Escola Universitària
Politècnica de Mataró

Centre adscrit a:

UNIVERSITAT POLITÈCNICA
DE CATALUNYA

Enginyeria Tècnica en Informàtica de Gestió

WEDUGU

Memòria

NÚRIA SILVA SERRANO
PONENT: JOSEP MARIA GABRIEL SOLANILLAS

PRIMAVERA 2011

TecnoCampus
Mataró-Maresme

Dedicatòria

Dedico aquest projecte a aquells que m'han fet costat durant tots aquests anys quan no veia el final del camí.

Gràcies especialment a Pedro, Carla, Victor i Edu per totes les batalles compartides.

Resum

L'objectiu del present projecte es la construcció d'una xarxa social capaç de proposar plans d'oci als usuaris en funció del seus gustos i de la seva localització en el moment d'ús de l'aplicació. La xarxa social disposa de dues aplicacions clients, una aplicació web disponible des de qualsevol navegador i una aplicació per a telèfons mòbils Android. Com a nexe d'unió de les dues aplicacions client s'ha establert una base de dades comú a la qual accediran les dues aplicacions. L'aplicació web s'ha implementat seguint el patró Model-Vista-Controlador i l'aplicació Android s'ha implementat seguint les guies de la documentació oficial d'Android. El resultat son dues aplicacions totalment funcionals.

Resumen

El objetivo del presente proyecto es la construcción de una red social capaz de proponer planes de ocio a sus usuarios en función de sus gustos y de su localización en el momento de uso de la aplicación. La red social dispone de dos aplicaciones cliente, una aplicación web disponible desde cualquier navegador y una aplicación para teléfonos móviles Android. Como nexo de unión de las dos aplicaciones cliente se ha establecido una base de datos común a la que acceden las dos aplicaciones. La aplicación web ha sido implementada siguiendo el patrón Modelo-Vista-Controlador y la aplicación Android ha sido implementada siguiendo les guías de la documentación oficial de Android. El resultado son dos aplicaciones totalmente funcionales.

Abstract

The objective of this project is to build a social network capable of offering its users entertainment plans based on their preferences and location at the time of application's use. The social network has two client applications, web application available from any browser and mobile application for Android smartphones. As the nexus of the two client applications has established a common database which will access two applications. The web application has been implemented following the Model-View-Controller and the Android application has been implemented following the guidelines and official documentation of Android. The result is two fully functional applications.

Índex.

Índex de figures.

Índex de taules.

Glossari de termes.

PHP	Hypertext Pre-processor
SQL	Llenguatge de consulta estructurat
MVC	Model Vista Controlador
JEE	Java Enterprise Edition
JSE	Java Standard Edition
URL	Uniform Resource Locator
XML	Extensible Markup Language
JSP	JavaServer Pages
HTML	Hypertext Markup Language
GNU	GNU is Not Unix
DOM	Document Object Model
AJAX	Asynchronous JavaScript And XML
GPL	General Public License
MIT	Massachusetts Institute of Technology
JSON	JavaScript Object Notation
CSS	Cascading Style Sheets
W3C	World Wide Web Consortium
API	Application programming interface
JPA	Java Persistence API1. Objectius

1. Objectius

1.1. Propòsit

Aquest projecte es el resultat de sumar dos desitjos: aprendre a desenvolupar amb la plataforma Android i realitzar una aplicació web que aporti experiència i faciliti la inserció al món professional. Després d'una primera avaluació, va sorgir la idea de fusionar les dues idees en un sol projecte i realitzar-lo en parella, de manera que un fes l'aplicació mòbil i l'altre l'aplicació web.

L'aplicació mòbil seria una maqueta il·lustrativa dels coneixements adquirits al llarg dels mesos de desenvolupament mentre que l'aplicació web hauria de tenir un volum considerable ja que el propòsit era sintetitzar i anar més enllà del que ja s'havia après durant els tres anys.

1.2. Objecte

L'objecte principal d'aquest projecte és respondre a l'eterna pregunta que sorgeix a gairebé totes les reunions d'amics: i ara què fem?

S'ha volgut desenvolupar una aplicació útil per l'usuari, que l'ajudés en el seu dia a dia, fent un ús intensiu d'una de les característiques que més exploten els smartphones, la geolocalització, que ajuda a l'usuari a conèixer més la ciutat, a obtenir informació just en el moment en que la necessita si es troba al carrer, o bé, a prendre decisions si l'usuari no té les idees del tot clares.

2. Introducció

2.1. Què és WeDugu?

WeDugu és una xarxa social basada en l'oferta d'esdeveniments d'oci. El seu objectiu es proposar als seus usuaris opcions atractives a partir del coneixement dels llocs a on acudeixen i dels seus gustos o rutines.

Gràcies a la localització geogràfica incorporada en els telèfons d'última generació (smartphones) i en els navegadors web més moderns, de manera fàcil es podrà saber on és l'usuari i en quina zona s'haurà de focalitzar la cerca.

La part social servirà perquè l'aplicació pugui conèixer l'entorn de l'usuari i així proposar-li opcions que hagin realitzat els seus amics i que potser l'interessen.

2.2. A qui va dirigit.

La xarxa social WeDugu va dirigida a tot tipus d'usuaris majors de 18 anys. Vol aprofitar el fet de que moltes vegades la gent no sap on anar o com buscar el millor pla per divertir-se. Aquest tipus de gent perd el temps pensant i pensant, decidint amb la seva parella o amics que poden fer aquella tarda o nit, WeDugu els facilitarà una llista de propostes de coses a fer, a prop d'ells, en tant sols uns segons i només amb el requeriment d'estar enregistrat i haver omplert un perfil indicant els seus gustos.

2.3. Disseny funcional.

2.3.1. Punt de partida.

El punt de partida serà la creació d'un perfil d'usuari on s'aporten unes dades bàsiques referents a preferències i professió que permetin a l'aplicació fer-se una idea de quin esdeveniment podria acceptar l'usuari.

2.3.2. Utilització del producte.

Una vegada obtingut el perfil es podria fer ús de l'aplicació mitjançant dos suports diferents:

- Aplicació per dispositius mòbils amb sistema operatiu Android.

- Aplicació Web.

2.3.3. Serveis.

WeDugu permetrà realitzar les següents accions:

- Gestió d'amistat: L'usuari tindrà l'opció d'afegir a la seva llista d'amics a altres usuaris amb el que tingui algun punt en comú.
- Propostes d'esdeveniments: En funció del perfil que s'anirà creant gracies a l'ús, el sistema escollirà esdeveniments disponibles que interessin a l'usuari, tenint en compte la seva situació geogràfica. Si a l'usuari l'interessa alguna de les opcions, l'acceptarà i aquesta romandrà com un esdeveniment programat en la seva agenda.
- Registre d'esdeveniments realitzats: Serà una manera de conèixer més a l'usuari i així encertar en futures propostes. Si l'usuari ha realitzat un esdeveniment en el dia d'avui, podrà enregistrar-ho a la seva agenda de la següent manera:

  Introduirà el nom del lloc on ha estat i el sistema cercarà coincidències i mostrarà uns registres que s'apropin. Si el lloc no es troba entre els resultats, l'usuari podrà introduir el lloc a la base de dades i localitzar-lo.
- Planificació d'esdeveniments futurs: De la mateixa manera que en el registre d'esdeveniments ja realitzats, la planificació d'esdeveniments futurs servirà per tenir una idea més formada de l'usuari i permetrà a ell mateix disposar d'una agenda complerta que li permetrà organitzar el seu dia a dia.

  El funcionament de la planificació serà el mateix que el del registre d'esdeveniments realitzats. L'usuari introduirà el lloc on vol anar i el sistema cercarà una sèrie de registres que s'apropin, si el lloc desitjat no es troba entre els resultats serà el propi usuari qui l'introdueixi i el localitzi.

Totes dues plataformes permetran a l'usuari consultar les seves amistats, accedir al servei de propostes d'esdeveniments i consultar esdeveniments programats a la seva

agenda. La planificació d'esdeveniments futurs i el registre d'esdeveniments ja realitzats només estaran disponibles per a l'aplicació web.

2.4. Planificació.

En la taula següent es mostra la relació de les diferents tasques a realitzar per a la construcció del projecte, conjuntament amb les dates previstes per a la realització d'aquestes.

