1. Introducció

La motivació d'aquest projecte ha estat d'oferir a la gent una aplicació on puguin posar a vendre les seves propietats, i que altres persones necessitades de buscar una propietat tinguin la possibilitat de trobar-la sense problemes. Creant doncs un servei web que permeti la comunicació de les dues parts implicades en l'acció de la compravenda de vivendes. Aquest mercat ha estat sempre actiu degut a les necessitats de la població. I tenint en compte que en els darrers anys la venda s'ha incrementat, cal posar a l'abast de la població noves tècniques que els permetin la interacció entre venedors i compradors, per facilitar-los tant el contacte com la informació requerida.

Aquest servei web ha estat dissenyat de manera que sigui utilitzable per tota la població, posant al seu abast mètode senzills de comunicació, cerques avançades i la possibilitat d'observar la vivenda a partir d'imatges i vídeo.

S'ha plantejat el projecte de manera que resulti atractiu pels usuaris, fent servir el conegudíssim Google Maps per al posicionament dels immobles posats a la venda, així sigui molt fàcil la seva ubicació en el mapa.

Personalment, el repte que m'ofereix el projecte és el fet d'estudiar el ja mencionat Google Maps, i en definitiva, Ajax, que és la tecnologia que usa Google Maps per modificar la informació continguda dins el mapa. També m'he proposat fer servir un framework de persistència que es demana molt en el mon laboral, que és iBatis, que inclou un framework propi de iBatis DAO, i el framework SQlMap. Utilitzant les tecnologies comentades, el que pretenc és sortir de la universitat amb un valor afegit de cara al mon laboral, ja que el fet de aprendre i entendre el funcionament des de zero és una gran experiència ja que més endavant, potser no hi haurà ningú per assessorar-me quan tingui algun dubte.

2. Objectius

L'objectiu del projecte és el de donar una eina a totes aquelles persones que vulguin vendre o comprar una vivenda o immoble, de manera fàcil i còmoda. Podem diferenciar els possibles usuaris que tindrà l'aplicació en l'acció que voldran realitzar, comprar, o vendre.

Un usuari que tingui la necessitat de comprar una propietat, no li caldrà un registre previ per tal de poder realitzar cerques o altres consultes. Per altre banda, un usuari de l'aplicació que vulgui posar a vendre una propietat, sí que li caldrà registrar-se prèviament per poder estructurar la venda de l'immoble i facilitar al comprador les dades i informacions necessàries.

Un usuari que vulgui comprar una vivenda, no serà necessari el seu registre en l'aplicació; podrà realitzar una cerca general, o una cerca acotada dels immobles que estiguin a la venda. Un cop feta aquesta cerca, podrà veure l'immoble detallat, amb les seves característiques, com les fotos i el vídeo del immoble, la seva posició en el Google Maps, les dades de contacte del venedor, i els detalls descrits del immoble.

L'usuari que vulgui vendre una propietat, primer de tot haurà d'omplir un formulari per tal de que es registri en l'aplicació. Un cop registrat, podrà anunciar la propietat a vendre, omplint el formulari de registre d'anunci, que en ell és on es localitzarà la seva posició en el mapa. Un cop registrat l'anunci, podrà afegir els detalls de les fotos i el vídeo. També podrà definir la visibilitat del anunci (en el cas que ja hagi venut la propietat, podrà fer que deixi de ser visible per als usuaris que realitzin cerques). A part de tot això, també podrà realitzar totes aquelles accions de un usuari sense registrar anomenades anteriorment.

3. Estudi de mercat

Avui en dia, a internet hi ha moltes aplicacions web relacionades amb la compra venda d'immobles o propietats, algunes d'elles gratuïtes, d'altres de pagament. Per veure com podíem innovar en aquest sector ja tant treballat, hem fet un estudi comparatiu entre tres pàgines webs que es dediquen a això mateix. D'aquest estudi, farem una taula comparativa que ens ajudarà a veure de quina manera podem innovar.

Primer de tot, especificar que aquestes pàgines web donen molts serveis a part del d'anunciar una propietat en venda, algunes d'elles tenen calculadores d'hipoteques, informacions del mercat immobiliari, anuncis de diferents empreses relacionades oferint els seus serveis... Nosaltres, per realitzar l'estudi comentat abans, només ens centrarem en com un usuari pot buscar una propietat, com pot veure les característiques de l'anunci escollit, com es pot posar en contacte amb el propietari.

Les pàgines webs escollides per a realitzar l'estudi seran www.fotocasa.es, www.inmobiliaria.com i www.hogaria.net.

3.1. Web www.fotocasa.es

Imatge 1. Web fotocasa

Aquesta pàgina web, és de les més visitades a l'hora de comprar una propietat, ja que ofereix un munt de serveis, a part d'informar a la gent dels possibles immobles en venda.

Des d'un principi, la web de fotocasa et permet fer una cerca de tres maneres diferents, fer una cerca general mitjançant una o més paraules claus, o acotar la cerca dins d'una regió del mapa, o també et permet buscar per províncies. Si es fa la cerca mitjançant el mapa, et va ampliant el mapa de la zona delimitant les províncies o municipis fins que et trobes en el desitjat.

Un cop realitzada la cerca, el llistat que se'ns ofereix mostra les següents característiques: el número de fotos de l'anunci, la direcció, els metres quadrats, el número d'habitacions, la relació preu per metre quadrat i el preu de la vivenda.

Seleccionant un anunci qualsevol, per veure el menú de l'anunci detallat, observem que a dalt de tot ens surten quatre accions, veure la fitxa, veure les fotos, conèixer la zona, i veure el preu mig. Veiem també les característiques del immoble, així com la possibilitat d'afegir l'immoble a una llista de preferits, recomanar l'anunci a un amic, comunicar un error i enviar un mail al anunciant. També ens informa del número de referència del anunci, així com la última vegada que es va actualitzar, i el nombre de visites que ha tingut.

3.2. Web www.inmobiliaria.com

Imatge 2. Web inmobiliaria

Aquesta pàgina, només carregar-se ens mostra un buscador per paraula clau, i una opció de buscar també per província. Ens informa també dels últims anuncis que s'han afegit recentment.

Un cop feta la cerca, podem anar reduint el llistat que ens ofereix, perquè a la banda esquerra de la pàgina ens surt unes opcions discriminatòries per anar acotant la cerca. A la banda dreta, ens deixa escollir diferents poblacions de la província seleccionada prèviament.

Seleccionem un anunci per veure el seu detall, i observem les característiques, però també molts anuncis pel mig que dificulten trobar-les. Veiem també que ens permet enviar un mail al anunciant, o veure el seu telèfon per si volem contactar nosaltres directament i ens permet recomanar l'anunci a un amic.

També comentar que la informació que ens dona el llistat de la cerca i la que ens mostra l'anunci detallat només varia en la possibilitat de veure les fotos en el detall. La resta és el mateix.

3.3. Web www.hogaria.net

Imatge 3. Web Hogaria

En aquesta pàgina web, podem observar només carregar-se un mapa per seleccionar la província en la qual volem buscar, i unes opcions de cerca. En fer la cerca aquesta pàgina no es gaire intuïtiva, t'has de fixar bé per trobar el botó que et mostra el llistat trobat amb les especificacions demanades.

Un cop veiem el llistat, observem les següents característiques: si l'anunci conté fotografies, la zona, l'estat del immoble, els metres quadrats, el número d'habitacions, el número de lavabos, el preu i una relació del preu per metre quadrat.

En el detall del anunci un cop seleccionat, ens mostra les característiques de la propietat, juntament amb les fotografies. Tenim una opció per veure la zona en el mapa, però no la localització exacte del immoble. També podem enviar un mail al anunciant, o contactar amb ell ja que ens mostra el telèfon. Ens permet també enviar-ho a un amic, guardar l'anunci com a preferit, veure el número de visites i el número de referència.

3.4. Taula resum

Categories/Webs	www.fotocasa.es	www.inmobiliaria.com	www.hogaria.net
Classificació extensa	si	si	si
Localització en Google Maps	si	no	Només la zona
Cercador avançat	si	si	si
Possibilitat de penjar vídeos	no	no	no
Possibilitat de penjar fotografies	si	si	si
Estadístiques de visita	si	no	si
Correu intern	si	si	si
Possibilitat de recomanació	si	si	si

Taula 1. Taula comparativa

Com podem apreciar en la comparativa, aquest tema està més que treballat. El gran avantatge que podem oferir nosaltres, és la bona localització del immoble dins del mapa, ja que com hem vist, no totes tenen l'opció de veure la localització exacte de la propietat dins el mapa, només mostren un mapa de l'entorn aproximat. L'altre característica important i també ens donaria una gran avantatge, és la d'incorporar la possibilitat que l'usuari pugui mostrar un vídeo de la seva propietat, d'aquesta manera sempre es pot veure molt millor que no amb unes quantes fotografies. Nosaltres oferirem les dues opcions, fotografies, i

vídeos, d'aquesta manera l'usuari que estigui buscant una propietat per comprar, podrà veure clarament les seves característiques visualment. Com a últim avantatge, també hem d'incorporar un cercador detallat, on a l'usuari li sigui fàcil poder acotar la cerca dins les seves exigències o necessitats.

4. Tecnologies i metodologia

4.1. Tecnologies

4.1.1. Elecció

El llenguatge utilitzat pel desenvolupament de l'aplicació és Java, fent ús de la tecnologia Java Server Pages (JSP) per a les vistes (GUI).

A l'hora d'escollir els marcs de treball, es va decidir fer ús del framework Struts, que és un marc de treball per al desenvolupament d'aplicacions Web Java, que es basa en el patró Model Vista Controlador (MVC). També es va decidir fer ús del framework iBatis, que és un marc de treball per a la comunicació del domini amb la persistència, basat amb el patró Data Access Object (DAO), que el mateix iBatis implementa el seu propi framework DAO (ibatis-dao.jar). Seguidament explicarem, amb detalls, els frameworks usats i els patrons amb els que es basen.

4.1.2. Patró Model Vista Controlador (MVC) i Framework Struts

El principal objectiu de l'arquitectura MVC és aïllar tant les dades de l'aplicació com l'estat de la mateixa (model), del mecanisme utilitzat per a representar aquest estat (vista), així com per a modular aquesta vista i modelar la transició entre estats del model (controlador). Les aplicacions MVC es divideixen en tres grans àrees funcionals:

- Vista: La presentació de les dades.
- Controlador: El que atendrà les peticions i components per a la presa de decisions de l'aplicació.
- Model: La lògica del negoci o servei i les dades associades amb l'aplicació.

El propòsit del MVC és aïllar els possibles canvis. És una arquitectura preparada per als canvis, que desacobla dades i lògica de negoci de la presentació, permetent l'actualització i desenvolupament independent de cada un dels components anomenats.

En el nostre cas, les vistes s'han implementat usant les JSP, el controlador amb el servlet de Struts, i el model amb les classes Java que contenen la lògica de l'aplicació.

El projecte Struts va sorgir el Maig del 2000, el seu creador, Craig R.McClanahan volia proporcionar un marc de treball MVC estàndard en la comunitat Java.

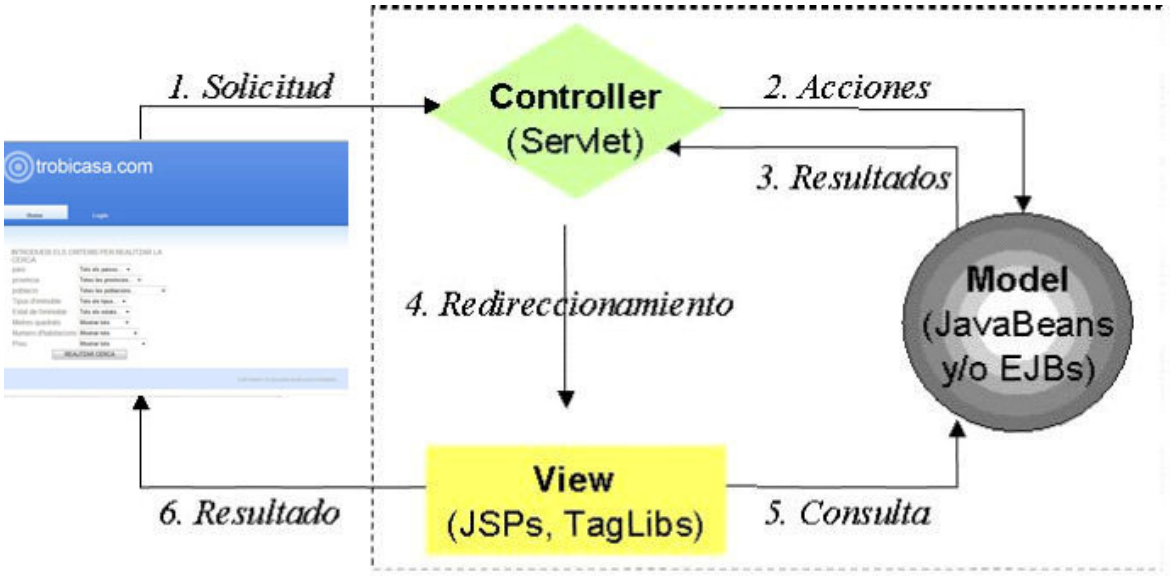
Struts implementa el patró MVC i per això ens ha de proveir i donar accessibilitat a un Controlador, al Model, i a la Vista. El Controlador es troba implementat per Struts, tot i que si fos necessari es pot heretar i ampliar, i el workflow de l'aplicació es programa des d'un arxiu XML. Les accions que s'executen sobre el model d'objectes de negoci s'implementen basant-se en classes predefinides pel framework.

La generació de les interfícies es suporta mitjançant un conjunt de Tags predefinits pel framework Struts, l'objectiu del qual és evitar l'ús de Scriplets, l'avantatge que això genera és una gran mantenibilitat. Logísticament, separa clarament el desenvolupament d'interfície del workflow i lògica del negoci, permetent desenvolupar-les en paral·lel o amb personal especialitzat.

Funcionament de Struts en aplicacions Web

El navegador genera una sol·licitud que és atesa pel Controlador. Ell mateix s'encarrega d'analitzar la sol·licitud, seguir la configuració que se li ha definit en el seu XML, i cridar al Action corresponent passant-li els paràmetres enviats. L'Action instanciarà i utilitzarà els objectes de negoci necessaris per concretar la feina. Segons el resultat que retorni l'Action, el Controlador derivarà la generació de la interfície a una o més JSP's, les quals podran consultar objectes del Model de negoci per mostrar informació dels mateixos en la Vista.

El següent gràfic ens dona una visió més detallada del funcionament del framework Struts:



Imatge 4. Esquema funcionament Struts

Vista

La Vista està formada per un conjunt de pàgines JSP. Struts prové suport per a construir aplicacions amb moltes utilitats gràcies a la utilització de Tags especials (TagLibraries).

Controlador

El Controlador comprèn la funcionalitat involucrada des de que un usuari genera una comanda HTTP fins que es genera la interfície de resposta. Pel mig, cridarà als objectes de negoci del Model per a que resolguin la funcionalitat pròpia de la lògica de negoci i segons el resultat d'aquesta cridarà la JSP que ha de generar la interfície resultant.

Model

El Model comprèn tots els Objectes de Negoci on s'implementa la lògica de negoci i on s'han de suportar tots els requisits funcionals del Sistema sense barrejar-ho amb parts corresponents al workflow que correspon al Controlador.

4.1.3. Frameworks iBatis, DAO i SqlMap

iBatis és un marc de treball de codi obert que es basa en capes, desenvolupat per Apache Software Foundation. S'ocupa de la capa de persistència, és a dir, es situa entre la lògica del negoci i la capa de base de dades.

iBatis associa objectes del domini amb sentències SQL o processos emmagatzemats en fitxers descriptius XML, simplificant així la utilització de la base de dades.

iBatis està constituït per dos frameworks independents que generalment s'utilitzen junts: DAO y sqlMaps. El primer simplifica la implementació del patró DAO, i el segon simplifica la persistència d'objectes en bases de dades relacionals.

