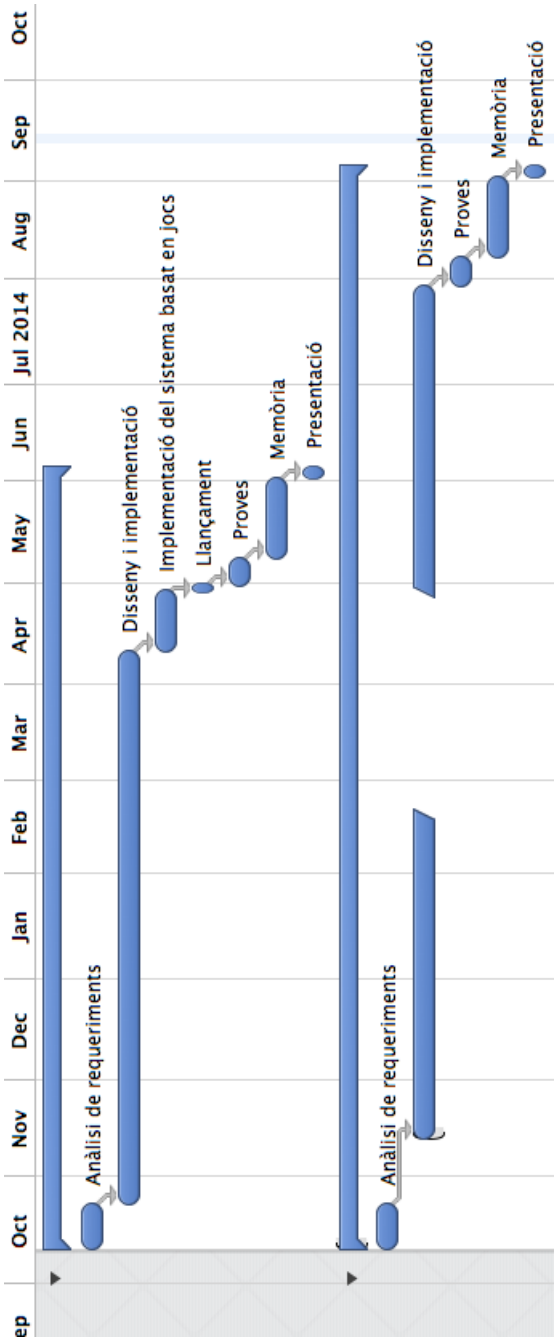

Figura 7.1: Comparació entre la planificació temporal inicial i final del projecte.

7.3 Treball futur

La feina realitzada en aquest projecte ha servit per crear el punt de partida d'un conjunt de serveis, que en un futur poden millorar de forma notable l'ús dels sistemes públics de bicicletes.

L'estratègia a llarg termini és la de proporcionar als usuaris una aplicació que facin servir a diari, de la mateixa manera que ocorre amb aplicacions com *Waze*, i emmagatzemar la posició geogràfica des d'on obren i consulten l'aplicació. Aquesta informació és d'un gran valor, doncs permet conèixer els llocs des d'on els usuaris volen recollir i retornar les bicicletes obrint un ampli ventall de noves possibilitats.

La implementació d'un mòdul de tractament i visualització de la informació generada a partir de l'ús de l'aplicació és un punt important per poder comprendre millor com els usuaris finals fan ús del servei i determinar patrons que permetin millorar, en un futur, el servei proporcionat.

D'una banda, trobem l'opció de millorar la distribució d'estacions al voltant de la ciutat, doncs es pot determinar a quins punts es necessiten aquestes estacions. Aquesta solució suposaria una millora de l'actual degut a que ja no caldria realitzar prediccions basades en el nombre real d'habitants per zona, sinó que es podria basar en el nombre d'usuaris d'aquest sistema en una zona específica.

Una altra possibilitat és la de desenvolupar algoritmes que permetin fer prediccions més precises sobre la distribució de bicicletes entre les estacions. És a dir, saber en cada instant de temps quin és el nombre òptim de bicicletes i de llocs lliures a cada estació. No només això, sinó informar de la necessitat d'obrir noves estacions o tancar aquelles per sota de la mitja d'utilització. Aquest problema és conegut com *rebalance problem* [28] i és objecte d'estudi de molts equips de recerca, tant acadèmics com empresarials d'arreu del món en l'àmbit de l'aprenentatge automatitzat.

7.4 Valoració personal

En el transcurs d'aquest projecte s'ha pogut experimentar de primera mà el repte que suposa liderar un projecte d'aquestes característiques. S'han hagut de prendre decisions que no estaven planificades en un principi i comprometre tasques que sí ho estaven. S'ha observat també, la dificultat en concretar des d'un principi el conjunt de funcionalitats a implementar en un temps limitat. No obstant, el nombre d'adversitats afrontat tant en el camp tècnic com operatiu ha demostrat que l'objectiu més gran assolit en aquest projecte ha sigut el de poder entregar un document com l'actual, on la capacitat de sintetitzar un procés complexe com el d'un Projecte de Final de Carrera té una especial importància.

Bibliografia

[1] Ajuntament de Barcelona. *Bicing*. URL: https://www.bicing.cat.