Nom de la tasca	**Duració**	**Inici**	**Fi**	**Tasca predecessora**	**Recurs humà**
Planificació	9 dies	dilluns 07/02/11	dijous 17/02/11		Núria;Pedro
Anàlisis(Casos d'ús)	9 dies	dilluns 07/02/11	dijous 17/02/11		Núria;Pedro
Tecnologies	9 dies	dilluns 07/02/11	dijous 17/02/11		Núria;Pedro
Disseny web	16 dies	divendres 18/02/11	divendres 11/03/11	1	Núria
Domini amb JPA	16 dies	divendres 18/02/11	divendres 11/03/11	1	Núria
Formació Android	16 dies	divendres 18/02/11	divendres 11/03/11	1	Pedro
Casos d'ús generals Web	20 dies	dissabte 12/03/11	dijous 07/04/11	5	Núria
Casos d'ús general Android	20 dies	dissabte 12/03/11	dijous 07/04/11	6	Pedro
Casos d'ús socials Web	10 dies	divendres 08/04/11	dijous 21/04/11	7	Núria
Casos d'ús socials Android	10 dies	divendres 08/04/11	dijous 21/04/11	8	Pedro
Casos d'ús agenda Web	22 dies	dimecres 27/04/11	dijous 26/05/11	9	Núria
Casos d'ús agenda Android	22 dies	dimecres 27/04/11	dijous 26/05/11	10	Pedro
Revisió final Web	6 dies	dijous 02/06/11	dijous 09/06/11	11	Núria
Revisió final Android	6 dies	dijous 02/06/11	dijous 09/06/11	12	Pedro
Documentació	92 dies	dilluns 07/02/11	dimarts 14/06/11		Núria;Pedro

Taula. 2.1 Tasques

A continuació es mostra el diagrama de Gantt de les tasques per a la realització del projecte

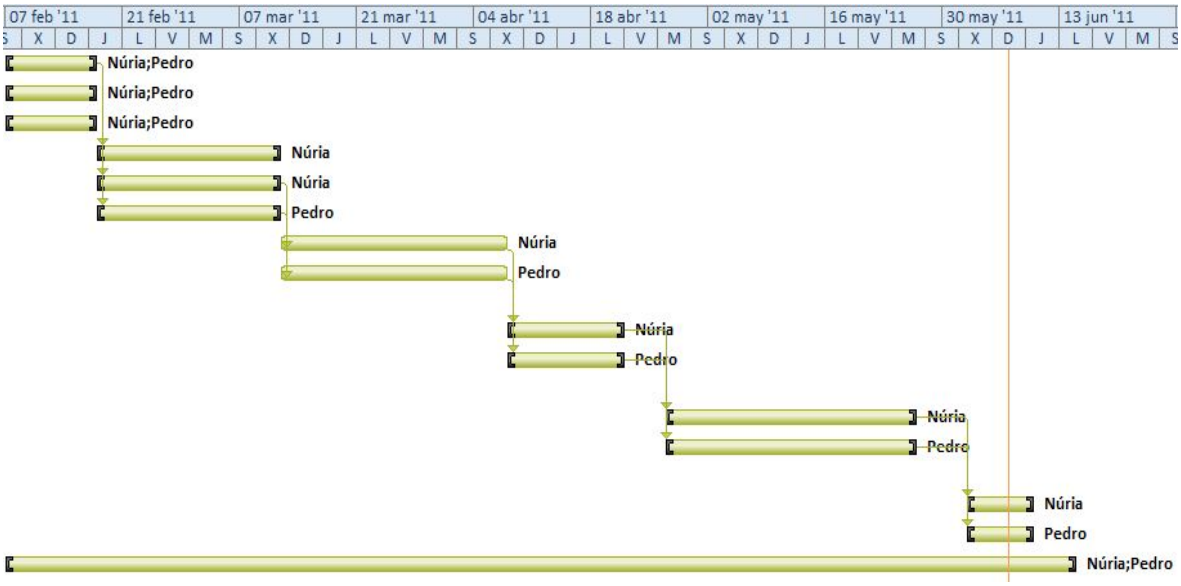

Figura 2. 1 Diagrama de Gantt

3. Anàlisi Tècnic.

En aquest apartat es definirà l'arquitectura sobre la qual es muntarà l'aplicació web i el servei web que proveirà d'informació a l'aplicació Android, els diferents casos d'ús que es podran realitzar tant des de l'aplicació web com des de l'aplicació Android i la base de dades on s'emmagatzemaren totes les dades.

3.1. Arquitectura del projecte.

L'arquitectura del projecte serà la que es pot veure en la imatge següent:

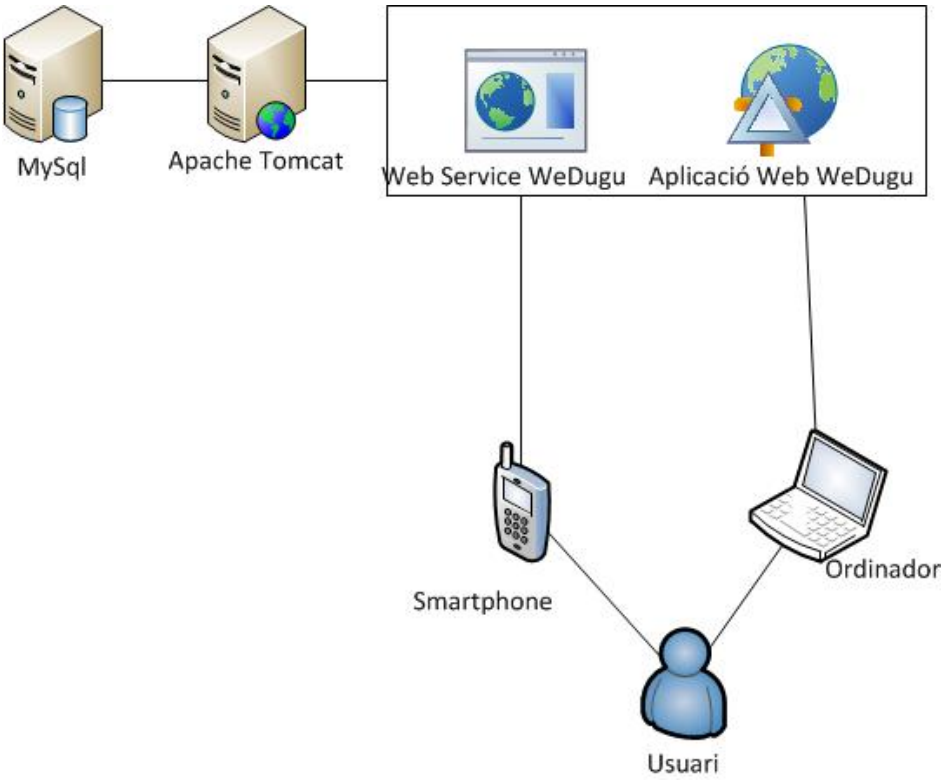

Figura 3. 1 Esquema Arquitectura

És necessari disposar d'un servidor de Base de dades on s'emmagatzemaran totes les dades referents a l'aplicació. S'ha escollit el sistema gestor de base de dades MySQL per la seva facilitat d'instal·lació i ús. Alhora la solució disposa també d'un servidor d'aplicacions Apache Tomcat que serà el contenidor tant de l'aplicació web com del servei web que proveirà a l'aplicació Android. Tots dos servidors (base de dades i aplicacions) s'executaran sobre una mateixa màquina. L'usuari farà servir tant l'aplicació web, com l'aplicació Android. En el cas de l'aplicació Android, aquesta,

internament, es comunicarà amb un servei web allotjat al servidor Apache Tomcat que, a la seva vegada es comunicarà amb el servidor de base de dades, evitant d'aquesta manera la connexió directa del smartphone amb la base de dades.

3.2. Casos d'ús.

A continuació es descriuran els casos d'ús que es podran realitzar des de l'aplicació Android i des de l'aplicació Web.

3.2.1. Casos d'ús Aplicació Android.

Diagrama de casos d'ús.