El patró DAO ens ajuda a organitzar les tasques de persistència del objectes i ens permet definir múltiples implementacions per a un mateix objecte mitjançant la definició d'una interfície. Amb el framework iBatis DAO podem configurar quan fer servir la implementació d'aquesta interfície que pertoca a cada moment sense tenir la necessitat de modificar codi.

El framework iBatis sqlMap simplifica les tasques de comunicació amb la base de dades resumint-les en la configuració de fitxers XML, i funciona amb pràcticament qualsevol base de dades , només cal especificar-li el Driver.

Podem dividir la capa de persistència de iBatis en 3 capes:

- La capa d'Abstracció és l'interface amb la capa de lògica del negoci, fent la funció de façana entre l'aplicació i la persistència. S'implementa amb el ja comentat framework DAO propi de iBatis.

- La capa del framework de persistència és l'interface amb el gestor de la base de dades, fent-se càrrec de la gestió de les dades mitjançant un API. Per aquesta capa s'utilitza el ja comentat framework sqlMap propi de iBatis.

- La capa del Driver s'encarrega de la comunicació amb la pròpia base de dades fent servir el Driver necessari per a cada cas.

4.1.4. Google Maps

Google Maps és un servidor d'aplicacions de mapes gratuït (sempre que no sigui per extreure'n beneficis amb el seu ús) que ofereix Google.

Aquesta aplicació es pot integrar com a un servei en una altre aplicació web, de forma gratuïta, només cal registrar-se com a usuari de Google, i sol·licitar el Google Code. Amb aquest codi, ja es pot fer ús del API de Google Maps, i ja es pot integrar el codi JavaScript que inicialitzarà el mapa dins de la nostre aplicació.

Google Maps funciona bàsicament amb JavaScript, fent servir AJAX per actualitzar la informació mostrada en el mapa en qualsevol moment.

4.1.5. NetBeans

NetBeans es refereix a una plataforma pel desenvolupament d'aplicacions d'escriptori utilitzant Java i a un entorn de desenvolupament integrat (IDE) desenvolupat utilitzant la Plataforma NetBeans.

S'ha decidit realitzar el projecte amb el NetBeans 6.7.1 ja que és una eina de software lliure, i a part, és l'eina que s'utilitza a l'escola per realitzar les pràctiques. NetBeans té el recolzament d'una gran base d'usuaris, una comunitat en constant creixement, i gairebé amb 100 socis arreu del món. Sun MicroSystems va fundar el projecte de codi obert NetBeans el Juny del 2000 i continua sent el patrocinador principal dels projectes.

4.2. Metodologia

Si tenim en compte els nivells de complexitat en els que es troben actualment els projectes de desenvolupament de software, la industria s'ha vist obligada a realitzar models en els que es detallen els diferents processos software per elaborar productes de qualitat.

El model de desenvolupament de software que s'ha utilitzat per a realitzar aquest projecte ha estat el model iteratiu i incremental.

La idea principal del model iteratiu i incremental és desenvolupar un sistema de programes de manera incremental, d'aquesta manera, el programador treu partit del que ha après durant el transcurs del desenvolupament anterior. Perquè això passi, prèviament s'ha d'haver dividit el projecte en diferents iteracions. Aquestes iteracions, nosaltres les hem definit agrupant en cada una d'elles diferents casos d'ús, d'aquesta manera fem possible la proba de les funcionalitats especificades en cada una d'elles.

El disseny és normal que vagi canviant al llarg de les iteracions, perquè en cada una d'elles s'implementa només el necessari per complir les funcionalitats especificades.

5. Pressupost i planificació

5.1. Pressupost del desenvolupament del projecte

A continuació es detallaran els costos que s'han generat en el desenvolupament del projecte. Els costos de software han estat 0 € ja que tot el programari utilitzat a estat open source (software lliure).

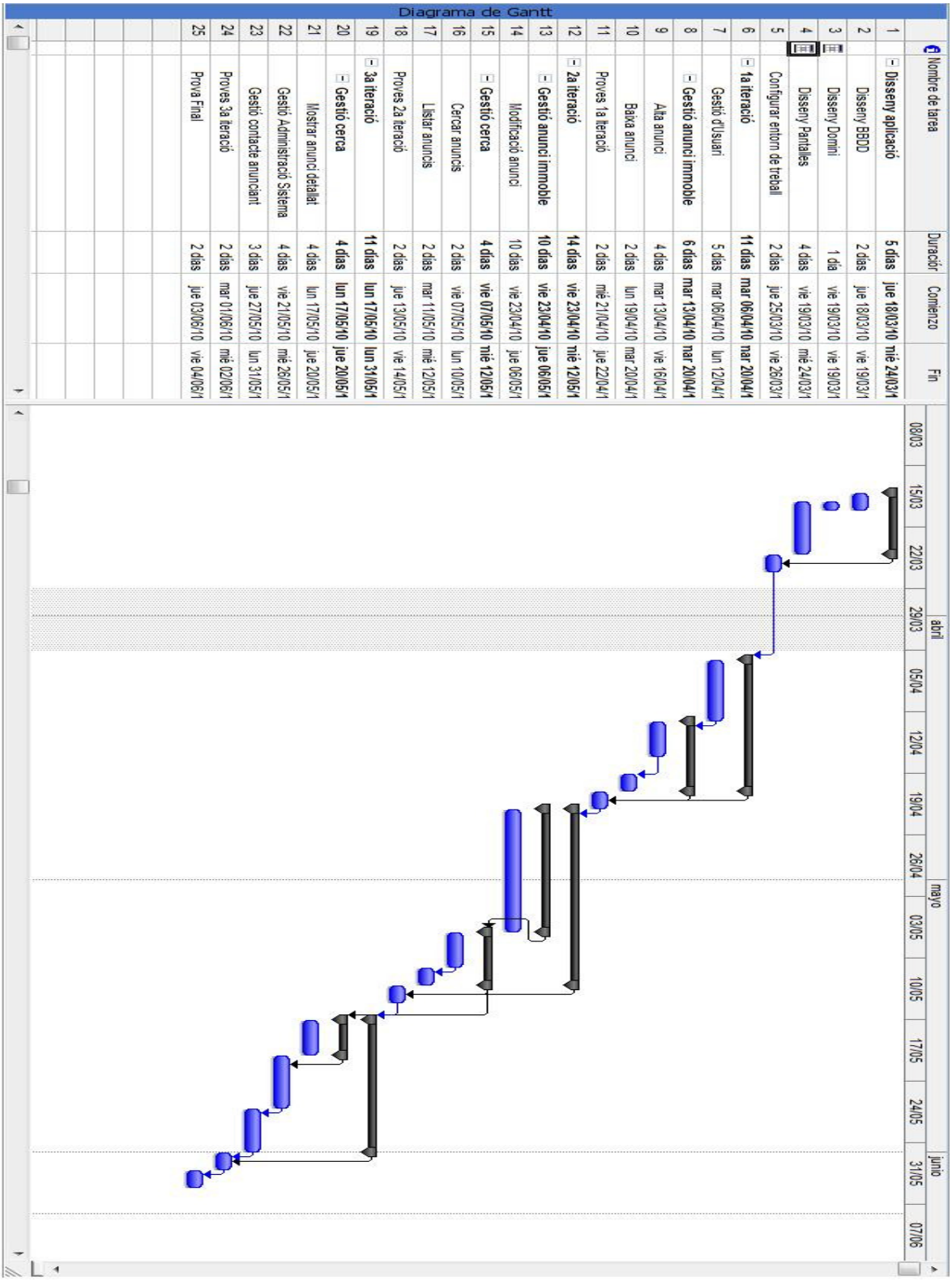
- Portàtil Acer Aspire 6920G té un cost aproximat de 800 €, si tenim en compte que la duració del projecte és de 4 mesos, hem d'imputar un cost aproximat de 266 €.
- El cost de la connexió a internet subministrat per part de Telefònica és de 70 € aproximadament, si comptem els 4 mesos, són 280 €.
- El cost del programador que realitzarà el projecte, contant que és aproximadament d'uns 30 €/h, i contant que el projecte durarà unes 300h aproximadament, surt per 9.000 €.

Resumint les dades anteriors en una taula de costos:

	Preu	Quantitat	Subtotal
Recursos Software	0	0	0
Recursos Hardware	800	4 mesos	266 €
Cost internet	70	4 mesos	280 €
Programador	30	300 hores	9.000 €
		Total:	**9.546 €**

Taula 2. Taula de costos

5.2. Planificació temporal



Imatge 5. Diagrama de Gantt

6. Anàlisi

6.1. Anàlisi de requeriments

El desenvolupament del projecte passa primer de tot per un estudi previ dels requisits perquè l'usuari tingui accés a les funcionalitats descrites en l'especificació dels casos d'ús més endavant.

Primer de tot, s'ha de realitzar un anàlisi dels possibles usuaris que faran ús de l'aplicació, ja que tindran accés a diferents funcionalitats. Podem diferenciar entre 2 tipus de rols, Usuari invitat i Usuari registrat. Algunes d'aquestes funcionalitats estaran compartides per els diferents rols.

L'usuari invitat és aquell usuari que visita l'aplicació per tal de recollir informació, d'aquesta manera, se li permetrà realitzar cerques dels immobles que estiguin a la venda, i també es permetrà l'accés al detall d'aquest immoble, podent observar llavors les característiques del immoble, com poden ser el tipus d'immoble, el seu estat, la direcció, la descripció que fa el propietari, el preu, les fotografies, el vídeo, i la localització en el mapa de la propietat.

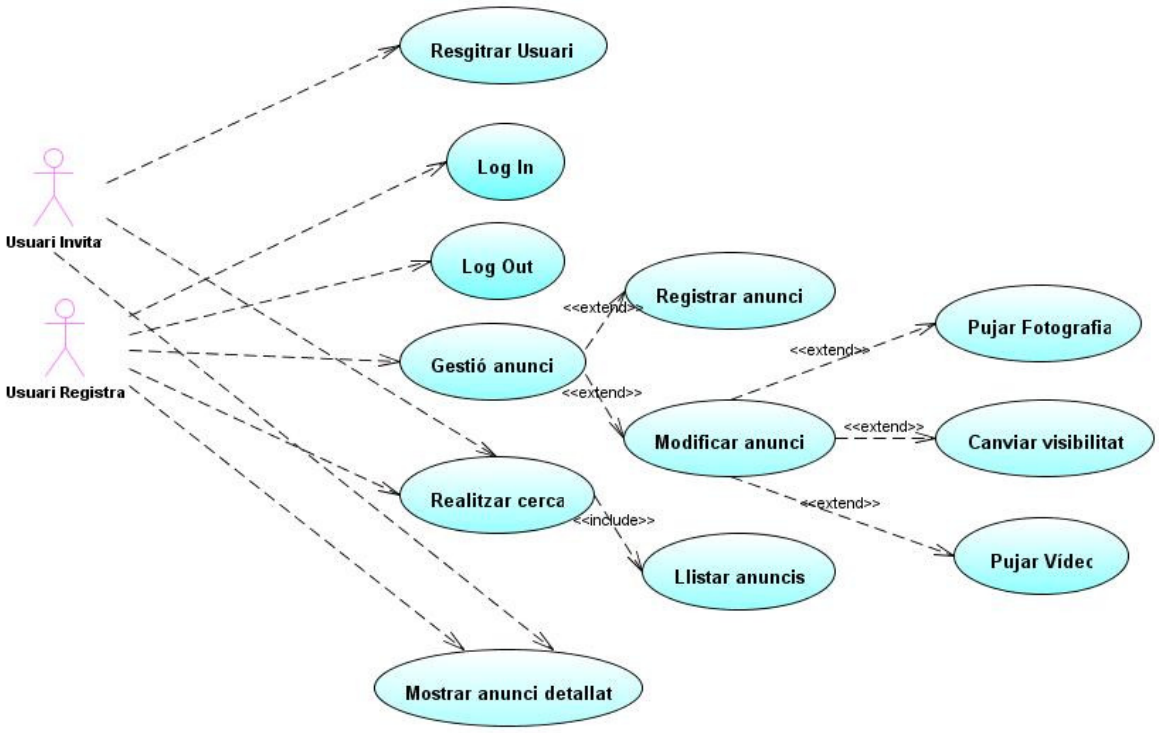
L'usuari registrat, que per ser-ho, prèviament ha estat un usuari invitat i ha dut a terme un registre. En aquest registre, l'usuari ha d'haver introduït el mail, nom, cognoms i el telèfon. Un cop l'usuari ja s'ha registrat en l'aplicatiu, el següent cop que entri, podrà fer Login per canviar el rol. L'usuari registrat, té com a funcionalitats registrar una propietat per a la seva venda, omplint un formulari en el qual se li demana el país, província, població, direcció, metres quadrats, número d'habitacions, descripció i preu. També té com a funcionalitats penjar fotos o un vídeo a un immoble que hagi estat registrat prèviament, i també pot canviar la visibilitat d'un immoble registrat de cara als usuaris invitats, en el cas que hagués estat venut aquest immoble.

Els dos rols diferents que pot tenir un usuari, poden a vegades compartir funcionalitats, com és el cas de la cerca, i la visualització detallada d'un immoble resultant de la cerca.

A continuació es detallaran les diferents funcionalitats que hem anomenat anteriorment, comentant quin és el seu objectiu, qui és l'actor principal que realitzarà l'acció, les pre-condicions, post-condicions, flux normal i, en el cas de tenir-ne, flux alternatiu.

6.2. Especificació dels casos d'ús

6.2.1. Model de casos d'ús



Imatge 6. Diagrama Casos d'ús

6.2.2. Actors principals

Com es pot observar en el model de casos d'ús, el sistema interactuarà amb un actor, que podrà tenir els dos rols definits, invitat, o registrat. Els diferents actors fan una sèrie de

peticions al sistema. Per últim, recordar que diferents actors poden compartir peticions al sistema.

6.2.3. Casos d'ús

Cas d'ús Registrar Usuari

Actor principal: Usuari Invitat

Pre-condicions: No hi ha cap pre-condició.

Post-condició: L'usuari quedarà enregistrat en el sistema, i el rol passarà a ser de usuari registrat.

Flux normal:

1. L'usuari executa l'acció registrar usuari.
2. El Sistema li demana les dades.
3. L'usuari introdueix el mail, nom, cognoms, telèfon y contrasenya i prem el botó registrar.
4. El Sistema comprova les dades i les emmagatzema.

Flux alternatiu:

*. En qualsevol moment l'usuari pot tancar l'aplicació.

4.1. Les dades no són correctes.

4.1.1. El Sistema informa del error, i torna al pas 2.

Cas d'ús Log In

Actor principal: Usuari Invitat

Pre-condicions: Per poder fer Log In satisfactòriament, l'usuari s'ha d'haver registrat en el Sistema en algun moment previ.

Post-condició: L'usuari quedarà identificat pel Sistema, canviant el seu rol a usuari registrat.

Flux normal:

1. L'usuari executa l'acció Log In.
2. El Sistema li demana les dades.
3. L'usuari introdueix el mail i la contrasenya i prem el botó entrar.
4. El Sistema comprova les dades.

Flux alternatiu:

*. En qualsevol moment l'usuari pot tancar l'aplicació.

4.1. Les dades no són correctes.

4.1.1. El Sistema informa del error, i torna al pas 2.

Cas d'ús Log Out

Actor principal: Usuari Registrat

Pre-condicions: L'usuari ha d'haver fet el cas d'ús Log In amb èxit.