[2] RACC Automóvil Club. *La congestión en los corredores de acceso a Barcelona*. URL: http://imagenes.w3.racc.es/uploads/file/1359_adjuntos_congestio_esp_versiondiciembre_ok_jzq_810fdcaf.pdf.

[3] Ajuntament de Barcelona. *Pla de Mobilitat Urbana de Barcelona 2013, 2018*. URL: http://w110.bcn.cat/Mobilitat/Continguts/PMU2013-2018_DocAprovInicial_Parte1.pdf.

[4] Wikipedia. *List of bicycle sharing systems*. URL: http://goo.gl/rveFBd.

[5] Ajuntament de Barcelona. *Informació del Sistema Bicing*. URL: https://www.bicing.cat/ca/content/informaci%C3%83%C2%B3-del-sistema.

[6] Sam Richard. *North - Design and development standards to align and guide your project*. URL: http://pointnorth.io/#signal-to-noise-ratio.

[7] Viquipèdia. *Bicing*. URL: http://ca.wikipedia.org/wiki/Bicing.

[8] Jakob Nielsen. *Website Reponse Times*. URL: http://www.nngroup.com/articles/website-response-times/.

[9] Google. *Android Design*. URL: http://developer.android.com/design/index.html.

[10] Jefatura del Estado. *Ley Orgánica 15/1999, de 13 de diciembre, de Protección de Datos de Carácter Personal*. URL: http://www.boe.es/boe/dias/1999/12/14/pdfs/A43088-43099.pdf.

[11] Sigwart C. *Software Engineering: a project oriented approach*. 1990.

[12] Robert Martin. *The Clean Architecture*. URL: http://blog.8thlight.com/uncle-bob/2012/08/13/the-clean-architecture.html.

[13] Antonin Guttman. *R-Trees: A Dynamic Index Structure for Spatial Searching*. URL: http://www-db.deis.unibo.it/courses/SI-LS/papers/Gut84.pdf.

[14] N. Roussopoulos et al. *Nearest Neighbor Queries*. URL: http://users.dcc.uchile.cl/~bebustos/cursos/2010/cc5202/papers/RKV95%20Nearest%20neighbor%20queries.pdf.

[15] Wikipedia. *Thin Client*. URL: http://en.wikipedia.org/wiki/Thin_client.

[16] Wikipedia. *Fat Client*. URL: http://en.wikipedia.org/wiki/Fat_client.

[17] Wikipedia. *UI Data Binding*. URL: http://en.wikipedia.org/wiki/UI_data_binding.

[18] Inc. Google. *The Go Programming Language*. URL: http://golang.org/.

[19] Wikipedia. *Cron*. URL: http://en.wikipedia.org/wiki/Cron.

[20] Wikipedia. *Cooperative multitasking*. URL: http://en.wikipedia.org/wiki/Computer_multitasking#Cooperative_multitasking.2Ftime-sharing.

[21] The Go Programming Language. *Share Memory By Communicating*. URL: http://blog.golang.org/share-memory-by-communicating.

[22] Robert Figueiredo. *cron*. URL: https://github.com/robfig/cron.

[23] Daniel Connelly. *rtreego*. URL: https://github.com/dhconnelly/rtreego.

[24] Alexandros Nanopoulos Yannis Manolopoulos et al. *Rtrees: Theory and Applications*.

[25] Inc. Google. *Android*. URL: http://www.android.com.

[26] Inc. Google. *ViewPager*. URL: http://developer.android.com/reference/android/support/v4/view/ViewPager.html.

[27] Will Glozer. *wrk*. URL: https://github.com/wg/wrk.

[28] G.R.Raidl M.R.Harbach P.Papazek et al. *PILOT, GRASP and VNS Approaches for the Static Balancing of Bicycle Sharing Systems*. URL: http://goo.gl/MWpDPD.

Resum

En aquest projecte es presenta el desenvolupament d'un sistema integrat per facilitar i incentivar l'ús dels serveis de bicicletes públiques. Seguint una arquitectura client/servidor, l'aplicació és capaç de proporcionar les millors rutes en bicicleta que connectin dos punts de la ciutat. A més, l'aplicació agrega informació que les empreses gestores dels serveis públics d'aquest mitjà de transport posen a disposició dels seus clients. Tot i que de moment l'aplicació es centra en el servei de Bicing de la ciutat de Barcelona, no es descarta la seva ampliació a altres ciutats del mon.

Resumen

En este proyecto se presenta el desarrollo de un sistema integrado para facilitar e incentivar el uso de los servicios de bicicletas públicas. Siguiendo una arquitectura cliente/servidor, la aplicación es capaz de proporcionar las mejores rutas en bicicleta entre dos puntos de la ciudad. Además, la aplicación agrega información que las empresas gestoras de los servicios públicos de este medio de transporte ponen a disposición de sus clientes. A pesar de que la aplicación se centra en el servicio de Bicing de la ciudad de Barcelona, no se descarta su ampliación a otras ciudades del mundo.

Abstract

In this project we are presenting the development of an integrated system that eases and incentives the use of public bicycles services. Following a client/server architecture, the application is able to provide the best bicycle routes between two points of the city. Moreover, the application aggregates information that the companies in charge of the public bicycle sharing services provide to their users. Although the application focuses on the Bicing service of the city of Barcelona, future plans might involve other cities.