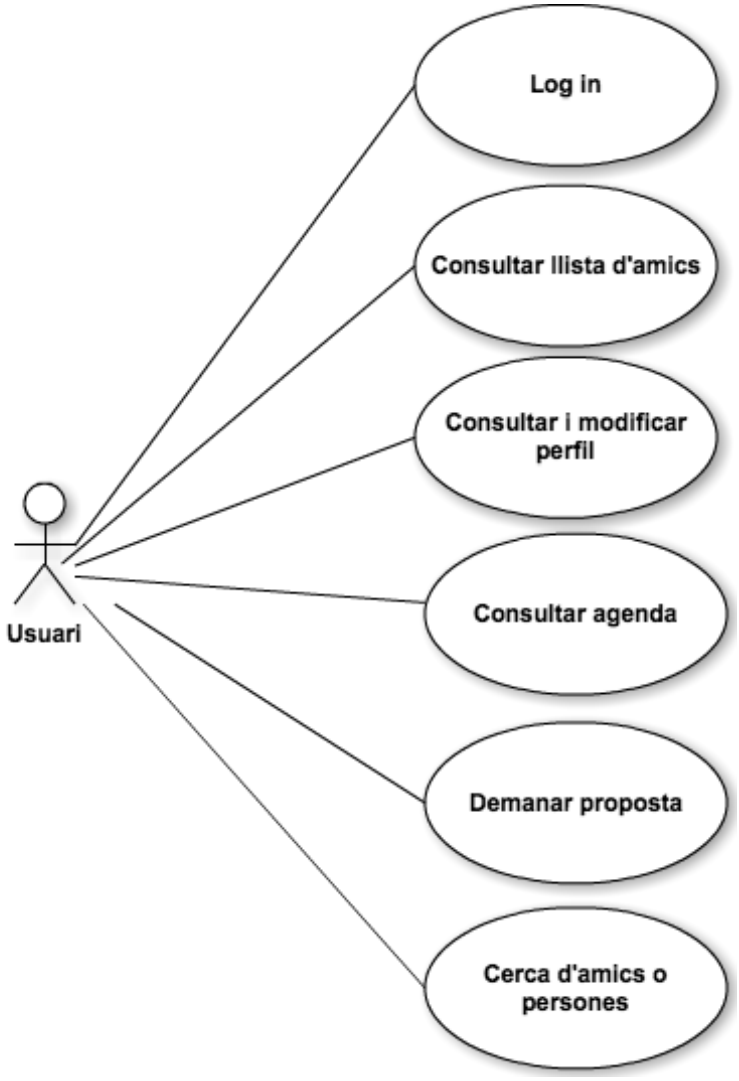

Figura 3. 2 Diagrama de casos d'ús de l'aplicació Android

Descripcions de casos d'ús.

Log in.

Actors: Usuari

Descripció: L'usuari vol iniciar sessió a l'aplicació.

Precondicions: La sessió no està iniciada.

Flux normal:

1. L'usuari introdueix el seu e-mail i la seva contrasenya.

2. L'e-mail existeix i la contrasenya és correcte

3 S'inicia sessió i es passa al menú de l'usuari.

Flux alternatiu:

2. L'e-mail no existeix o la contrasenya no són correctes

3. Es torna al pas 1.1

Consultar llista d'amics.

Actors: Usuari

Descripció: L'usuari vol consultar la seva llista d'amics

Precondicions: La sessió està iniciada.

Flux normal:

1. L'usuari prem sobre el botó "Amics".

2. Es mostra per pantalla una llista amb tots els amics de l'usuari.

3. L'usuari prem sobre un element de la llista.

4. Es mostra el perfil de l'amic en que ha premut l'usuari.

Flux alternatiu:

2.1 L'usuari no té amics

2.2 Es mostra per pantalla un missatge indicant que l'usuari no té amics.

Consulta i modificació del Perfil.

Actors: Usuari

Descripció: L'usuari vol consultar i modificar el seu perfil.

Precondicions: La sessió està iniciada.

Flux normal:

1. L'usuari prem sobre el boto "Perfil".

2. Es mostra per pantalla el detall del perfil de l'usuari seguit d'un botó "Modificar".

3. L'usuari prem sobre "Modificar".

4. Es substitueix el detall del perfil per caixes de text omplertes amb els diferents camps del perfil.

5. Una vegada l'usuari a modificat les que creu adients prem sobre "Ok".

6. Es modifica el perfil de l'usuari i es torna al menú principal.

Flux Alternatiu:

3.1 L'usuari decideix tornar enrere i es torna al menú principal

5.1 L'usuari decideix tornar enrere i es torna al menú principal.

Consultar agenda.

Actors: Usuari

Descripció: L'usuari vol consultar la seva agenda

Precondicions: La sessió està iniciada.

Flux normal:

1. L'usuari prem sobre l'opció "Agenda".

2. Es demana per pantalla el dia i el mes del qual es vol consultar els esdeveniments que es troben inclosos a l'agenda.

3. Es mostra una llista amb els esdeveniments programats per el dia indicat.

Flux Alternatiu:

2.1 L'usuari decideix tornar enrere i es torna al menú principal.

Demanar proposta.

Actors: Usuari, Sistema

Descripció: El sistema proposa una activitat a l'usuari .

Precondicions: La sessió està iniciada.

Flux normal:

1. L'usuari prem sobre l'opció "Propuesta".

2. Es mostra per pantalla un desplegable amb els tipus de proposta que estan disponibles i un botó per que siguin mostrades les propostes.

3. L'usuari prem sobre el botó "GO!"

4. El sistema obté les coordenades actuals en que es troba l'usuari i llença una consulta per tal d'obtenir les propostes segons el tipus indicat per l'usuari, filtrades per les preferències indicades al perfil d'aquest i la posició geogràfica en que es troba actualment.

5. Es mostra per pantalla una llista amb les propostes.

6. L'usuari prem sobre una de les propostes per obtenir més detall.

7. Es mostra per pantalla tota la informació de la proposta i dos botons, el primer per veure un mapa, el segon per incloure la proposta a l'agenda de l'usuari.

8. L'usuari prem sobre "Ver Mapa".

9. Es mostra un mapa de Google Maps on es troba marcada la posició geogràfica de l'usuari i la posició geogràfica de la proposta.

10. L'usuari torna enrere.

11. Es torna a mostrar la informació detallada de la proposta.

12. L'usuari prem sobre "Guardar".

13. La proposta és inclosa a l'agenda de l'usuari i es mostra un missatge per pantalla confirmant-ho.

Flux Alternatiu:

* Des de qualsevol punt l'usuari pot tornar a la pantalla anterior.

Cerca d'amics o persones.

Actors: Usuari

Descripció: L'usuari vol cercar usuaris entre els seus amics o entre la resta d'usuaris.

Precondicions: La sessió està iniciada.

Flux normal:

1. L'Usuari prem sobre el botó "Buscar" al menú principal.

2. Es mostra per pantalla un caixa de text, on l'usuari introduirà la cerca, i un botó.

3. L'Usuari introdueix la cerca i prem sobre el botó "Buscar".

4. Es mostra per pantalla una visió amb dues pestanyes. La pestanya fixada per defecte mostra els resultats de la cerca entre els amics de l'usuari actual.

5. L'usuari prem sobre la pestanya "Personas".

6. Es mostra el resultat de la cerca entre els usuaris de l'aplicació que no son amics de l'usuari actual.

Flux Alternatiu:

* L'usuari pot tornar enrere des de qualsevol punt

3.2.2. Casos d'ús Aplicació Web

Diagrama de casos d'ús.