Post-condició: El Sistema ja no tindrà cap usuari en sessió.

Flux normal:

1. L'usuari executa l'acció Log Out.
2. El Sistema realitza l'acció.

Flux alternatiu:

*. En qualsevol moment l'usuari pot tancar l'aplicació.

Cas d'ús Realitzar Cerca

Actor principal: Usuari Invitat, Usuari Registrat

Pre-condicions: No hi ha cap pre-condició.

Post-condició: S'executarà el cas d'ús llistar anuncis.

Flux normal:

1. El Sistema mostra els paràmetres disponibles.
2. L'usuari tria els paràmetres escaients.
3. El Sistema consulta les dades.
4. El Sistema executa el cas d'ús Llistar anuncis.

Flux alternatiu:

*. En qualsevol moment l'usuari pot sortir de l'aplicació.

Cas d'ús Llistar anuncis

Actor principal: Sistema.

Pre-condicions: S'ha d'haver realitzat una cerca.

Post-condició: L'usuari visualitzarà els resultats de la cerca previa.

Flux normal:

1. El Sistema mostra els anuncis trobats en la cerca.

Flux alternatiu:

*. En qualsevol moment l'usuari pot tancar l'aplicació.

Cas d'ús Mostrar anunci detallat

Actor principal: Usuari Invitat, Usuari Registrat

Pre-condicions: S'ha d'haver executat el cas d'ús realitzar cerca satisfactòriament, i han d'estar llistats els anuncis trobats.

Post-condició: L'usuari veurà en detall les característiques del anunci que hagi seleccionat.

Flux normal:

1. L'usuari prem el botó veure detall del anunci que l'interessi.
2. El Sistema consulta les dades.
3. El Sistema mostra al usuari el detall del anunci.

Flux alternatiu:

*. En qualsevol moment l'usuari pot tancar l'aplicació.

Cas d'ús registrar anunci

Actor principal: Usuari Registrat

Pre-condicions: L'usuari ha d'haver executat el cas d'ús Log In.

Post-condició: L'anunci quedarà registrat, i podrà ser vist exeutant el cas d'ús Realitzar cerca.

Flux normal:

1. L'usuari executa l'acció registrar anunci.
2. El Sistema li demana que triï el país.
3. L'usuari selecciona el país.
4. El sistema li demana que triï una província.
5. L'usuari selecciona una província.

6. El Sistema li demana que triï una població.
7. L'usuari selecciona una població.
8. El Sistema li mostra una aproximació en el mapa de la població escollida i li demana l'adreça.
9. L'usuari introdueix l'adreça, i prem el botó Localitzar adreça en el mapa.
10. El Sistema li mostra la localització de l'adreça en el mapa i li demana que introdueixi el resta de dades.
11. L'usuari escolleix el tipus d'immoble, l'estat del immoble, i introdueix els metres quadrats, el numero d'habitacions, la descripció, el preu i prem el botó Registrar anunci.
12. El Sistema comprova les dades i les emmagatzema.

Flux alternatiu:

*. En qualsevol moment l'usuari pot tancar l'aplicació.

10.1. El Sistema no troba l'adreça en el mapa.

10.1.1. El Sistema informa que no ha pogut localitzar l'adreça.

10.1.2. L'usuari busca la localització de l'adreça en el mapa i fa clic sobre ella.

10.1.3. El Sistema registra el punt on a fet clic l'usuari i continua amb l'execució del cas d'ús.

12.1. Les dades no són correctes.

12.1.1. El Sistema mostra l'error i torna al pas 1.

Cas d'ús Pujar foto

Actor principal: Usuari Registrat

Pre-condicions: S'ha d'haver executat amb èxit un cas d'ús de Registrar anunci.

Post-condició: La foto quedarà registrada en l'anunci corresponent.

Flux normal:

1. L'usuari executa l'acció pujar foto.
2. El Sistema li demana les dades.
3. L'usuari introdueix la ubicació de la foto i prem el botó Pujar foto.
4. El Sistema comprova les dades i les emmagatzema.
5. El Sistema torna al pas 2.

Flux alternatiu:

*. En qualsevol moment l'usuari pot tancar l'aplicació.

4.1. Les dades no són correctes.

4.1.1. El Sistema informa del error, i torna al pas 2.

Cas d'ús Pujar vídeo

Actor principal: Usuari Registrat

Pre-condicions: S'ha d'haver executat amb èxit un cas d'ús de Registrar anunci.

Post-condició: El vídeo quedarà registrat en l'anunci corresponent.

Flux normal:

1. L'usuari executa l'acció pujar vídeo.
2. El Sistema li demana les dades.
3. L'usuari introdueix la ubicació de la vídeo i prem el botó Pujar vídeo.
4. El Sistema comprova les dades i les emmagatzema.

Flux alternatiu:

*. En qualsevol moment l'usuari pot tancar l'aplicació.

4.1. Les dades no són correctes.

4.1.1. El Sistema informa del error, i torna al pas 2.

Cas d'ús Canviar visibilitat

Actor principal: Usuari Registrat

Pre-condicions: S'ha d'haver executat amb èxit un cas d'ús de Registrar anunci.

Post-condició: El vídeo ja no serà visible executant el cas d'ús Realitzar cerca.

Flux normal:

1. L'usuari executa l'acció canviar visibilitat.
2. El Sistema mostra els anuncis que té publicats l'usuari, mostrant la visibilitat actual.
3. L'usuari selecciona un anunci i prem el botó Venut..
4. El Sistema canvïa la visibilitat del anunci escollit, i ho emmagatzema.
5. El Sistema torna al pas 2.

Flux alternatiu:

*. En qualsevol moment l'usuari pot tancar l'aplicació.

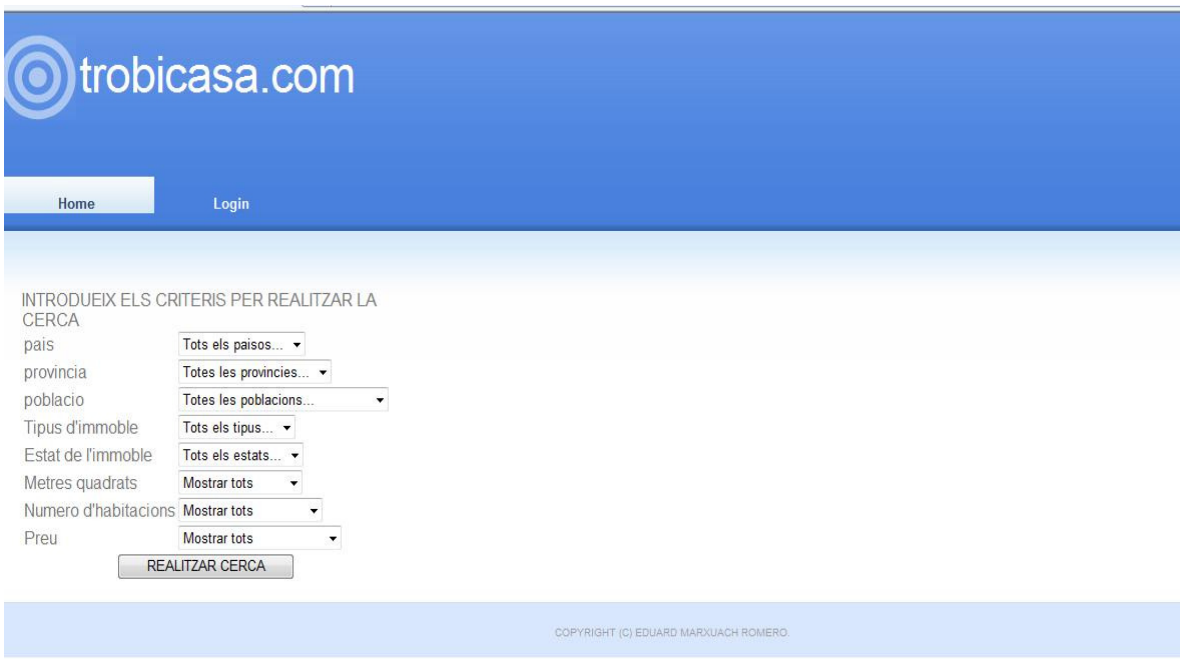
7. Disseny

7.1. Disseny de la interfície gràfica d'usuari

La interfície gràfica d'usuari (IGU) s'ha dissenyat de manera que sigui còmode per a qualsevol tipus d'usuari que la pugui fer servir.

L'estructura de la pàgina web és sempre la mateixa. Tenim el logo i el títol de la web en la part superior esquerra de la pantalla i els botons del menú a sota d'aquests. Tant la part del títol com la del menú són d'un color blau més fosc per tal de separar-ho de la part variable de la web, que és la part on es mostra el contingut, en blau clar. D'aquesta manera separem la part variable, de la part estàtica.

La navegabilitat és fàcil i pràctica, en tot moment es sap en quina secció s'està ja que el botó del menú que s'ha seleccionat canvia de color. També hi ha indicacions en tots els apartats perquè l'usuari sàpiga que fer. Es pot obtenir més informació de la navegabilitat en l'apartat 9, la prova significativa.



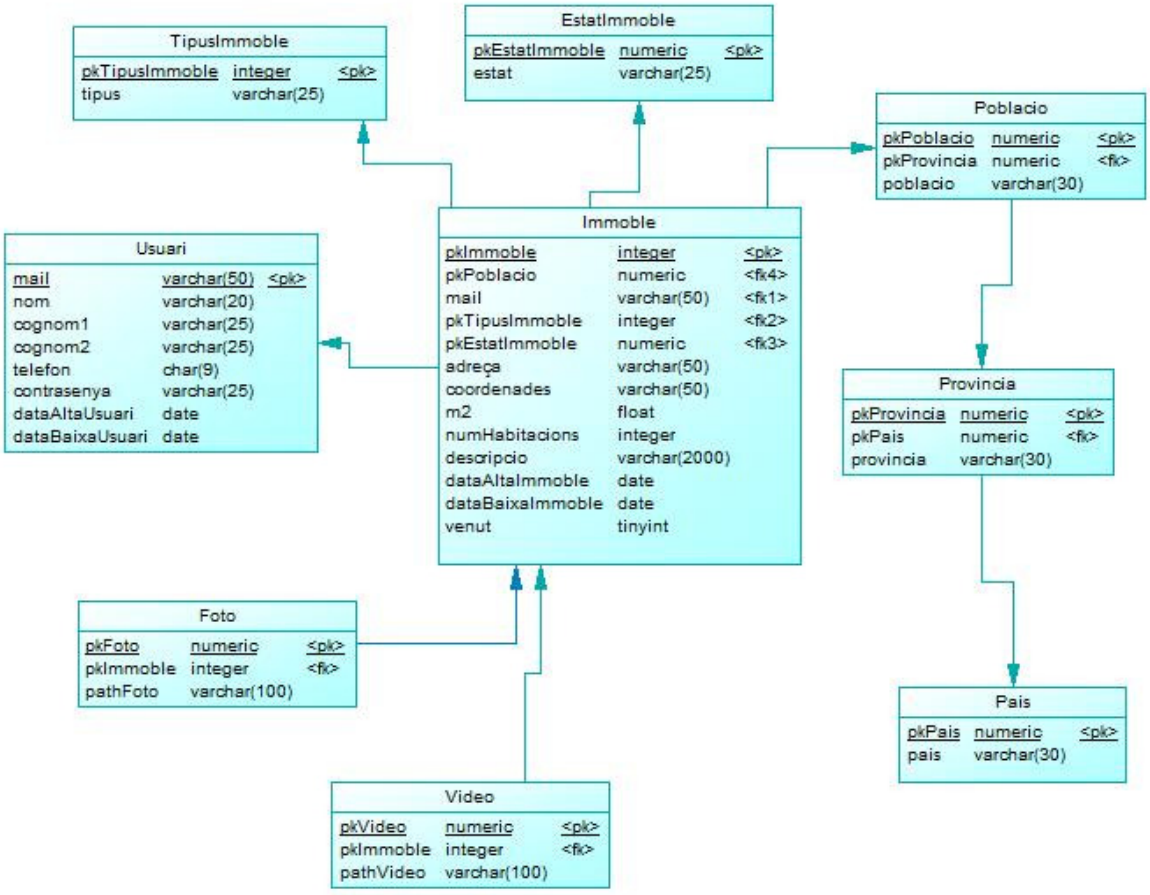
Imatge 7. Disseny IGU

7.2. Disseny BB.DD.

La base de dades serà l'encarregada de mantenir persistents les dades que l'aplicació pugui necessitar, a més, s'ocuparà de part de la concurrència de la web, i subministrarà totes les dades requerides per l'aplicació.

La base de dades elegida és de tipus relacional. Tot seguit es mostrarà el diagrama físic de la base de dades en què ens hem basat per fer una bona base de dades estructurada i ferma.

En quant al sistema gestor de base de dades (SGBD), hem escollit un de codi lliure, com és el MySql.



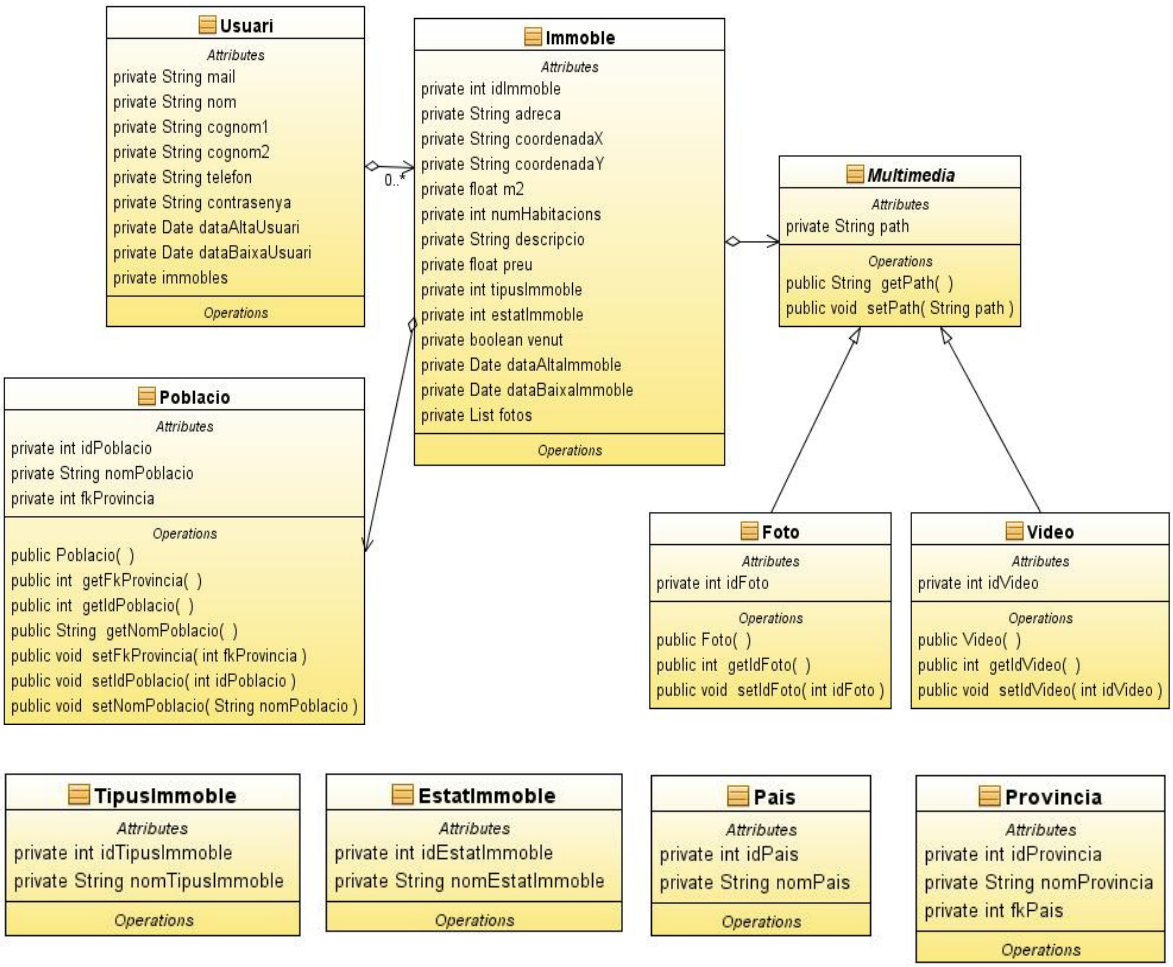
Imatge 8. Model BB.DD.