Aquest és l'esquema dels casos d'ús implementats a l'aplicació separats per temàtiques:

Generals:

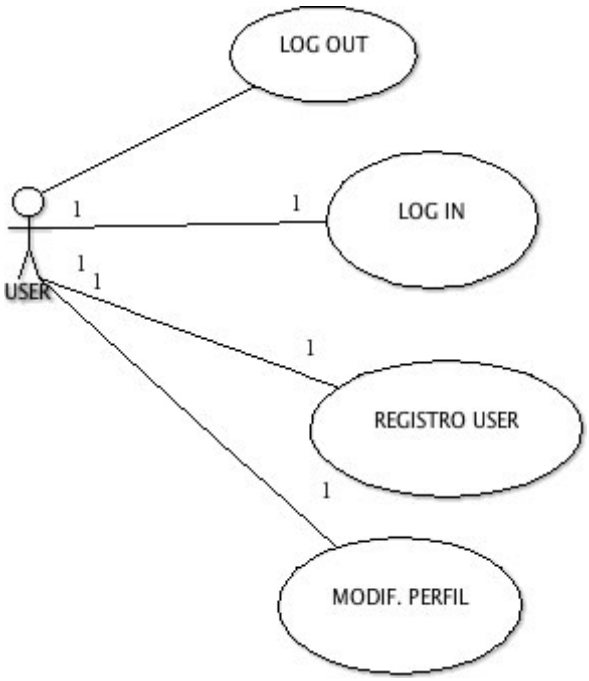

Figura 3. 3 Diagrama de casos d'ús generals

Referents a Amistats:

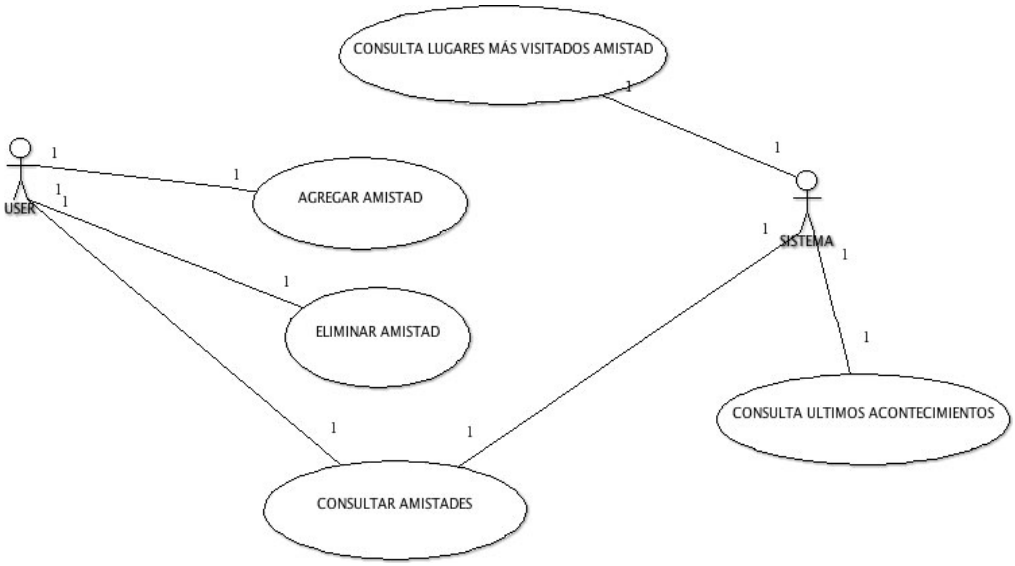

Figura 3. 4 Diagrama de casos d'ús d'amistats

Referents a l'agenda:

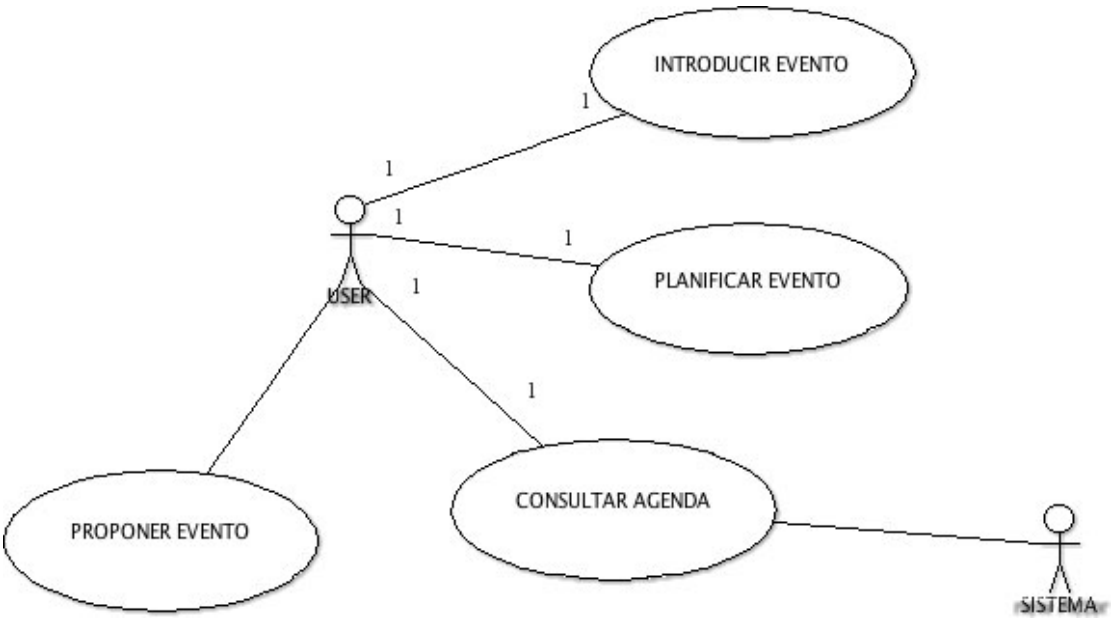

Figura 3. 5 Diagrama de casos d'ús agenda

Referents al lloc:

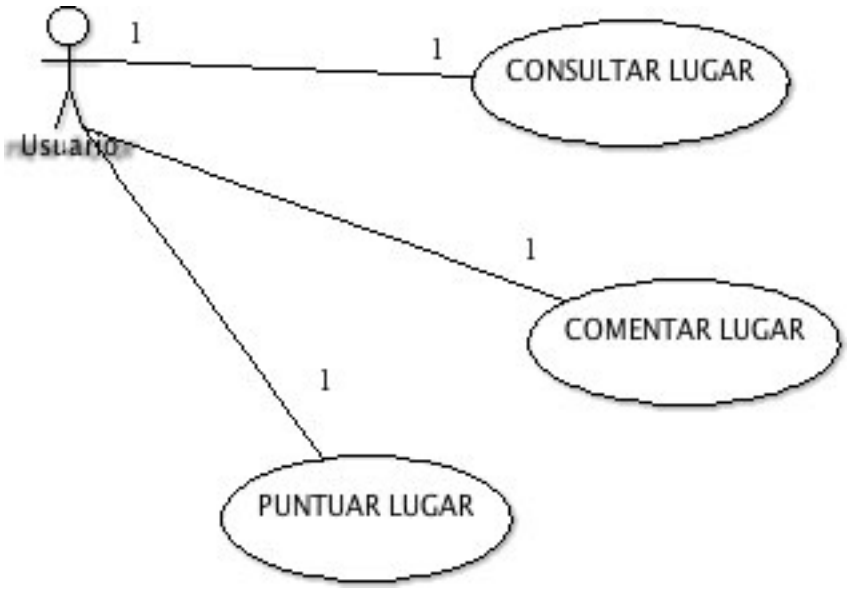

Figura 3. 6 Diagrama de casos d'ús del lloc

Descripcions de casos d'ús.

Generals:

- **Log in:**

  Actors: Usuari

  Descripció: L'usuari vol iniciar sessió a l'aplicació.

  Precondicions: La sessió no està iniciada.

  Flux normal:

1. L'usuari introdueix el nom d'usuari i la contrasenya.
2. L'usuari prem al botó iniciar sessió.
3. Les credencials són correctes i el sistema el redirigeix a la pàgina principal.

Flux alternatiu:

3.a. L'usuari ha introduit un nom d'usuari que no existeix.

3.b. L'usuari introdueix una contrasenya equivocada.

4.a. El sistema mostra un missatge indicant que l'usuari no existeix a la Base de Dades.

4.b. El sistema mostra un missatge indicant que la contrasenya no és correcta.

- **Log out:**

Actors: Usuari

Descripció: L'usuari vol tancar la sessió a l'aplicació.

Precondicions: La sessió està iniciada.

Flux normal:

1. L'usuari prem al botó tancar la sessió.
2. El sistema elimina la sessió actual i la cookie afegida al navegador per que mantingui la sessió oberta.
3. El sistema redirigeix a l'usuari a la pàgina inicial.

- **Registro user:**

Actors: Usuari

Descripció: L'usuari vol registrar-se a l'aplicació.

Precondicions:

Flux normal:

1. L'usuari introdueix totes les dades demanades al formulari inicial.
2. L'usuari prem al botó registrar.
3. El sistema dona d'alta a l'usuari i el redirigeix a la pàgina que conté el formulari per la creació de la primera fase del perfil.
4. L'usuari omple els camps demanats i prem al botó "Siguiente" per passar a la segona fase.
5. El sistema dona d'alta les dades indicades i mostra el formulari per la creació dels estils d'usuari.

6. L'usuari afegeix els estils corresponents al seu perfil i prem al botó "Siguiente".
7. El sistema dona d'alta les dades indicades i mostra el formulari per pujar una imatge de perfil.
8. L'usuari escull una imatge del seu directori d'arxius.
9. Es genera la petició al servidor que portarà la imatge i el sistema emmagatzemarà aquesta al servidor.
10. L'usuari prem al botó "Finalizar".
11. El sistema redirigeix a l'usuari a la pàgina principal amb la sessió iniciada.

Flux alternatiu:

1.a. L'usuari deixa un camp buit.

1.b. L'usuari introdueix una data de naixement que no correspon a tenir 18 anys.

1.c. L'usuari introdueix una direcció d'e-mail mal formulada.

1.d. L'usuari introdueix una direcció d'e-mail que ja està donada d'alta a la Base de Dades.

1.e. L'usuari introdueix dues contrasenyes que no coincideixen.

3.a. El sistema mostra un missatge advertint que s'ha d'informar totes les dades.

3.b. El sistema mostra un missatge advertint que s'ha de ser major de 18 anys.

3.c. El sistema mostra un missatge advertint que la direcció d'e-mail no és correcte.

3.d. El sistema mostra un missatge advertint que l'usuari ja existeix.

3.e. El sistema mostra un missatge advertint que les contrasenyes no coincideixen.

4.a. L'usuari no indica totes les dades.

5.a. El sistema passa a mostrar el formulari següent sense registrar les dades que passen a quedar buides a la Base de dades.

8.a. L'usuari no puja cap imatge al servidor.

9.a. No es genera cap petició.

- **Modif. Perfil:**

  Actors: Usuari

  Descripció: L'usuari vol modificar alguna de les dades del seu perfil.

Precondicions: La sessió està iniciada.

Flux normal:

1. L'usuari prem al botó d'edició de dades personals.
2. El sistema canvia la vista per mostrar les dades editables.
3. L'usuari introdueix els canvis necessaris.
4. L'usuari prem al botó guardar.
5. El sistema emmagatzema les dades i torna a canviar la vista a mode no editable.

Flux alternatiu:

1.a. L'usuari prem al botó d'edició d'estils de música.

1.b. L'usuari prem al botó d'edició d'estils de menjar.

1.c. L'usuari prem al botó d'edició d'estils de beguda.

1.d. L'usuari prem al botó d'edició d'estils d'activitats.

1.e. L'usuari prem al botó d'edició d'estils de cinema.

Referents a Amistats:

- **Consulta lugares más visitados amistades:**

  Actors: Sistema

  Descripció: Es busquen els llocs més visitats de la llista d'amistats.

  Precondicions: La sessió està iniciada. L'usuari té amistats.

  Flux normal:

  1. El sistema recull la llista d'amistats de l'usuari.
  2. Per cada amistat, el sistema recull la llista d'esdeveniments realitzats.
  3. Un cop obtinguda la llista d'esdeveniments, el sistema calcula els 7 més visitats.
  4. El sistema mostra una llista amb els 7 llocs més visitats.

  Flux alternatiu:

  3.a. Les amistats de l'usuari encara no han realitzat esdeveniments.

  4.a. El sistema mostra un missatge indicant que encara no hi ha informació d'esdeveniments de les seves amistats.

- **Consulta últimos acontecimientos:**

  Actors: Sistema

  Descripció: Es genera un llistat amb els últims esdeveniments ocorreguts a l'aplicació, entre les amistats i el propi usuari.

Precondicions: La sessió està iniciada.

Flux normal:

1. El sistema recull el llistat d'esdeveniments de l'usuari de la Base de Dades.
2. El sistema suma la llista d'esdeveniments de cada usuari que és amistat a la llista anterior.
3. La llista total, el sistema l'ordena per data d'ocurrència.
4. El sistema mostra una llista amb els últims esdeveniments.

Flux alternatiu:

3.a. Ni l'usuari ni les seves amistats han realitzat cap esdeveniment.

4.a. El sistema mostra un missatge indicant que encara no hi ha informació d'esdeveniments.

- **Agregar Amistad:**

Actors: Usuari

Descripció: L'usuari vol afegir una persona com amistat.

Precondicions: La sessió està iniciada.

Flux normal:

1. L'usuari prem al botó "Agregar amistad".
2. El sistema mostra una finestra amb un camp per introduir el nom de la persona.
3. L'usuari introdueix el nom de la persona que busca.
4. L'usuari prem al botó "Buscar amigo".
5. El sistema mostra un llistat amb usuaris el nom del qual s'aproxima al text introduït.
6. L'usuari selecciona de la llista la persona que cercava.
7. L'usuari prem al botó "Agregar".
8. El sistema afegeix a la Base de Dades a aquesta persona com amistat seva.
9. El sistema torna a mostrar la pàgina principal on ja figura la nova amistat a la caixa d'amistats.

Flux alternatiu:

3.a. L'usuari no introdueix un nom.

5.a. El sistema mostra un missatge indicant que ha d'indicar un nom.

5.b El sistema no troba coincidències. Mostra un missatge indicant el fet i l'opció de realitzar una nova cerca o sortir de la finestra.

6.c L'usuari no selecciona cap persona.

6.d L'usuari prem a sobre de la persona.

7.d El sistema obre una nova finestra amb el perfil de la persona premuda.

8.c El sistema mostra un missatge advertint que ha de seleccionar la persona que vol agregar com amistat.

- **Eliminar amistad:**

  Actors: Usuari

  Descripció: L'usuari vol eliminar de la seva llista d'amistats una persona.

  Precondicions: La sessió està iniciada.

  Flux normal:

  1. L'usuari accedeix al perfil d'una amistat.
  2. El sistema mostra un missatge indicant que la persona és amistat nostre i l'opció d'eliminar-la de la nostre llista.
  3. L'usuari prem al text "Quiero dejar de serlo" (per eliminar-la).
  4. El sistema esborra aquesta persona de la llista d'amistats.
  5. El sistema redirigeix a l'usuari a la pàgina principal.

- **Consultar amistades:**

  Actors: Sistema, usuari

  Descripció: Es genera un llistat amb totes les amistats de l'usuari.

  Precondicions: La sessió està iniciada.

  Flux normal:

  1. El sistema recull de la Base de Dades les amistats de l'usuari.
  2. El sistema genera una llista i en mostra 5 persones diferents cada vegada que s'executa la vista.

  Flux alternatiu:

  2.a. L'usuari no té amistats. El sistema mostra un missatge indicant-ho.

  3.b. L'usuari vol veure la llista sencera d'amistats i prem al botó "+Muestra todos".

  4.b. El sistema obre una finestra amb totes les amistats de l'usuari ordenades per ordre alfabètic.

Referents a l'agenda:

- **Introducir evento:**

  Actors: Usuari

  Descripció: L'usuari indica on ha estat.

  Precondicions: La sessió està iniciada.

  Flux normal:

  1. L'usuari introdueix el nom del lloc on ha estat a la caixa "Donde has estado hoy?".
  2. El sistema genera una llista de llocs que s'aproximen al text introduït per l'usuari i la mostra.
  3. L'usuari selecciona de la llista el lloc que cercava.
  4. L'usuari prem al botó "Ok".
  5. El sistema emmagatzema el nou esdeveniment amb data d'avui.
  6. El sistema mostra un missatge preguntant si vol continuar a la pàgina principal o bé consultar l'agenda.

  Flux alternatiu:

  2.a. El sistema no troba cap coincidència amb el text introduït i mostra el formulari per donar d'alta el lloc.

  3.a. L'usuari selecciona el tipus de lloc.

  3.a.1 El sistema mostra el formulari per donar d'alta el perfil del lloc.

  3.a.2 L'usuari prem al botó "Ok".

  3.a.3 El sistema mostra un mapa on l'usuari haurà d'indicar la ubicació del lloc.

  3.a.4 L'usuari indica la ubicació del lloc.

  3.a.5 L'usuari prem al botó "Ok".

  3.b El lloc indicat és un cinema. El sistema mostra el formulari per donar d'alta el lloc afegint el camp nom de la pel·lícula.