A continuació es comentaran les diferents relacions i les seves cardinalitats:

- Usuari - Immoble: Un usuari pot tenir cap o molts immobles, i un immoble només és d'un sol usuari.
- Immoble - Estat Immoble: Un immoble només té un estat d'immoble, i un estat d'immoble pot pertànyer a cap o molts immobles.
- Immoble - Tipus Immoble: Un immoble només té un tipus d'immoble, i un tipus d'immoble pot pertànyer a cap o molts immobles.
- Immoble – Població: Un immoble és d'una sola població, i una població pot tenir cap o molts immobles.
- Població – Província: Una població només pertany a una província, i una província pot tenir moltes poblacions.
- Província – País: Una província pertany a un sol país, i un país pot tenir moltes províncies.
- Immoble – Vídeo: Un immoble pot tenir cap, o un vídeo, i un vídeo només pot pertànyer a un immoble.
- Immoble – Foto: Un immoble pot tenir cap, o moltes fotos, i una foto només pot ser d'un immoble.

7.3. Disseny del Domini

En un primer moment, es va dissenyar un domini amb les classes suficients per poder fer anar el projecte, però, més endavant, ens vam adonar que fent servir el framework iBatis, era millor fer tantes classes com taules tenim a la base de dades, com a resultant, el domini final que es va fer de l'aplicació és el següent:



Imatge 9. Model del Domini

Com podem observar, les classes amb les que treballarà el domini són aquelles en les que apareixen les relacions, les altres classes, són les resultants de treballar amb iBatis, ja que

per realitzar consultes necessàries des de la presentació a la base de dades, per obtenir els resultats són necessaris les classes "mirall" de les taules de la base de dades.

També hem considerat necessària la generalització de la classe *Multimèdia* en les classes *Foto* i *Vídeo*, ja que compartien el mateix atribut principal.

Anem ara a analitzar les agregacions:

- Usuari: Un usuari contindrà una llista dels anuncis d'immobles que hagi donat d'alta.
- Immoble: Un immoble, contindrà la població a la qual pertany. A part, també tindrà una llista de les fotos pròpies i el seu vídeo.

8. Desenvolupament

En aquest apartat, s'explicaran detalls del desenvolupament del projecte que han estat costosos, com per exemple el fet de treballar amb iBatis, o el de utilitzar l'API de Google Maps. Per mostrar-ho, què millor que mostrar la implementació dels casos d'ús que impliquen aquestes tecnologies, (en el cas de iBatis, tots el casos d'ús ho fan), però es mostrarà la Realització de la cerca que ha estat el més dificultós, i en el cas de Google Maps, es mostrarà el cas d'ús Registrar Anunci, que és on millor es pot apreciar el seu funcionament.

8.1. Configuració del Framework iBatis, DAO i SqlMap.

Primer de tot és necessari comentar com funciona iBatis, ja que tots els casos d'ús l'usen per recollir dades o inserir-les en la base de dades.

Primer de tot, hem de baixar les llibreries necessàries, i afegir-les al nostre projecte:

Imatge 10. Llibreries iBatis

Seguidament, configurem l'arxiu *SqlMapConfig*.xml. En aquest arxiu, es configura la connexió amb la base de dades, i és el fitxer on es declaren els diferents *.xml* corresponents a les classes del domini per enllaçar-les amb la seva taula corresponent de la base de dades.

```
<!DOCTYPE sqlMapConfig
    PUBLIC "-//ibatis.apache.org//DTD SQL Map Config 2.0//EN"
    "http://ibatis.apache.org/dtd/sql-map-config-2.dtd">

<sqlMapConfig>
<settings useStatementNamespaces="true"/>
  <transactionManager type="JDBC" commitRequired="false">
    <dataSource type="SIMPLE">
      <property name="JDBC.Driver" value="com.mysql.jdbc.Driver"/>
      <property name="JDBC.ConnectionURL" value="jdbc:mysql://localhost/pfc"/>
      <property name="JDBC.Username" value="root"/>
      <property name="JDBC.Password" value=""/>
    </dataSource>
  </transactionManager>

  <sqlMap resource="com/persistence/ibatis/dao/maps/usuari_SqlMap.xml"/>
  <sqlMap resource="com/persistence/ibatis/dao/maps/immoble_SqlMap.xml"/>
  <sqlMap resource="com/persistence/ibatis/dao/maps/poblacio_SqlMap.xml"/>
  <sqlMap resource="com/persistence/ibatis/dao/maps/provincia_SqlMap.xml"/>
  <sqlMap resource="com/persistence/ibatis/dao/maps/pais_SqlMap.xml"/>
  <sqlMap resource="com/persistence/ibatis/dao/maps/tipusImmoble_SqlMap.xml"/>
  <sqlMap resource="com/persistence/ibatis/dao/maps/estatImmoble_SqlMap.xml"/>
  <sqlMap resource="com/persistence/ibatis/dao/maps/foto_SqlMap.xml"/>
  <sqlMap resource="com/persistence/ibatis/dao/maps/video_SqlMap.xml"/>


</sqlMapConfig>
```

Imatge 11. Exemple SqlMapConfig

En aquests fitxers *.xml* declarats en el *SqlMapConfig*.xml, és on s'allotjaran les diferents accions realitzades cap a la base de dades, ja siguin *inserts*, *selects* o *updates*. També contindran les diferents relacions de les taules de la base de dades a les classes del domini.

Seguidament podem veure un exemple de la definició d'un *SqlMap*.xml:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE sqlMap PUBLIC "-//ibatis.apache.org//DTD SQL Map 2.0//EN"
"http://ibatis.apache.org/dtd/sql-map-2.dtd">

<sqlMap namespace="foto">

<!-- ES DEFINEIX LA RELACIÓ DE LA COLUMNA DE LA TAULA AMB L'ATRIBUT DE LA CLASSE-->
    <resultMap id="FotoResult"  class="com.beans.Foto">
        <result column="idFoto" property="idFoto" />
        <result column="pathFoto" property="path" />
    </resultMap>

<!-- SELECT QUE RETORNA LES FOTOGRAFIES D'UN IMMOBLE-->
    <select id="selectFotosByImmoble" resultMap="FotoResult">
                SELECT *
                FROM FOTO
                WHERE (FKIMMOBLE) = (#IDIMMOBLE#);
    </select>

<!-- INSERT QUE INSEREIX UNA FOTOGRAFIA, AMB LA FK DEL IMMOBLE AL QUE CORRESPON-->
    <insert id="inserirFoto" parameterClass="HashMap">
        INSERT INTO FOTO (PATHFOTO, FKIMMOBLE)
        VALUES (#path#,#idImmoble#);
    </insert>

</sqlMap>
```

Imatge 12. Exemple FotoSqlMap.xml

El *ResultMap* que podem veure en la imatge, és la relació de la taula de la base de dades amb la classe del domini, on la *column* és la columna de la base de dades, i la *property* és l'atribut de la classe del domini que li fa referència. Això el que fa, és a l'hora de fer una *select*, crea un objecte buit de la classe corresponent amb el constructor per defecte, i utilitza els sets() per donar valors als atributs del objecte creat. Pot haver-hi tants *ResultMaps* definits diferents com ens sigui necessari.

En el select, podem apreciar que té un identificador, que és amb el que es cridarà l'acció, i té un resultMap, que fa referència al que ja em comentat anteriorment.

En *l'insert*, podem veure també l'identificador amb el que es crida, i a part, veiem el tipus d'objecte que entra en l'acció, en aquest cas, un *HashMap*. Això es defineix amb la paraula clau *ParameterClass*. Aquests valors del *HashMap*, simplement posant les claus que fan

referència al valor dins del *insert* entre el caràcters #Clau#, ja ho transforma ell directament com es pot apreciaren la imatge anterior.

Per últim, a l'hora de configurar iBatis, també s'ha de crear l'arxiu *daoConfig.xml*, que és el que contindrà les definicions de les interfícies, i les classes que implementen a aquella interfície de la capa de persistència (que usa el framework DAO). En aquesta definició també s'ha d'incloure la ubicació del *SqlMapConfig.xml* comentat anteriorment, ja que aquestes classes que implementen a les interfícies definides són les que cridaran les accions definides en els *SqlMaps.xml*. Podem veure un exemple de la configuració seguidament:

```
<!DOCTYPE daoConfig
PUBLIC "-//ibatis.apache.org//DTD DAO Configuration 2.0//EN"
"http://ibatis.apache.org/dtd/dao-2.dtd">
<daoConfig>
    <context>
        <transactionManager type="SQLMAP">
            <property name="SqlMapConfigResource"
            value="com/persistence/ibatis/SqlMapConfig.xml" />
        </transactionManager>

        <dao interface="com.persistence.ibatis.dao.UsuariIbatisDAO"
        implementation="com.persistence.ibatis.dao.UsuariIbatisDAOImpl" />
        <dao interface="com.persistence.ibatis.dao.ImmobleIbatisDAO"
        implementation="com.persistence.ibatis.dao.ImmobleIbatisDAOImpl" />
        <dao interface="com.persistence.ibatis.dao.PoblacioIbatisDAO"
        implementation="com.persistence.ibatis.dao.PoblacioIbatisDAOImpl" />
        <dao interface="com.persistence.ibatis.dao.ProvinciaIbatisDAO"
        implementation="com.persistence.ibatis.dao.ProvinciaIbatisDAOImpl" />
        <dao interface="com.persistence.ibatis.dao.PaisIbatisDAO"
        implementation="com.persistence.ibatis.dao.PaisIbatisDAOImpl" />
        <dao interface="com.persistence.ibatis.dao.TipusImmobleIbatisDAO"
        implementation="com.persistence.ibatis.dao.TipusImmobleIbatisDAOImpl" />
        <dao interface="com.persistence.ibatis.dao.EstatImmobleIbatisDAO"
        implementation="com.persistence.ibatis.dao.EstatImmobleIbatisDAOImpl" />
        <dao interface="com.persistence.ibatis.dao.FotoIbatisDAO"
        implementation="com.persistence.ibatis.dao.FotoIbatisDAOImpl" />
        <dao interface="com.persistence.ibatis.dao.VideoIbatisDAO"
        implementation="com.persistence.ibatis.dao.VideoIbatisDAOImpl" />

    </context>
</daoConfig>
```

Imatge 13. Exemple daoConfig

8.2. Implementació Realitzar Cerca

Aquí definiré la línea d'execució del cas d'ús realitzar cerca. Aquesta acció es troba en el mateix índex de la pàgina web, ja que és una acció que qualsevol usuari pot realitzar.

Per realitzar l'acció, primer de tot, haurem d'escollir els paràmetres amb els quals volem realitzar la cerca, un cop escollits, premem el botó Realitzar Cerca, que cridarà al *Action* corresponent. En la imatge següent, podem apreciar els diferents paràmetres del índex amb els quals podem acotar la cerca, i el botó que cridarà l'acció:

Imatge 14. Formulari de Cerca

I per veure el funcionament intern, veiem en la següent imatge el tros de codi que defineix quina acció realitza el formulari anterior:

```
<form id="buscador" name="buscador" method="post" action="/PFC/buscar.do">
```

Imatge 15. Exemple de l'acció que realitza el formulari

Aquest */PFC/buscar.do*, està definit en el *struts-config.xml*, i el que fa aquesta definició és dirigir l'acció al *Action* corresponent. Ho podem veure a continuació:

```
<action name="BuscarActionForm" path="/buscar" scope="session"
type="com.web.actions.BuscarAction">
    <forward name="loadProvinciesOK" path="/index.jsp"></forward>
    <forward name="loadPoblacionsOK" path="/index.jsp"></forward>
    <forward name="buscarOK" path="/index.jsp"></forward>
</action>
```

Imatge 16. Exemple definició d'una acció dins StrutsConfig.xml

També cal dir, que tota *Action* té una *ActionForm* associada (en el cas que es necessitin recollir dades de la JSP que a cridat la *Action*), aquesta definició de les *ActionForm* també es recull en el fitxer *struts-config.xml*:

```
<form-beans>
    <form-bean name="UploadFileActionForm" type="com.web.actionForms.UploadFileActionForm"/>
    <form-bean name="VeureDetallActionForm" type="com.web.actionForms.VeureDetallActionForm"/>
    <form-bean name="ModificarImmobleActionForm" type="com.web.actionForms.ModificarImmobleActionForm"/>
    <form-bean name="BuscarActionForm" type="com.web.actionForms.BuscarActionForm"/>
    <form-bean name="RegistrarImmobleActionForm" type="com.web.actionForms.RegistrarImmobleActionForm"/>
    <form-bean name="LogoutActionForm" type="com.web.actionForms.LogoutActionForm"/>
    <form-bean name="LoginActionForm" type="com.web.actionForms.LoginActionForm"/>
    <form-bean name="RegistrarUsuariActionForm" type="com.web.actionForms.RegistrarUsuariActionForm"/>
    <form-bean name="PruebaActionForm" type="com.web.actionForms.PruebaActionForm"/>
</form-beans>
```

Imatge 17. Exemple definició dels FormBeans

Les *ActionForm* contenen els atributs que s'han de recollir de la JSP, i els gets i sets d'aquests atributs, d'aquesta manera les *Action* poden consultar les dades mitjançant una instancia del seu *ActionForm*.

Les classes *Action*, són les encarregades de cridar al controlador principal de l'aplicació, i d'ordenar les dades que aquest li retorna, en el cas que es retornin dades.

En l'execució que estem seguint, l'*Action* prepara les dades per cridar la funció corresponent del controlador. Per fer-ho, l'*action* ha de tenir una instancia del Controlador, i cridar a la funció *realitzarCerca()*, amb els corresponents paràmetres. Ho podem apreciar a continuació:

```
if(accio.equalsIgnoreCase("buscar")){
    int m2Min=0;
    int m2Max=999999999;
    float preuMin=0;
    float preuMax=999999999;
    switch(formulari.getM2()){
        case(1): m2Max=50;break;
        case(2): m2Min=49; m2Max=71;break;
        case(3): m2Min=69; m2Max=91;break;
        case(4): m2Min=89; m2Max=111;break;
        case(5): m2Min=109;break;
    }
    switch(formulari.getPreu()){
        case(1): preuMax=100001;break;
        case(2): preuMin=99999; preuMax=250001;break;
        case(3): preuMin=249999; preuMax=400001;break;
        case(4): preuMin=399999; preuMax=550001;break;
        case(5): preuMin=549999; preuMax=700001;break;
        case(6): preuMin=699999;break;
    }

    List resultat=Controlador.getInstance().realitzarCerca(formulari.getPais(),
            formulari.getProvincia(), formulari.getPoblacio(), formulari.getTipusImmoble(),
            formulari.getEstatImmoble(), m2Min, m2Max, formulari.getNumHab(), preuMin, preuMax);
    session.setAttribute("resultatCerca", resultat);