  3.b.1 L'usuari introdueix el nom de la pel·lícula.

  3.b.2.a El sistema busca pel·lícules el nom de les quals s'aproximi al text introduït i genera un llistat.

  3.b.2.b El sistema no troba coincidències, per tant entén que la pel·lícula s'haurà de donar d'alta.

3.b.3.a L'usuari selecciona de la llista la pel·lícula vista i prem al botó "Aceptar".

3.b.3.b L'usuari no selecciona cap pel·lícula i prem al botó "Aceptar".

3.b.3.b.1 El sistema mostra un missatge informant que l'ha de seleccionar.

5.a El sistema emmagatzema el lloc nou i l'esdeveniment a l'agenda de l'usuari.

7.e L'usuari no indica la ubicació del lloc.

8.e El sistema mostra un missatge indicant que ha de fer-ho.

- **Planificar evento:**

Actors: Usuari

Descripció: L'usuari afegeix a l'agenda un esdeveniment que realitzarà pròximament.

Precondicions: La sessió està iniciada.

Flux normal:

1. L'usuari tecleja sobre un dia qualsevol de l'agenda (el dia que vol realitzar l'activitat).
2. El sistema obre una finestra amb un formulari on figura la data de l'esdeveniment a planificar i la llista d'estils.
3. L'usuari escull l'estil que desitja.
4. El sistema genera un llistat de propostes segons els gustos de l'usuari per l'estil indicat i les mostra a una nova finestra.
5. L'usuari selecciona del llistat el lloc al que desitja anar.
6. L'usuari prem al botó "Añadir Evento".
7. El sistema emmagatzema l'esdeveniment a l'agenda
8. El sistema pregunta a l'usuari si vol dirigir-se a la pàgina principal o continuar a l'agenda.

Flux alternatiu:

4.a. El sistema no troba propostes adequades a l'usuari i mostra un missatge indicant-ho.

5.b. L'usuari no selecciona cap lloc del llistat.

7.b. El sistema mostra un missatge indicant que ha de seleccionar el lloc desitjat.

5.c L'usuari prem a sobre d'un lloc de la llista.

6.c El sistema obre una nova finestra amb el perfil del lloc demanat [cas d'ús consultar lugar].

- **Consultar agenda:**

  Actors: Usuari, sistema

  Descripció: Es mostren els esdeveniments emmagatzemats a la Base de Dades.

  Precondicions: La sessió està iniciada.

  Flux normal:

  1. El sistema genera un llistat amb els esdeveniments que s'ha de realitzar pròximament (s'han planificat).
  2. El sistema mostra el llistat a la pàgina principal i la pàgina de perfil de l'usuari.

  Flux alternatiu:

  2.a. No hi han esdeveniments pròxims programats i el sistema mostra un missatge informant-ho.

  1.b. L'usuari vol consultar l'agenda.

  2.b. El sistema obté tots els esdeveniments de la Base de Dades.

  3.c El sistema col·loca al dia del calendari pertinent cada esdeveniment.

- **Proponer evento:**

  Actors: Usuari

  Descripció: L'usuari vol propostes de llocs al seu voltant que podrien interessar-li.

  Precondicions: La sessió està iniciada. L'usuari ens permet obtenir la seva ubicació des del seu navegador.

  Flux normal:

  1. L'usuari prem al botó "No sabes donde ir o qué hacer? Accede a tus propuestas!"
  2. El sistema obté la ubicació de l'usuari.
  3. El sistema genera un llistat amb una proposta per tipus de lloc que es trobi dintre d'un radi de 3km i comparteixi perfil amb l'usuari.
  4. L'usuari selecciona una proposta que l'interessa.
  5. L'usuari prem al botó "Go!";
  6. El sistema dona d'alta l'esdeveniment a la Base de Dades.

7. El sistema pregunta a l'usuari si vol dirigir-se a la pàgina principal o continuar a l'agenda.

Flux alternatiu:

3.a. El sistema no troba cap proposta per l'usuari i mostra un missatge informant-ho.

3.a.1. L'usuari decideix obtenir una llista per tipus de lloc i prem al botó "Afinar el resultado".

3.a.2 El sistema mostra la llista de tipus de llocs per l'usuari escolli.

3.a.3 L'usuari selecciona el tipus del qual vol propostes.

3.a.4.a El sistema genera una llista amb 5 propostes només d'aquest tipus de lloc i la mostra.

3.a.4.b El sistema no troba propostes per aquest tipus de lloc i mostra un missatge indicant-ho.

4.a L'usuari no selecciona una proposta de la llista.

6.a El sistema mostra un missatge indicant que ha de seleccionar el pla que accepta.

4.b L'usuari prem a sobre d'un lloc.

5.b El sistema obre una nova finestra amb el perfil del lloc [cas d'ús consultar lugar].

Referents al lloc:

- **Consultar lugar:**

Actors: Usuari.

Descripció: Es mostren les dades referents a un lloc.

Precondicions: La sessió està iniciada.

Flux normal:

1. L'usuari a premut al nom d'un lloc.
2. El sistema obté les dades del lloc i la seva valoració.
3. El sistema obre una nova finestra on mostra totes aquestes dades.

Flux alternatiu:

3.a. Si el lloc encara no té valoració feta (comentaris o puntuació), el sistema mostrarà un missatge que indicarà que encara no ha estat valorat.

- **Comentar lugar:**

Actors: Usuari.

Descripció: S'afegeix un comentari a les valoracions del lloc.

Precondicions: La sessió està iniciada.

Flux normal:

1. L'usuari a premut al botó "Agregar un comentario".
2. El sistema obre un diàleg amb una caixa on l'usuari introduirà el comentari.
3. L'usuari escriu el comentari.
4. L'usuari prem al botó "Añade".
5. El sistema emmagatzema el comentari a la Base de Dades.
6. El sistema mostra el comentari afegit a la llista de comentaris.

Flux alternatiu:

3.a. L'usuari no escriu res a l'espai destinat al comentari.

4.a El sistema mostra un missatge informant que el comentari no pot ser buit.

- **Puntuar lugar:**

Actors: Usuari.

Descripció: S'afegeix una puntuació a les valoracions del lloc.

Precondicions: La sessió està iniciada.

Flux normal:

1. L'usuari a premut a sobre de la nota mitja del lloc o bé a sobre del número de votacions.
2. El sistema obre un desplegable amb puntuacions del 0 al 10.
3. L'usuari prem a la nota desitjada.
4. L'usuari prem al botó de tancar el desplegable.
5. El sistema emmagatzema la puntuació de l'usuari
6. El sistema re-calcula la nota mitja del lloc i la mostra.

Flux alternatiu:

1.a. El lloc no tenia puntuacions encara, per tant, l'usuari prem a sobre del text "Todavía no ha sido valorado" o bé el número de votacions (en aquest cas 0).

2.b.1 Si l'usuari havia puntuat anteriorment aquest lloc, el sistema mostra la puntuació que li va donar anteriorment.

3.c L'usuari no marca cap puntuació.

5.c El sistema tanca el desplegable sense emmagatzemar res.

3.3. Descripció de la base dades.

La Base de Dades d'aquest projecte és comú per l'aplicació web i l'aplicació mòbil.

3.3.1. Model de Dades.

A continuació es passarà a descriure el model de dades de l'aplicació. El diagrama general és bastant gran, per tant, és millor descriure taula per taula les dades necessàries. Només s'explicaran les relacions a una de les taules d'una de les dos direccions per no duplicar explicacions.

Diagrama físic:

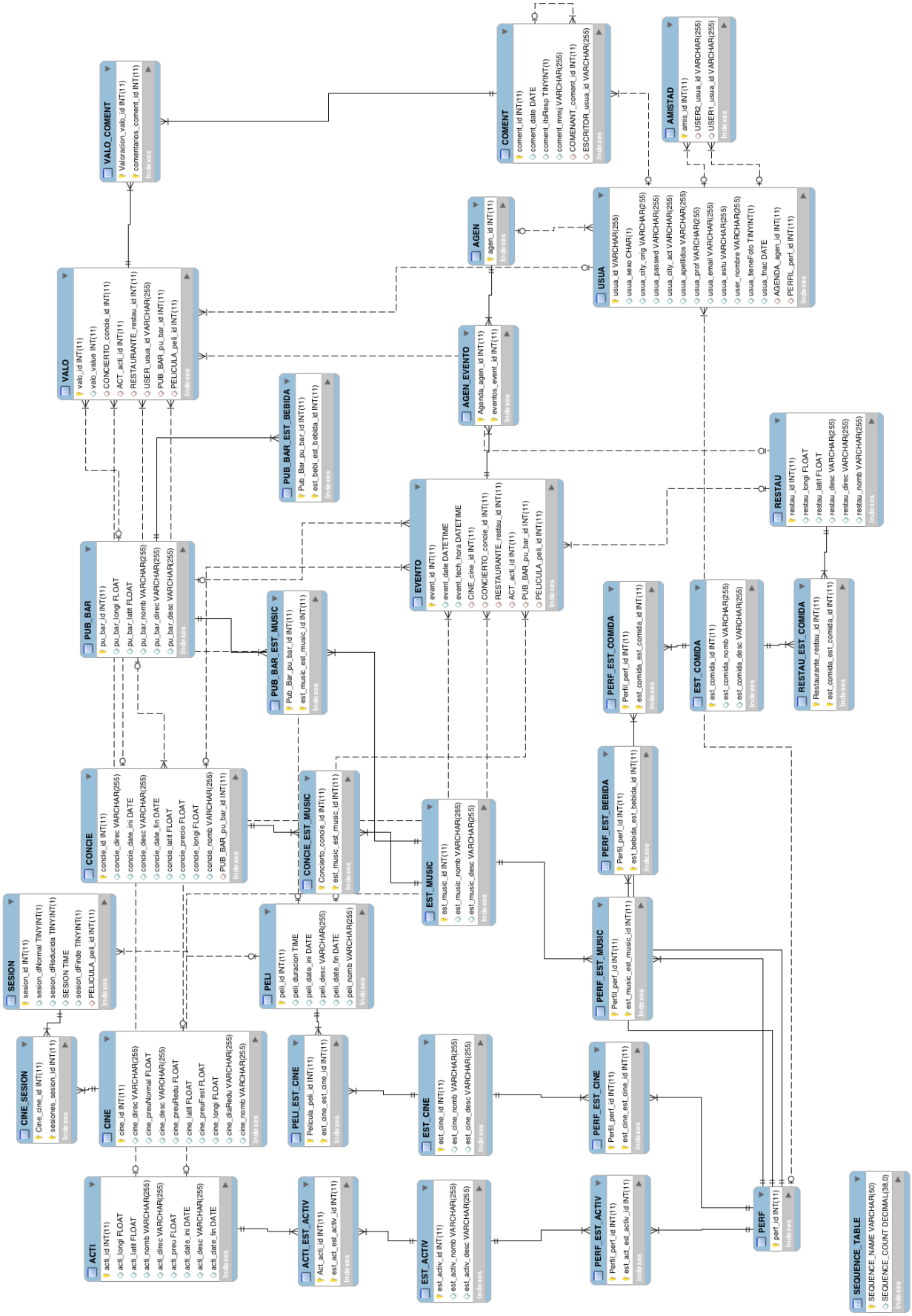

Figura 3.7 Diagrama físic Base de Dades

Taula Usua: És la taula on s'emmagatzemen les dades bàsiques de l'usuari.

Atributs:

- USUA_ID: És generat automàticament i són incrementals.
- USUA_SEXO: Sexe de l'usuari. F: Femení, M: Masculí o no informat.
- USUA_CITY_ORIG: Ciutat on va néixer l'usuari. Pot no ser informat.
- USUA_PASSWORD: Contrasenya de l'usuari per accedir a l'aplicació.
- USUA_CITY_ACT: Ciutat on viu l'usuari. Pot no ser informat.
- USUA_APELLIDOS: Cognoms de l'usuari.
- USUA_PROF: Ocupació actual de l'usuari. Pot no ser informat.
- USUA_EMAIL: Direcció d'e-mail que l'usuari utilitzarà com a nom d'usuari. Aquest camp és únic i no poden existir dos usuaris amb la mateixa direcció d'e-mail.
- USUA_ESTU: Estudis realitzats per l'usuari. Pot no ser informat.
- USER_NOMBRE: Nom de l'usuari.
- USUA_TIENEFOTO: Camp que indica si l'usuari ha pujat una imatge de perfil al servidor. Valor 1: Té foto, valor 0: no té foto.
- USUA_FNAC: Data de naixement de l'usuari. Ha de ser major a 18 anys.

Relacions:

- AGENDA. La taula usuari té una relació 1-1 amb Agenda. Un usuari té una agenda i una agenda és d'un usuari. S'identifica per l'identificador de l'agenda.
- PERFIL. Relació 1-1 amb Perfil. Un usuari té un perfil i un perfil és d'un usuari. S'identifica per l'identificador del perfil.
- AMISTAD. Un usuari pot tenir 0 o més usuaris com amistat.
- COMENT. Un usuari pot realitzar 0 o més comentaris.
- VALO. Un usuari pot realitzar 0 o més puntuacions.

Taula Perf: És la taula on s'emmagatzemen els gustos de l'usuari.

Atributs:

- PERF_ID: És generat automàticament i són incrementals.

Relacions:

- PERF_EST_ACTIV. La taula on es relacionen els estils d'activitat escollits per l'usuari. Pot tenir 0 o varis.

- PERF_EST_CINE. La taula on es relacionen els estils de cinema escollits per l'usuari. Pot tenir 0 o varis.
- PERF_EST_BEBIDA. La taula on es relacionen els estils de beguda escollits per l'usuari. Pot tenir 0 o varis.
- PERF_EST_MUSIC. La taula on es relacionen els estils de música escollits per l'usuari. Pot tenir 0 o varis.
- PERF_EST_COMIDA. La taula on es relacionen els estils de menjar escollits per l'usuari. Pot tenir 0 o varis.

Taula Agen: És la taula on s'emmagatzemen els esdeveniments realitzats per l'usuari.
Atributs:

- AGEN_ID: És generat automàticament i són incrementals.

Relacions:

- AGEN_EVENTO. La taula on es relacionen els esdeveniments realitzats per l'usuari. S'identifica per l'identificador de l'agenda i l'identificador de l'esdeveniment. Un usuari pot realitzar 0 o varis esdeveniments.

Taula Evento: És la taula on s'emmagatzema el tipus d'esdeveniment realitzat i la seva data.
Atributs:

- EVENT_ID: És generat automàticament i són incrementals.
- EVENT_DATE: És la data quan s'ha realitzat l'esdeveniment.
- EVENT_FECH_HORA: És la data quan s'ha introduït aquest esdeveniment a la Base de Dades.

Relacions:

- CINE. Relació 0-varis amb la taula Cinema que indica que un esdeveniment pot ser o no un cinema i un cinema pot ser a varis esdeveniments.
- CONCIE. Relació 0-varis amb la taula Concert que indica que un esdeveniment pot ser o no un concert i un concert pot ser a varis esdeveniments.
- PELI. Relació 0-varis amb la taula Pel·lícula que indica que un esdeveniment pot ser o no una pel·lícula i una pel·lícula pot ser a varis esdeveniments.
- ACTI. Relació 0-varis amb la taula Activitat que indica que un esdeveniment pot ser o no una activitat i una activitat pot ser a varis esdeveniments.

- PUB_BAR. Relació 0-varis amb la taula Pub_Bar que indica que un esdeveniment pot ser o no una pub o bar i un pub o bar pot ser a varis esdeveniments.
- RESTAU. Relació 0-varis amb la taula Restaurant que indica que un esdeveniment pot ser o no un restaurant i un restaurant pot ser a varis esdeveniments.

Taula Cine: És la taula on s'emmagatzema les dades d'un cinema.

Atributs:

- CINE_ID: És generat automàticament i són incrementals.
- CINE_DIREC: És la direcció on es troba el cinema.
- CINE_PREUNORMAL: Preu que ofereix aquest cinema a sessions d'un dia normal.
- CINE_DESC: Observacions que es vulguin indicar sobre el cinema.
- CINE_PREUREDU: Preu que ofereix aquest cinema a sessions del dia de l'espectador o bé sessions que ofereixen descompte.
- CINE_LATIT: Coordenada latitud on s'ubica el cinema.
- CINE_PREUFEST: Preu que ofereix aquest cinema a sessions de cap de setmana, o bé, dies festius.
- CINE_LONGI: Coordenada longitud on s'ubica el cinema.
- CINE_DIAREDU: Dia de la setmana que el cinema té com a dia de l'espectador.
- CINE_NOMB: Nom del cinema.