    SUCCESS="buscarOK";
}
```

Imatge 18. Classe CercarAction

El Controlador, és l'encarregat de delegar la feina al Manager del *Bean* del Domini corresponent. Això és degut al patró capes, i al patró baix acoblament, que el que pretenen es que els *Beans* del Domini no interactuïn directament amb el controlador, de igual manera passa amb la persistència (ho veurem més endavant). En el nostre cas, com estem fent una cerca d'immobles, delega la feina al *ImmobleManager*:

```
public List realitzarCerca(int idPais, int idProvincia, int idPoblacio,
        int idTipusImmoble, int idEstatImmoble, int m2Min, int m2Max,
        int numHabitacions, float preuMin, float preuMax) throws SQLException{

    return immobleManager.realitzarCerca(idPais, idProvincia, idPoblacio,
            idTipusImmoble, idEstatImmoble, m2Min, m2Max, numHabitacions,
            preuMin, preuMax);

}
```

Imatge 19. Classe Controlador

El Manager del *Bean* del domini, és l'encarregat de realitzar la creació, o modificació del *Bean* al que correspongui, i a part, també és l'encarregat de cridar al DAOManager corresponent de la persistència en el cas que s'hagi de inserir, modificar, o consultar la base de dades.

En el nostre cas, per consultar els immobles existents en la base de dades amb els paràmetres que ha introduït l'usuari de l'aplicació, primer de tot haurem de buscar els immobles, després consultar si aquests immobles contenen fotos o vídeos. Com ja hem dit, l'*ImmobleManager* és qui delega la feina als diferents Managers de la persistència que són necessaris per retornar els immobles correctament creats de la llista demanada. Ho podem veure a continuació:

```
public List realitzarCerca(int idPais, int idProvincia, int idPoblacio,
        int idTipusImmoble, int idEstatImmoble, int m2Min, int m2Max,
        int numHabitacions, float preuMin, float preuMax) throws SQLException {
    List immobles = immobleDAO.realitzarCerca(idPais, idProvincia,
            idPoblacio, idTipusImmoble,
            idEstatImmoble, m2Min, m2Max,
            numHabitacions, preuMin, preuMax);
    Iterator it = immobles.iterator();
    while(it.hasNext()){
       Immoble i=(Immoble) it.next();
       i.setPoblacio(poblacioManager.getPoblacioByIdImmoble(i.getIdImmoble()));
       i.setFotos(fotoManager.selectFotosByImmoble(i.getIdImmoble()));
       i.setVideo(videoManager.selectVideoByImmoble(i.getIdImmoble()));
   }
   return immobles;
}
```

Imatge 20. Classe ImmobleManager

En aquesta imatge, observem com primer es crida al *immobleDAO* (instància de la classe *DAOImmobleManager*), i després, amb la llista retornada per l'*immobleDAO*, s'itera aquesta llista per modificar l'immoble i afegir-hi les seves fotografies i el seu vídeo.

En aquesta explicació, ens centrarem en com es recuperen els immobles de la base de dades, no de com se li afegeixen les fotos i el vídeo, ja que seria repetir el procés per a cada un, amb les classes pertinents.

Com ja hem dit, per buscar els immobles cridem a una funció de la instancia de la classe (*DAOImmobleManager*), que fent servir el patró DAO, i el propi framework DAO de iBatis, declararà un atribut de la interfície corresponent, i instanciarà en aquest atribut un objecte de la classe que implementi aquella interfície. Delegarà llavors la feina a aquesta última classe, que és l'encarregada de la comunicació amb la base de dades:

```
public List realitzarCerca(int idPais, int idProvincia, int idPoblacio,
        int idTipusImmoble, int idEstatImmoble, int m2Min, int m2Max,
        int numHabitacions, float preuMin, float preuMax) {

   return immobleDAO.realitzarCerca(idPais, idProvincia, idPoblacio,
           idTipusImmoble, idEstatImmoble, m2Min, m2Max, numHabitacions,
           preuMin, preuMax);

}
```

Imatge 21. Classe DAOImmobleManager

Veiem ara el codi d'aquesta última funció que hem cridat, en la classe *ImmobleIbatisDAOImpl*, que implementa la interfície *ImmobleIbatisDAO*:

```
public List realitzarCerca(int idPais, int idProvincia, int idPoblacio,
        int idTipusImmoble, int idEstatImmoble, int m2Min, int m2Max,
        int numHabitacions, float preuMin, float preuMax) {
    /*DECLAREM EL HASHMAP QUE LI PASSAREM A LA QUERY, I AFEGIM AQUELLS VALORS
     QUE NO HEM DE TRACTAR, JA QUE ENS ARRIVEN JA BÉ DESDEL ACTION DE LA
     PRESENTACIÓ*/
    HashMap hm = new HashMap();
    hm.put("m2Min", m2Min);
    hm.put("m2Max", m2Max);
    hm.put("preuMin", preuMin);
    hm.put("preuMax", preuMax);
    /*EN ELS SEGÜENTS IF's, TRACTEM LA INFORMACIÓ QUE QUEDA, DE MANERA
     QUE SI EL PARÀMETRE QUE ENS ARRIVA ÉS UN 0, ÉS QUE L'USUARI
     NO TÉ EN COMPTE AQUELL PARÀMETRE PER REALITZAR LA CERCA; LLAVORS,
     SUBSTITUÏM EL 0 PER UN NULL*/
    if(idPais==0){
        hm.put("pais", null);
    }else{
        hm.put("pais", idPais);
    }
    if(idProvincia==0){
        hm.put("provincia", null);
    }else{
    hm.put("provincia", idProvincia);
    }
    if(idPoblacio==0){
        hm.put("poblacio", null);
    }else{
    hm.put("poblacio", idPoblacio);
    }

    if(idTipusImmoble==0){
        hm.put("tipusImmoble", null);
    }else{
    hm.put("tipusImmoble", idTipusImmoble);
    }
    if(idEstatImmoble==0){
        hm.put("estatImmoble", null);
    }else{
    hm.put("estatImmoble", idEstatImmoble);
    }
    if(numHabitacions==0){
        hm.put("numHabitacions", null);
    }else{
    hm.put("numHabitacions", numHabitacions);
    }
    /* EXECUTEM LA CERCA, CRIDAN A LA QUERY DEL ImmobleSqlMap.XML AMB L'ID
     CORRESPONENT*/
    return queryForList("immoble.realitzarCerca", hm);
}
```

Imatge 22. Classe ImmobleIbatisDAOImpl

Aquesta funció, el que fa és acabar de preparar les dades per poder realitzar la consulta. En el nostre cas, hem de canviar tots aquells valors que son zeros, per *nulls* (seguidament es veurà el perquè), ja que són els paràmetres que l'usuari no ha tingut en compte en definir la cerca. Un cop preparat el *HashMap* que li pasarem a la *query* dins del *ImmobleSqlMap*, cridem la funció *queryForList*, i com a primer paràmetre li hem d'especificar a quin *SqlMap* ho ha de buscar, i amb quin identificador de consulta "immoble" es l'*SqlMap*, i "*realitzarCerca*" és l'identificador.

Observem ara el codi de la consulta amb l'id realitzarCerca del *ImmobleSqlMap*, que és el que connecta amb la base de dades i realitza la consulta.

```
<select id="realitzarCerca" parameterClass="HashMap" resultMap="ImmobleResult">
    <!--FEM EL SELECT DE LES COLUMNES NECESSARIES DEFINIDES EN EL IMMOBLERESULT-->
          SELECT IMM.IDIMMOBLE, IMM.ADRECA, IMM.COORDENADAX, IMM.COORDENADAY,
          IMM.M2, IMM.NUMHABITACIONS, IMM.DESCRIPCIO, IMM.PREU,
          IMM.DATAALTAIMMOBLE, IMM.DATABAIXAIMMOBLE, IMM.VENUT,
          IMM.TIPUSIMMOBLE, IMM.ESTATIMMOBLE, IMM.POBLACIO
    <!--UNIM LES TAULES NECESSARIES PER DEFINIR LA CERCA DEL USUARI-->
          FROM IMMOBLE IMM INNER JOIN POBLACIO POB ON IMM.POBLACIO=POB.IDPOBLACIO
          INNER JOIN PROVINCIA PRO ON POB.PROVINCIA=PRO.IDPROVINCIA
          INNER JOIN PAIS ON PRO.PAIS=PAIS.IDPAIS
    <!--ACOTEM LA CONSULTA AMB ELS PARAMETRES QUE SEGUR TINDREM-->
          WHERE IMM.PREU <![CDATA[ > ]]> #preuMin# AND IMM.PREU <![CDATA[ < ]]> #preuMax#
          AND IMM.M2 <![CDATA[ > ]]> #m2Min# AND IMM.M2 <![CDATA[ < ]]> #m2Max#
          AND IMM.VENUT='FALSE'
    <!--AQUESTS SÓN ELS PARAMETRES VARIABLES, PER TANT EM CREAT RESTRICCIONS DE
    CERCA DINÀMIQUES. AMB EL TAG ISNOTNULL EL QUE CONSEGUIM, ES QUE SI EL
    PARÀMETRE DEL HASHMAP CORRESPONENT AMB LA PROPERTY DEFINIDA ES NULL, NO S'INCLOGUI
    EN LA SELECT, DEL CONTRARI, S'INCLOU AMB EL VALOR CORRESPONENT DEL HASHMAP
    I SE LI POSA AL DAVANT LA CLAU DEFINIDA EN EL PREPEND-->
    <isNotNull property="pais" prepend="AND"> PAIS.IDPAIS=#pais#</isNotNull>
    <isNotNull property="provincia" prepend="AND" > PRO.IDPROVINCIA=#provincia#</isNotNull>
    <isNotNull property="poblacio" prepend="AND"> POB.IDPOBLACIO=#poblacio#</isNotNull>
    <isNotNull property="tipusImmoble" prepend="AND"> IMM.TIPUSIMMOBLE=#tipusImmoble#</isNotNull>
    <isNotNull property="estatImmoble" prepend="AND"> IMM.ESTATIMMOBLE=#estatImmoble#</isNotNull>
    <isNotNull property="numHabitacions" prepend="AND"> IMM.NUMHABITACIONS=#numHabitacions#</isNotNull>
</select>
```

Imatge 23. ImmobleSqlMap.xml

Aquesta consulta com podem observar, és una consulta dinàmica. Les clàusules *where* s'afegeixen només si el paràmetre del *HashMap* entrat no és *null*, i si s'afegeix el *where*, es posa al davant al condició que hi hagi en el *prepend*. D'aquesta manera podem estalviar-

nos la repetició de codi, és a dir, no fer tantes consultes com criteris i combinacions de cerca hi ha en el formulari que hem vist en la JSP de la presentació.

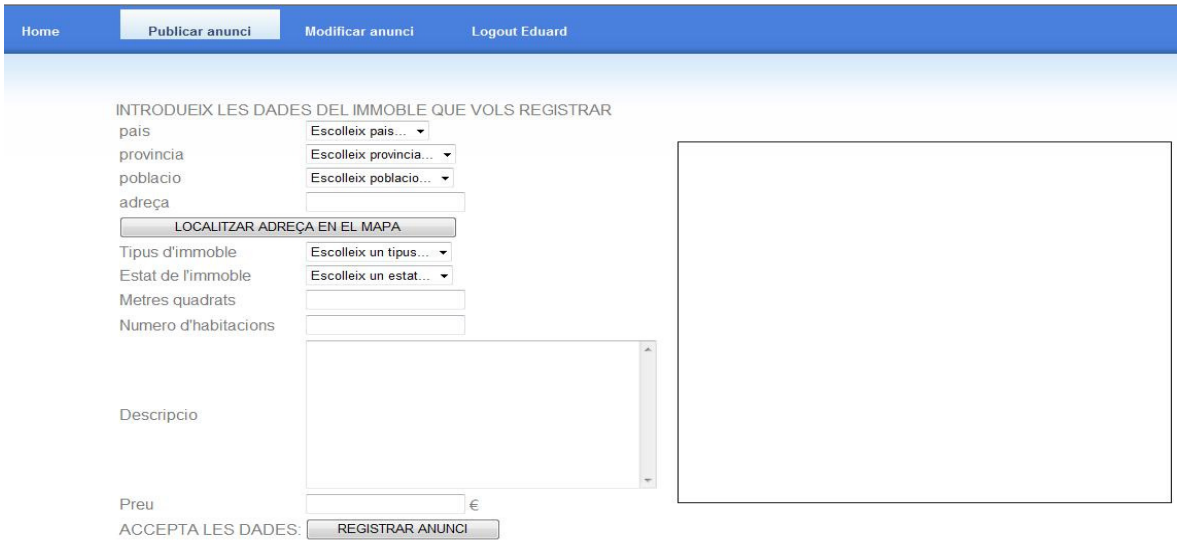
8.3. Implementació Registrar anunci

Aquí definirem la línea d'execució del cas d'ús Registrar anunci. Per tal de no repetir, i de no explicar obvietats, ens centrarem en el codi de com es crea el mapa de Google Maps i en com actualitzem la informació d'aquest. Per poder realitzar aquesta acció, l'usuari haurà d'haver realitzat amb èxit el cas d'ús Log In.

Per registrar l'anunci, l'usuari haurà de cridar l'acció des del botó del menú Publicar anunci, en la següent imatge podem apreciar el botó mencionat:

Imatge 24. Menú usuari registrat

Un cop haguem pitjat el botó, es cridarà a la JSP que conté el formulari per a registrar un nou anunci, és el següent:



Imatge 25. Formulari registrar immoble

L'usuari escollirà un país, aleshores, al haver canviat el valor de la llista de països, es cridarà a una funció javascript que el que farà serà omplir la llista de províncies, pertanyents al país seleccionat, i de igual manera s'omple la llista de poblacions, al escollir la província, es crida al javascript i la omple.

Ara bé, quan escollim una població, aleshores inicialitzem el mapa i l'aproximem a la població seleccionada. Per veure com es realitza aquest procés, veurem el codi del formulari de la JSP i a la funció del javascript que crida el fet de seleccionar una població:

```
<!--QUANT ELEGIM UNA POBLACIÓ, ES CRIDA A LA FUNCIÓ LOADMAPA()
DEL JAVASCRIPT GENERIC.JS-->
<select id="poblacio" name="poblacio"  onchange="loadMapa()">
    <option value="0">Escolleix poblacio...</option>
    <c:forEach items="${comboBean.llistaPoblacions}" var="poblacio">
        <option value="${poblacio.idPoblacio}">
            ${poblacio.nomPoblacio}
        </option>
    </c:forEach>
</select>
```

Imatge 26. Codi pàgina JSP

Veiem que quan es canvia el valor de la població, cridem el loadMapa(), que està ubicat en l'arxiu Generic.js:

```
function loadMapa(){
    //COMPROVEM QUE L'USUARI HA ESCOLLIT UNA POBLACIO
    if(document.getElementById("poblacio")=='0'){
        alert("Has d'escollir una poblacio")
    }else{

        init(); //INICIALITZEM EL MAPA
        primerCop=true;
        //A CONTINUACIO CRIDEM EL METODE GETNOMPOBLACIO, QUE MITJANÇANT AJAX
        //CENTRARÀ EL MAPA EN LA POBLACIÓ ESCOLLIDA
        getNomPoblacio(document.getElementById("poblacio").value);
    }
}
```

Imatge 27. Funció javascript loadMapa()

Primer de tot, mirem que l'usuari hagi escollit una població correctament, i no ha tornat al valor per defecte, que es el zero. Seguidament, cridem la funció *init*(), que el que fa és crear un mapa nou, després centra el mapa en les coordenades d'Espanya, modifica al zoom a 5, i crea les opcions de interactuar amb l'usuari per defecte.

Ho podem veure a continuació:

```
function init() {
    //ES MIRA QUE EL NAVEGADOR ES COMPATIBLE
    if (GBrowserIsCompatible()) {
        //ES CREA L'INSTANCIA EL MAPA I S'INSERTA EN LA DIV CORESPONENT
        map = new GMap2(document.getElementById("mapa"));
        //CENTREM EL MAPA A LES COORDENADES D'ESPANYA
        map.setCenter(new GLatLng(40.463667, -3.74922));
        //DEFINIM EL ZOOM PER DEFECTE A 5
        map.setZoom(5);
        //S'INICIALITZEN EN EL MAPA ELS CONTROL PREDEFINITS
        map.setUIToDefault();
        //COMPROVEM QUE LA PAGINA JSP QUE CRIDA L'ACCIO ES LA DE REGISTRAR'
        if(document.getElementById("pagina").value=="registrar"){
            //AMB AQUESTA FUNCIÓ, EL SISTEMA RECONEIXERÀ SI L'USUARI FA CLIC
            //SOBRE EL MAPA
            GEvent.addListener(map, "click", function(overlay, latlng) {
                if (! overlay) {
                    //GUARDEM EN L'INPUT DE LA JSP EL VALOR DE LAS COORDENADES
                    document.getElementById("coordenades").value = latlng.toUrlValue(6);
                    //MOSTRA LES COORDENADES ON HA FET CLIC EN EL MAPA
                    reverseGeocode(latlng);
                }
            });
        }
    }
}
```

Imatge 28. Funció javascript init()

La funció *getNomPoblacio*(), li passem com a paràmetre l'identificador de la població escollida en el formulari per l'usuari, aleshores s'executa una petició d'AJAX que consulta a la base de dades el nom d'aquesta població escollida, i un cop retornat el nom, cridem a la funció *forwardGeocode*(), passant-li com a paràmetre el nom retornat.

Podem veure el *forwardGeocode*() a continuació:

```
function forwardGeocode(address) {
    //INICIALITZA EL GCLIENTGEOCODER, QUE ÉS NECESSARI PER CONSULTAR LA BASE
    //DE DADES DE GOOGLE
    var geocoder = initGeocoder(address);
    //UN COP TENIM EL CLIENT, EXECUTEM LA CERCA
    geocoder.getLocations(address, function(response) {
        //UN COP REBEM LA RESPOSTA, LA MOSTREM EN EL MAPA
        showResponse(response, false);
    });
}
```

Imatge 29. Funció javascript forwardGeocode()

El *showResponse*(), comprova que la resposta ha tingut èxit, i en el cas de que sigui correcte, procedeix a mostrar la informació rebuda en el mapa mitjançant la funció *plotMatchesOnMap*(). Veiem ara el codi de *plotMatchesOnMap*():

```
function plotMatchesOnMap(response, reverse) {
    //INICIALITZEM LES VARIABLES
    var resultCount = 1;
    details   = new Array(resultCount);
    markers   = new Array(resultCount);
    var icons   = new Array(resultCount);
    var latlngs = new Array(resultCount);
    for (var i = 0; i < resultCount; i++) {
        //CREEM UN ICONA
        icons[i] = new GIcon(G_DEFAULT_ICON);
        //AFEGIM L'ADREÇA ON ES TROBA L'IMATGE DE L'ICONA
        icons[i].image = MAPFILES_URL + "marker" +
            String.fromCharCode(65 + i) + ".png";
        //GUARDEM LES COORDENADES DE LA RESPOSTA
        var coord = new GLatLng(response.Placemark[i].Point.coordinates[1],
            response.Placemark[i].Point.coordinates[0]);
        latlngs[i] = coord;
        //COMPROVEM QUE LA PAGINA JSP QUE CRIDA L'ACCIO ES REGISTRAR ANUNCI
        if(document.getElementById("pagina").value=="registrar"){
            //COMPROVEM QUE S'HA MODIFICAT LES COORDENADES DEL MAPA
            var coordAntigues=document.getElementById("coordenades").value;
            var coordNoves=coord.toUrlValue(6);
            if(coordAntigues==coordNoves){
                alert("L'ADREÇA DONADA NO S'HA TROBAT, BUSCA EN EL MAPA"+
                        " I FES CLIC SOBRE ELL PER ACCEPTAR");
            }

        else{
            //SI LES COORDENADES SÓN DIFERENTS A LES GENERADES ANTERIORMENT
            //LES GUARDEM COM A LA LOCALITZACIÓ DE L'ADREÇA
            document.getElementById("coordenades").value=coord.toUrlValue(6);
        }
    }
    //OMPLIM LA INFORMACIÓ DEL MARCADOR AMB UNA FUNCIÓ AUXILIAR QUE ENS RETORNA
    //LES DADES INTRODUIDES PER L'USUARI'
    details[i] = getAdrecaDetailHTML();
    //CREEM EL MARCADOR AMB LES COORDENADES I LA IMATGE DE LA ICONA
    markers[i] = new GMarker(latlngs[i], {
        icon: icons[i]
    } );

}
//AFEGIM EL MARCADOR AL MAPA
for (var i = 0; i < resultCount; i++) {
    map.addOverlay(markers[i]);
    //AFEGIM UN LISTENER, PER SI ES FA CLIC SOBRE EL MARCADOR,
    //ES MOSTRI LA INFORMACIÓ QUE CONTÉ
    addInfoWindowListener(i, markers[i], details[i]);
}
```

```
    if(primerCop){
        map.setCenter(latlngs[0],15);
        primerCop=false;
    }if(mapaDetallat){
        map.setCenter(latlngs[0],16);
        mapaDetallat=false;
    }else{
        map.setCenter(latlngs[0]);
    }
    //EN EL CAS QUE ES CRIDI DESDE LA JSP VEURE DETALL, S'OBRIRÀ
    //LA INFORMACIÓ DEL MARCADOR AUTOMATICAMENT
    if(document.getElementById("pagina").value=="immobleDetallat"){
        GEvent.trigger(markers[0], "click");
    }
}
```

Imatge 30. Funció javascript plotMatchesOnMap()

En aquesta funció, primer de tot s'inicialitzen les variables, seguidament creem la icona que es mostrarà en el mapa, amb la seva imatge corresponent, que aquesta imatge s'agafa del servidor de Google. Després guardem les coordenades de la resposta i també en *l'input* corresponent de la JSP. Un cop realitzat això, creem el marcador amb la imatge anterior, i li assignem les coordenades. Després, omplim el detall del marcador amb una funció auxiliar que hem creat, que ens retorna una taula *html* amb la informació de la població, l'adreça, i l'usuari al que pertany. Afegim llavors el marcador en el mapa i creem un *listener* que la seva funció serà la de visualitzar la informació continguda en el detall del marcador quan es faci clic sobre ell.

D'aquesta manera ja hem actualitzat la informació del mapa centrant-la en la població entrada. Ara bé, quan l'usuari ompli el camp de l'adreça i premi el botó Localitzar adreça en el mapa, aleshores es cridarà la següent funció, que reutilitza el codi anteriorment descrit:

```
function localitzarAdreca(){
    var html=document.getElementById("adreca").value+", "+
        document.getElementById("nomPoblacio").value;
    forwardGeocode(html);
}
```

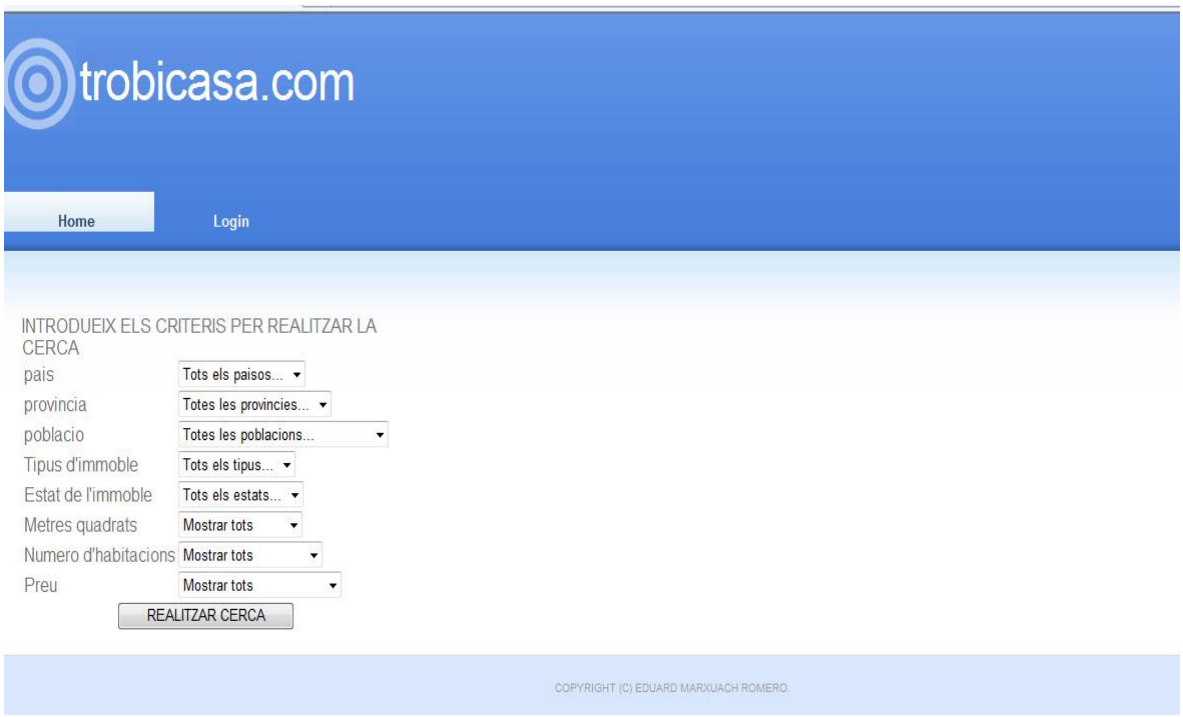
Imatge 31. Funció javascript localitzarAdreca()

Com veiem, *localitzarAdreca()*, un cop obté l'adreça entrada per l'usuari, torna a cridar a la funció *forwardGeocode*() per actualitzar de nou el mapa, ara centrant-lo en l'adreça.

Un cop realitzat això, l'usuari acaba d'introduir els paràmetres que falten del formulari, i prem el botó registrar anunci, d'aquesta manera s'inserirà en la base de dades.

9. Prova significativa

En l'índex podem apreciar el botó que ens portarà a la pàgina d'entrada al sistema, i també un altre botó que ens retorna al mateix índex. També observem el formulari de cerca, amb els diferents criteris amb els que podem buscar:



Imatge 32. Pàgina índex

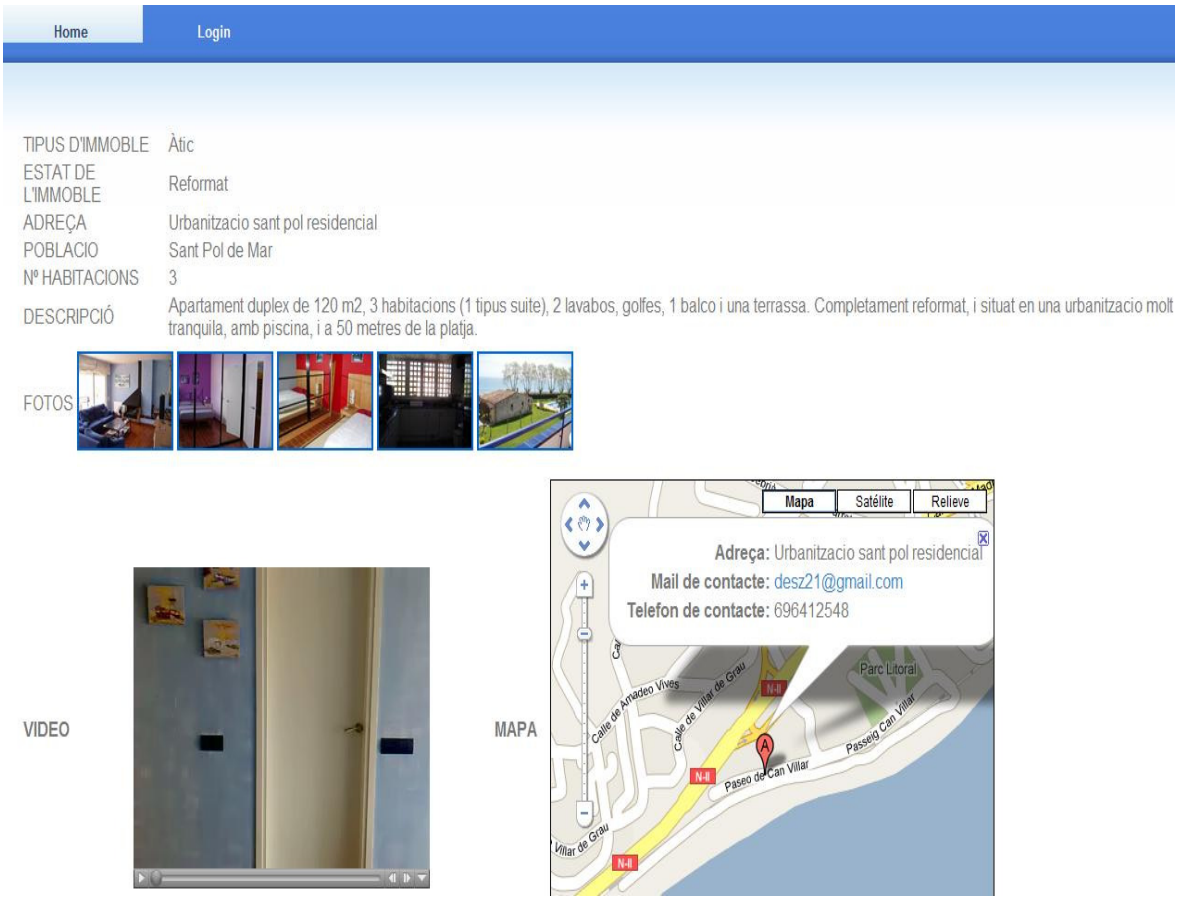
Per realitzar una cerca acotada, omplim els camps que vulguem, per realitzar aquesta prova, buscarem anuncis en la província de Barcelona, per tant, seleccionem el país, i la província, i premem el botó Realitzar Cerca:

Imatge 33. Exemple cerca

Després de prémer el botó, se'ns mostrarà la taula dels resultat obtinguts, podent pitjar el botó veure detall del anunci que ens interessi més:

Imatge 34. Resultat de la cerca

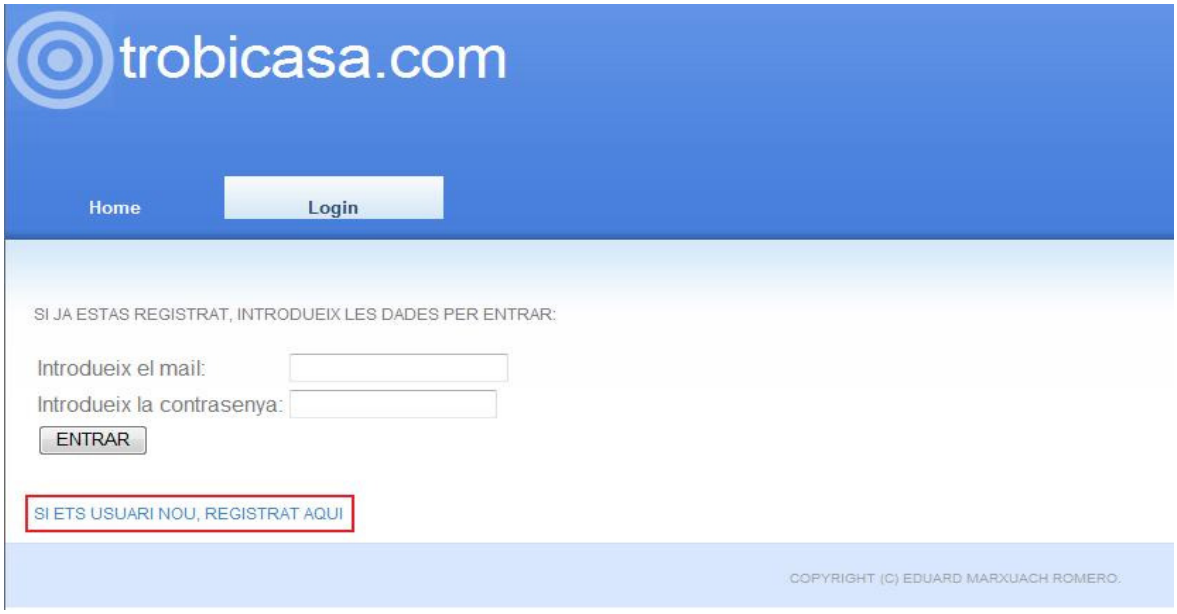
Quan escollim veure el detall d'un anunci, se'ns obrirà aquesta pàgina:



Imatge 35. Exemple anunci detallat

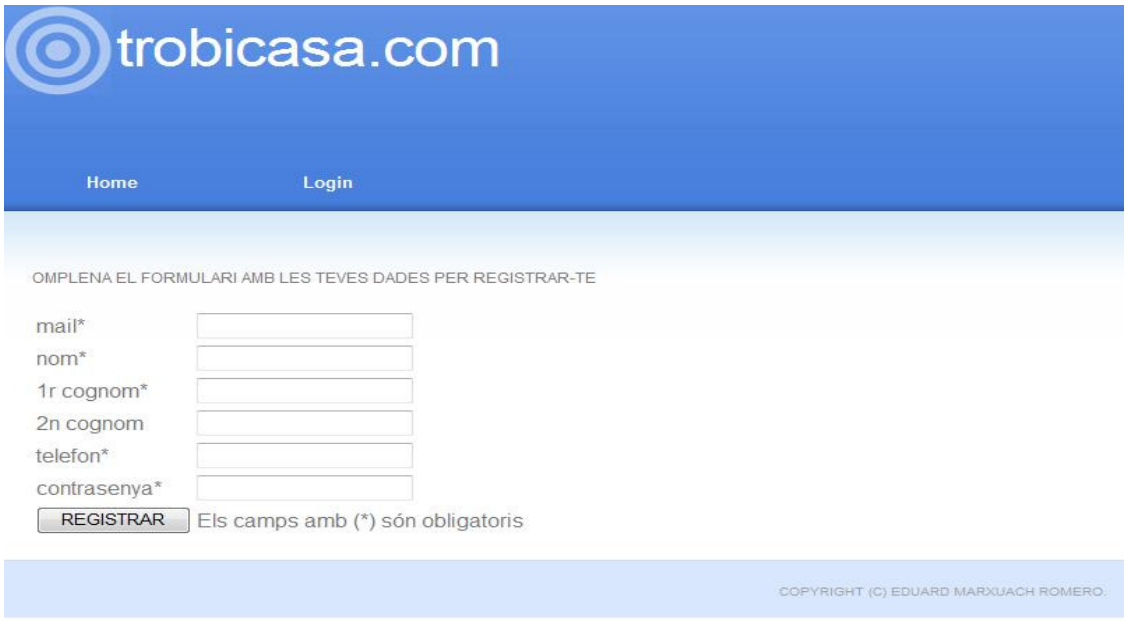
En la pàgina, es mostra el detall del immoble, així com les fotografies i el vídeo, i també el mapa de la localització, amb l'adreça, el telèfon del propietari i el seu mail.

Ara bé, si no volem buscar cap propietat, i volem pujar un anunci per vendre nosaltres un immoble, el que hem de fer és prémer el botó de Login, per autenticar-nos, o donar-nos d'alta. La pàgina on col·locar el mail i la contrasenya és la següent:



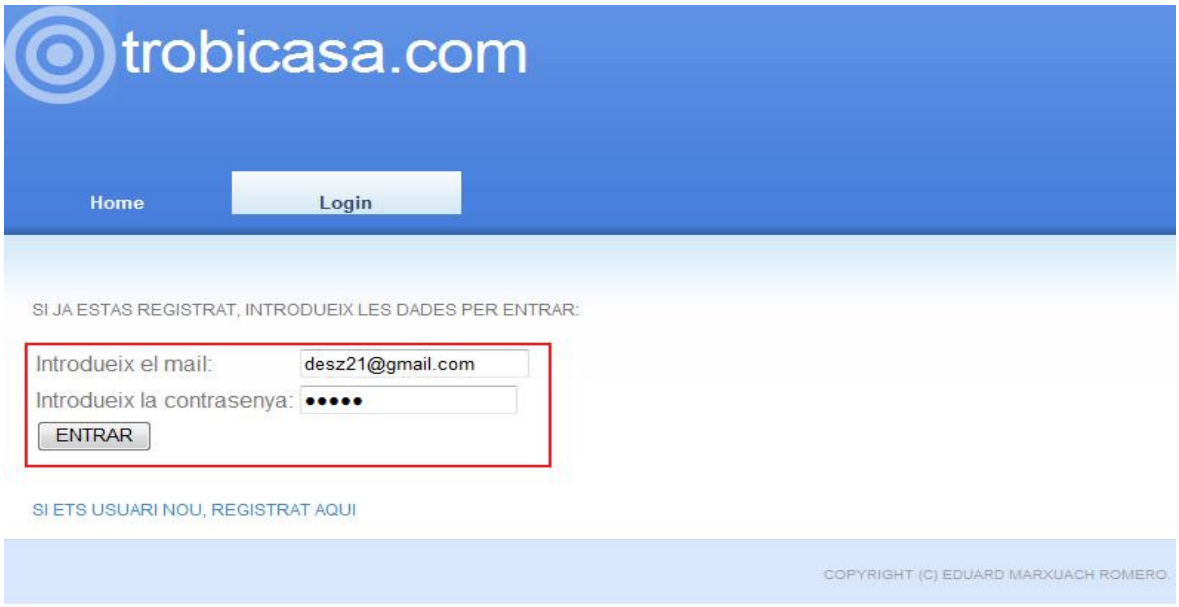
Imatge 36. Link per registrar-se

En el cas que ens tinguéssim que donar d'alta, premeríem les lletres blaves de la part inferior esquerra, i aniríem al formulari de registre:



Imatge 37. Formulari de registre

Si ja estem donats d'alta, introduïm el nostre mail i contrasenya, i premem el botó d'entrar:

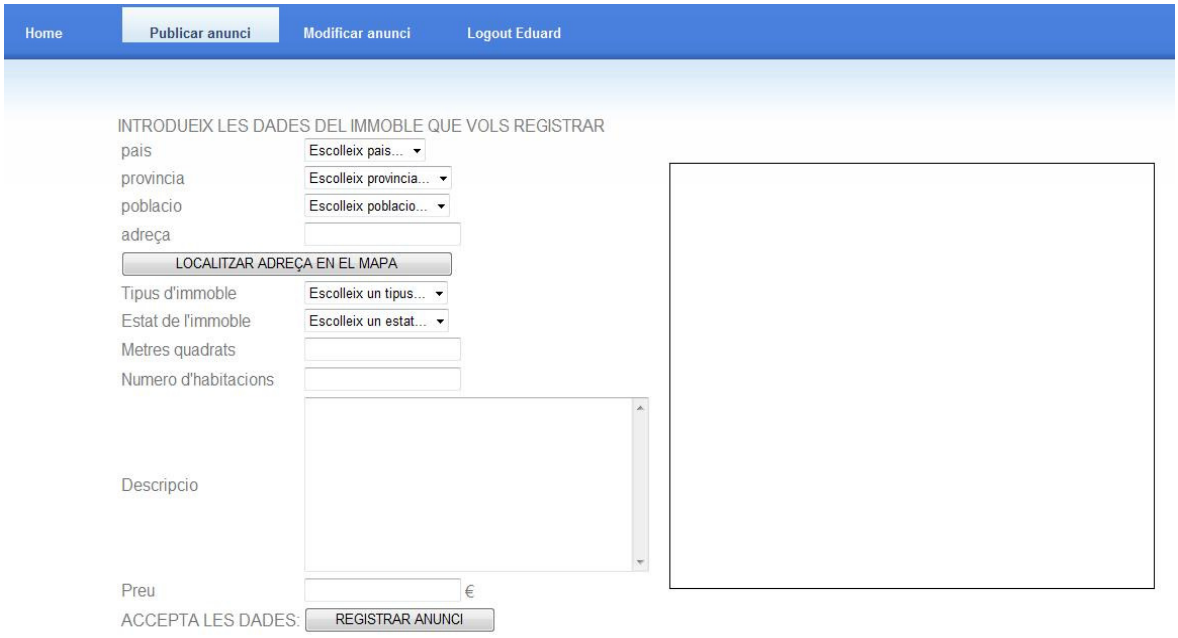


Imatge 38. Formulari per entrar

Un cop acceptats pel Sistema, ens retorna a la pàgina d'inici, i ja podem pitjar el botó de Publicar Anunci:

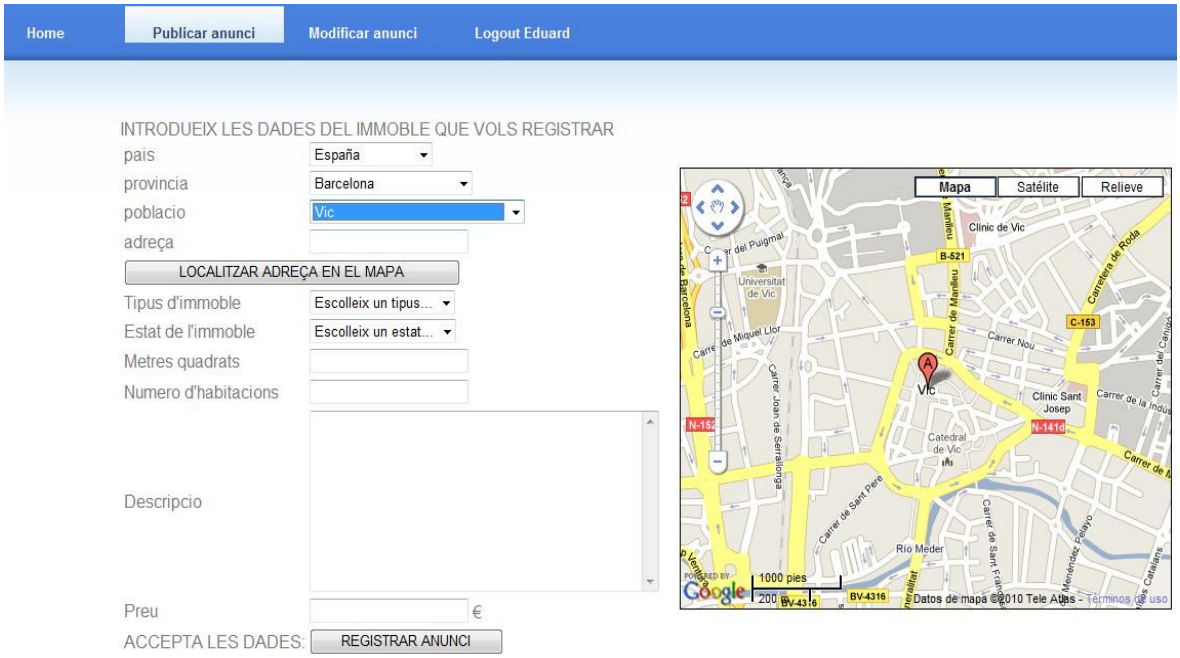
Imatge 39. Botó publicar anunci

Se'ns apareixerà el següent formulari que haurem d'omplir per tal de registrar l'anunci:



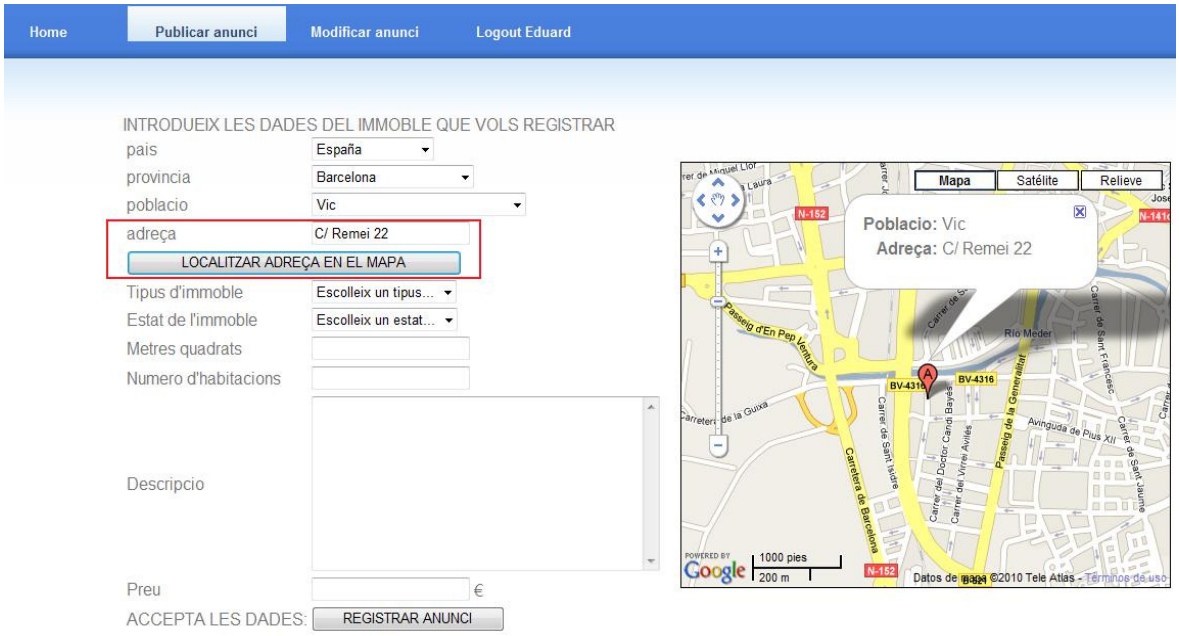
Imatge 40. Formulari per publicar anunci

Primer de tot, escollim el país (Espanya), després la província (Barcelona) i la població (Vic). Un cop haguem escollit la població, se'ns mostrarà el mapa de la població escollida:



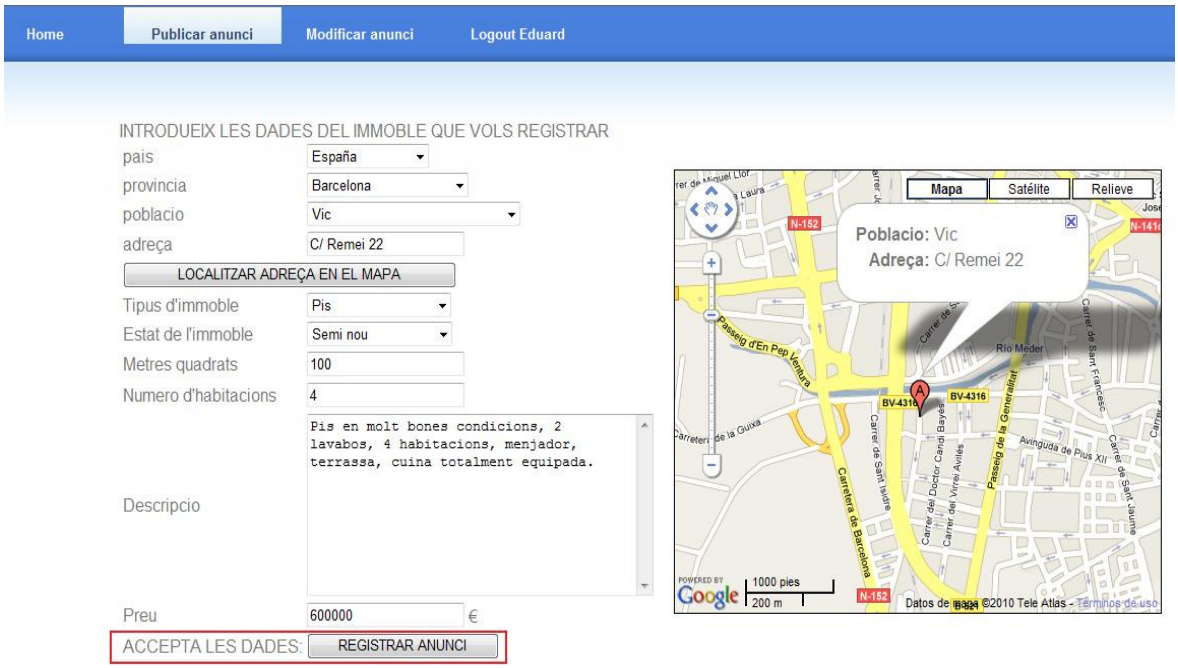
Imatge 41. Aproximació en el mapa de la població

Ara ve quan escrivim l'adreça, i pitgem el botó Localitzar adreça en el mapa:



Imatge 42. Aproximació en el mapa de l'adreça

Un cop fet això, omplim la resta del formulari i premem el botó Registrar Anunci:

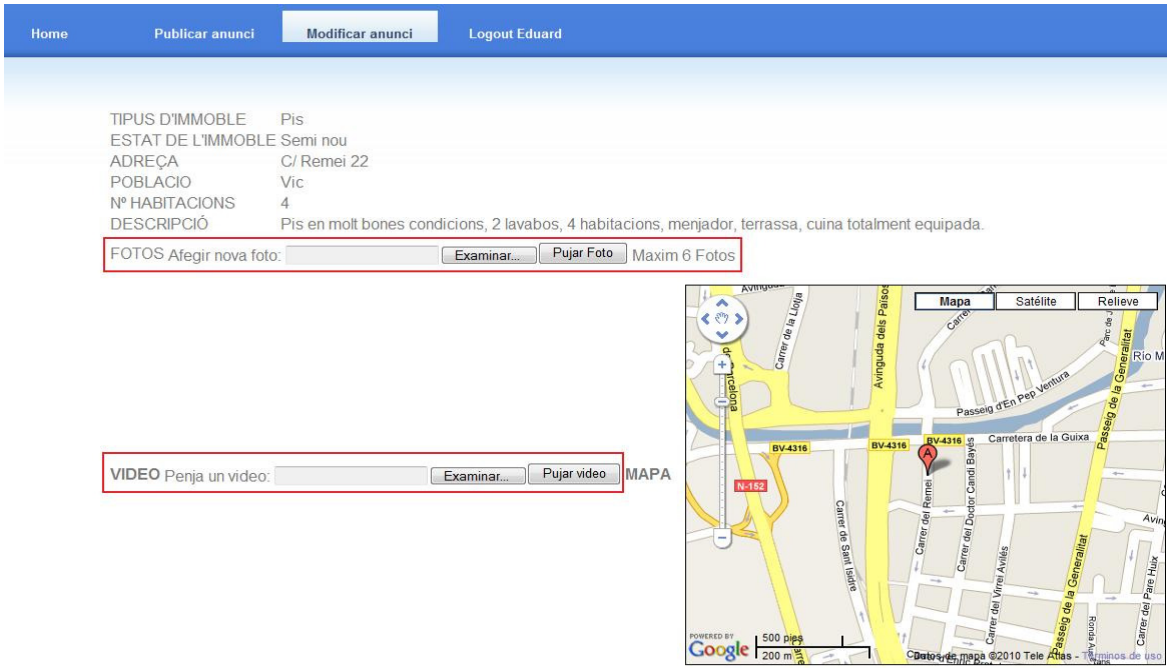


Imatge 43. Botó registrar anunci

D'aquesta manera, l'anunci ja serà visible per als usuaris que generin una cerca. Si ara el que volem fer és el nostre anunci més atractiu, posant-hi fotografies o un vídeo, hem de prémer el botó de Modificar immoble, i ens sortirà la següent pantalla:

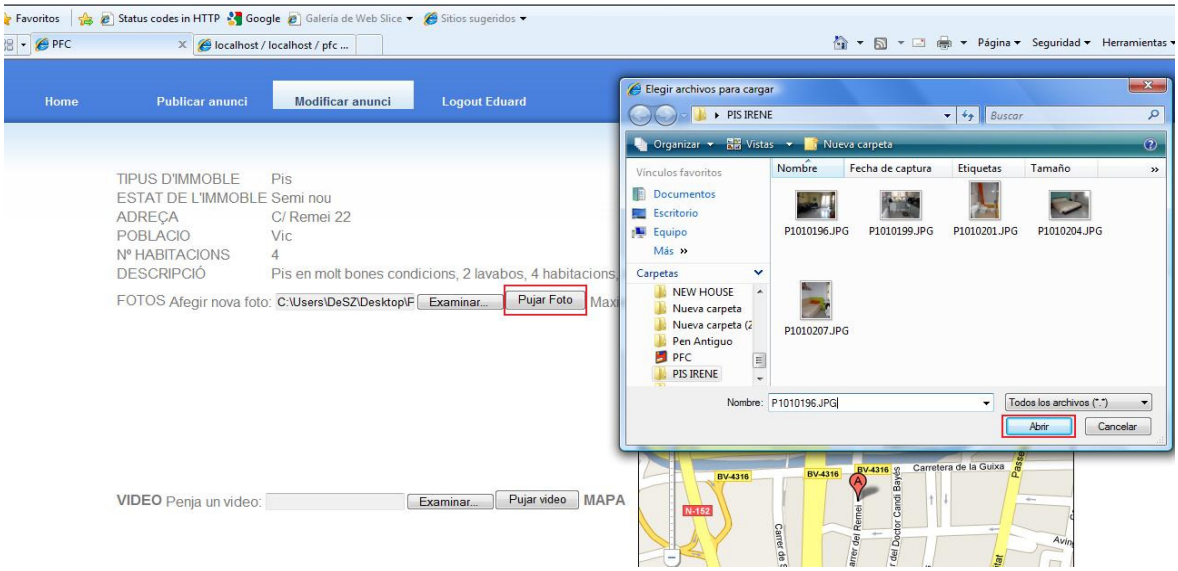
Imatge 44. Botó afegir foto/vídeo

Ara penjarem una fotografia a l'anunci que acabem de publicar, per fer-ho, premem el botó de Afegir foto/vídeo, i ens mostrarà la següent pàgina:



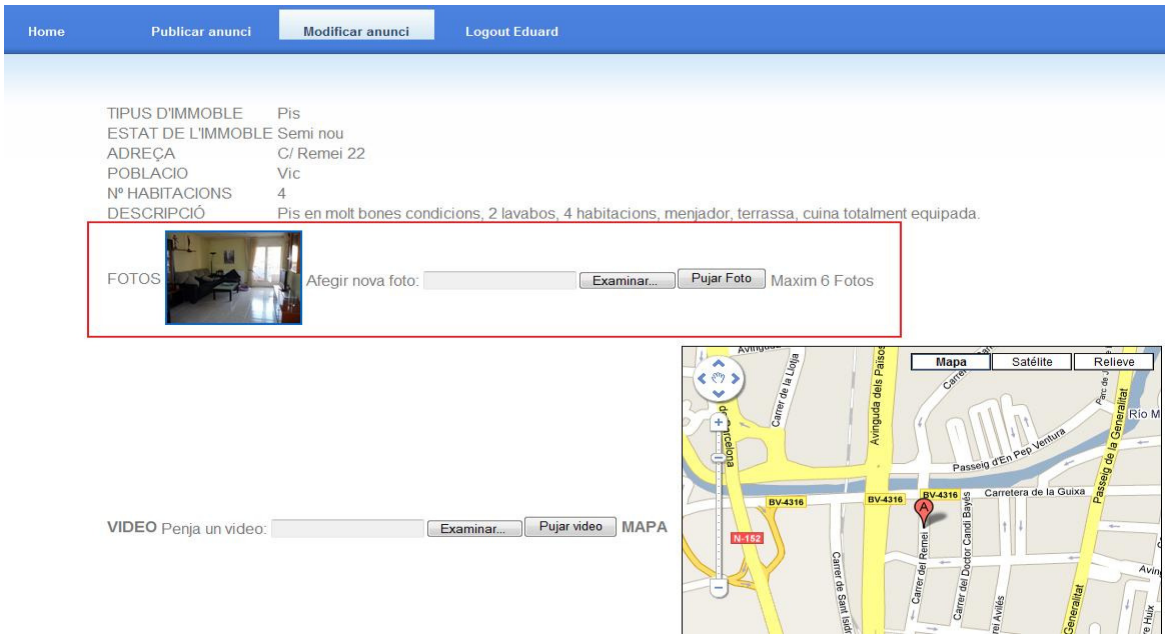
Imatge 45. Pàgina per publicar una foto o un vídeo

La mateixa pàgina ens permet pujar les fotografies i el vídeo. Per penjar una fotografia, només hem de prémer el botó examinar, localitzar la ubicació de la foto, i prémer el botó Pujar Foto. Per pujar el vídeo és pràcticament igual, premem el botó examinar, i un cop localitzat, pitgem Pujar Vídeo. Ara ens disposem a penjar una fotografia:



Imatge 46. Exemple pujar foto

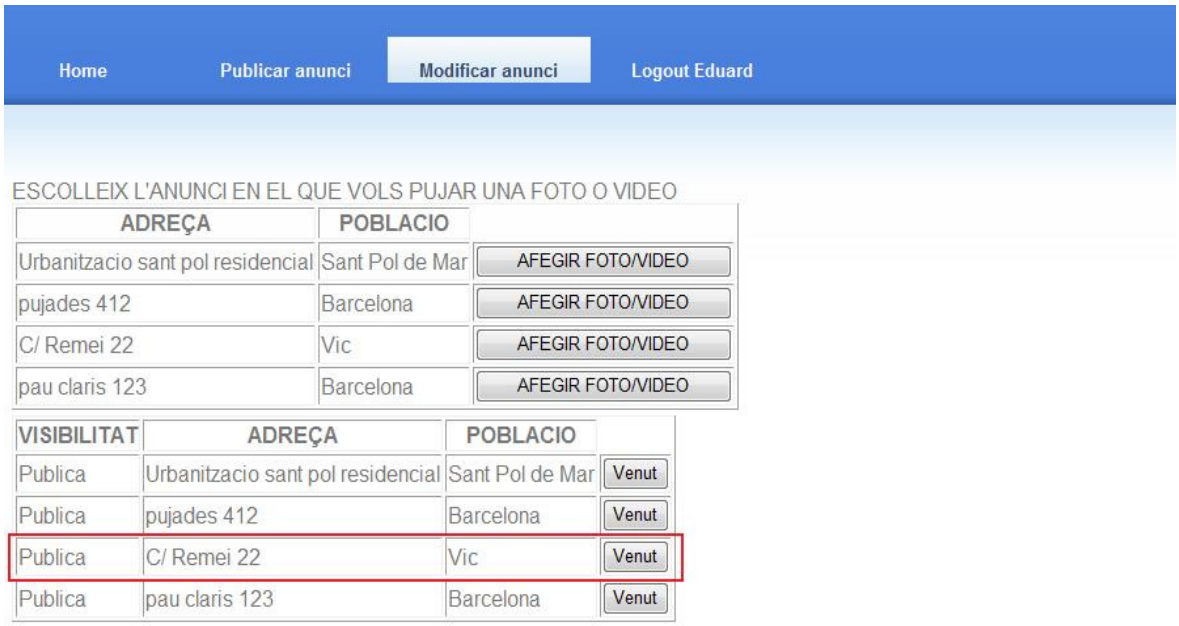
Un cop haguem pitjat el botó de Pujar Foto, el resultat serà la mateixa pàgina que estàvem veient, però amb la foto actualitzada, d'aquesta manera podrem penjar més fotografies si volem:



Imatge 47. Pàgina resultant de pujar una foto

Per penjar el vídeo, com s'ha comentat seria el mateix procés, però omplint el camp del Vídeo.

Suposem que ara el nostre anunci porta penjat ja unes setmanes, i algú s'ha posat en contacte amb nosaltres perquè vol comprar la nostre propietat. Ens interessarà que els altres usuaris ja no puguin veure l'anunci, per fer això anem a la pantalla de Modificar Immoble, i veiem la visibilitat dels nostres anuncis. Només cal fer clic sobre el botó Venut del anunci que ens interessi canviar la visibilitat, en aquest exemple, el del Carrer Remei 22 de Vic:



Imatge 48. Botó per canviar la visibilitat

Un cop pitjat el botó, observem com a canviat la visibilitat:

Imatge 49. Resultat de canviar la visibilitat

Ara ja no és visible per cap altre usuari.

Per finalitzar, sortirem de l'aplicació ja que no volem realitzar cap més acció, això es fa amb el botó de Logout:

Imatge 50. Botó Logout

10. Conclusions

Un cop finalitzat el projecte podem considerar que hem assolit els objectius principals plantejats inicialment.

Una persona que tingui la necessitat de comprar una vivenda o immoble, mitjançant un cercador avançat pot veure els anuncis que encaixen amb les seves necessitats, tenint accés a la informació d'aquest anunci, i a les dades de contacte del seu anunciant, en el cas de que es vulgui posar en contacte amb ell.

També una persona que tingui un immoble o propietat i el vulgui vendre, tindrà la possibilitat d'anunciar-lo fàcilment. Per fer-ho, només s'haurà de registrar en la nostre aplicació web.

Jo personalment, he complert els meus objectius, que eren aprendre a utilitzar el framework iBatis, ja que es demana molt en el mon laboral, i guanyar experiència també amb els *struts*. Un valor afegit que m'enduc al haver realitzat aquest projecte, és haver aprés també a utilitzar el *Google Maps* per adaptar-lo a les meves necessitats, ja que és una eina gratuïta i de gran utilitat.

11. Treball futur

Un proper projecte, seria ampliar aquest de manera que s'inclogués dins d'una xarxa social, d'aquesta manera la comunicació entre interessats seria més eficient.

Un altre opció seria d'incloure també la opció de llogar, compartir, o el que s'ha posat de moda ara, intercanviar les propietats durant les èpoques de vacances.

12. Bibliografia

[1] http://es.wikipedia.org (Juny 2010)

[2] http://www.adictosaltrabajo.com/tutoriales/tutoriales.php?pagina=iBatis (Juny 2010)

[3] http://webstyleguide.com/wsg3/index.html (Juny 2010)

[4] http://ibatis.apache.org/docs/dotnet/datamapper/index.html (Juny 2010)

[5] http://gmaps-samples.googlecode.com/svn/trunk/geocoder/v2-geocoder-tool.html (Juny 2010)

[6] http://www.roseindia.net/struts/strutsfileuploadandsave.shtml (Juny 2010)

[7] http://www.huddletogether.com/projects/lightbox2/ (Juny 2010)

[8] http://www.davidbayon.net/index.php?mostrar=posts&post_id=91 (Juny 2010)

13. Annex I (Contingut del CD-ROM)

1. Carpeta documentació.
   - Primeres pàgines de la documentació.
   - Índexs.
   - Contingut de la memòria.
2. Carpeta Base de Dades.
   - BBDD.sql (Scripts per a la creació de la Base de Dades).
3. Carpeta aplicació.
   - Codi font.
   - Llibreries.
4. Software necessari.
   - NetBeans 6.7.1 o superior.
   - Xampp