Relacions:

- CINE_SESION. Taula que relaciona el cinema amb les sessions que es projecten.

Taula Sesion: És la taula on s'emmagatzema les dades d'una sessió.

Atributs:

- SESION_ID: És generat automàticament i són incrementals.
- SESION_DNORMAL: Indica si la sessió és d'un dia considerat normal (ni festiu, ni reduït). Valor 0: no és un dia normal, valor 1: si és un dia normal.
- SESION_DREDUCIDA: Indica si la sessió és d'un dia considerat tarifa reduïda. Valor 0: no és un dia reduït, valor 1: si és un dia reduït.
- SESION: Hora a la qual es projecta la pel·lícula.
- SESION_DFINDE: Indica si la sessió és d'un dia considerat tarifa festiva o cap de setmana. Valor 0: no és un dia festiu, valor 1: si és un dia festiu.

Relacions:

- PELICULA. Relació 1-1 amb la pel·lícula que es projecta a aquesta sessió.

Taula Peli: És la taula on s'emmagatzema les dades d'una pel·lícula.

Atributs:

- PELI_ID: És generat automàticament i són incrementals.
- PELI_DURACION: Indica si el temps que dura la pel·lícula.
- PELI_DATE_INI: Indica la data quan comença a projectar-se la pel·lícula (l'estrena).
- PELI_DATE_FIN: Indica la data quan deixa de projectar-se la pel·lícula (surt de la cartellera).
- PELI_DESC: Sinopsis de la pel·lícula.
- PELI_NOMB: Nom de la pel·lícula.

Relacions:

- PELI_EST_CINE. Taula que relaciona la pel·lícula amb els estils amb els quals es pot classificar.

Taula Acti: És la taula on s'emmagatzema les dades d'una activitat.

Atributs:

- ACTI_ID: És generat automàticament i són incrementals.
- ACTI_LONGI: Coordenada longitud on s'ubica l'activitat.
- ACTI_LATIT: Coordenada latitud on s'ubica l'activitat.
- ACTI_NOMB: Nom de l'activitat.
- ACTI_DIREC: Direcció on es realitza l'activitat.
- ACTI_PREU: Preu de l'activitat.
- ACTI_DATE_INI: Data i hora quan comença a realitzar-se l'activitat.
- ACTI_DESC: Descripció de l'activitat.
- ACTI_DATE_FIN: Data i hora quan finalitza l'activitat.

Relacions:

- ACTI_EST_ACTIV. Taula que relaciona l'activitat amb els estils amb els quals es pot classificar.

Taula Concie: És la taula on s'emmagatzema les dades d'un concert.

Atributs:

- CONCIE_ID: És generat automàticament i són incrementals.
- CONCIE_LONGI: Coordenada longitud on s'ubica el concert.

- CONCIE_LATIT: Coordenada latitud on s'ubica el concert.
- CONCIE_NOMB: Nom del concert.
- CONCIE_DIREC: Direcció on es realitza el concert.
- CONCIE_PRECIO: Preu del concert.
- CONCIE_DATE_INI: Data i hora quan comença el concert.
- CONCIE_DESC: Descripció del concert.
- CONCIE_DATE_FIN: Data i hora quan finalitza el concert.

Relacions:

- CONCIE_EST_MUSIC. Taula que relaciona el concert amb els estils amb els quals es pot classificar.
- PUB_BAR. Hi ha la possibilitat que el concert es realitzi a un pub o bar que existeix a la base de dades. Per tant aquesta relació indica que un concert pugui realitzar-se a 0 o 1 pub o bar i a aquest pub o bar es realitzin 0 o varis concerts.

Taula Pub_Bar: És la taula on s'emmagatzema les dades d'un pub o bar.

Atributs:

- PU_BAR_ID: És generat automàticament i són incrementals.
- PU_BAR_LONGI: Coordenada longitud on s'ubica el pub o bar.
- PU_BAR_LATIT: Coordenada latitud on s'ubica el pub o bar.
- PU_BAR_NOMB: Nom del pub o bar.
- PU_BAR_DIREC: Direcció on s'ubica el pub o bar.
- PU_BAR_DESC: Descripció del pub o bar.

Relacions:

- PUB_BAR_EST_MUSIC. Taula que relaciona el pub o bar amb els estils de música amb els quals es pot classificar.
- PUB_BAR_EST_BEBIDA. Taula que relaciona el pub o bar amb els estils de beguda amb els quals es pot classificar.

Taula Restau: És la taula on s'emmagatzema les dades d'un restaurant.

Atributs:

- RESTAU _ID: És generat automàticament i són incrementals.
- RESTAU _LONGI: Coordenada longitud on s'ubica el restaurant.
- RESTAU _LATIT: Coordenada latitud on s'ubica el restaurant.
- RESTAU _NOMB: Nom del restaurant.

- RESTAU _DIREC: Direcció on s'ubica el restaurant.
- RESTAU _DESC: Descripció del restaurant.

Relacions:

- RESTAU_EST_COMIDA. Taula que relaciona el restaurant amb els estils de menjar amb els quals es pot classificar.

Taula Est_Comida: És la taula on s'emmagatzema les dades d'un estil de menjar.

Atributs:

- EST_COMIDA_ID: És generat automàticament i són incrementals.
- EST_COMIDA_NOMB: Nom de l'estil de menjar.
- EST_COMIDA_DESC: Descripció de l'estil de menjar.

Relacions:

- Ja comentades.

Taula Est_Music: És la taula on s'emmagatzema les dades d'un estil de música.

Atributs:

- EST_MUSIC_ID: És generat automàticament i són incrementals.
- EST_MUSIC_NOMB: Nom de l'estil de música.
- EST_MUSIC_DESC: Descripció de l'estil de música.

Relacions:

- Ja comentades.

Taula Est_Cine: És la taula on s'emmagatzema les dades d'un estil de cinema.

Atributs:

- EST_CINE_ID: És generat automàticament i són incrementals.
- EST_CINE_NOMB: Nom de l'estil de cinema.
- EST_CINE_DESC: Descripció de l'estil de cinema.

Relacions:

- Ja comentades.

Taula Est_Activ: És la taula on s'emmagatzema les dades d'un estil de cinema.

Atributs:

- EST_ACTIV_ID: És generat automàticament i són incrementals.
- EST_ACTIV_NOMB: Nom de l'estil de l'activitat.
- EST_ACTIV_DESC: Descripció de l'estil de l'activitat.

Relacions:

- Ja comentades.

Taula Est_Bebida: És la taula on s'emmagatzema les dades d'un estil de beguda.

Atributs:

- EST_ACTIV_ID: És generat automàticament i són incrementals.
- EST_ACTIV_NOMB: Nom de l'estil de beguda.
- EST_ACTIV_DESC: Descripció de l'estil de beguda.

Relacions:

- Ja comentades.

Taula Valo: És la taula on s'emmagatzema les valoracions d'un lloc.

Atributs:

- VALO_ID: És generat automàticament i són incrementals.
- VALO_VALUE: Puntuació donada.

Relacions:

- CONCIERTO. Una valoració pot ser o no d'un concert i el concert tindrà vàries o cap valoracions.
- ACT. Una valoració pot ser o no d'una activitat i l'activitat tindrà vàries o cap valoracions.
- RESTAURANTE. Una valoració pot ser o no d'un restaurant i el restaurant tindrà vàries o cap valoracions.
- PUB_BAR. Una valoració pot ser o no d'un pub o bar i el pub o bar tindrà vàries o cap valoracions.
- PELICULA. Una valoració pot ser o no d'una pel·lícula i la pel·lícula tindrà vàries o cap valoracions.
- USER. Una valoració és d'un usuari i un usuari pot fer vàries valoracions.
- VALO_COMENT. Taula que relaciona la valoració amb els comentaris realitzats.

Taula Coment: És la taula on s'emmagatzema els comentaris.

Atributs:

- COMMENT_ID: És generat automàticament i són incrementals.
- COMMENT_DATE: Data quan s'ha realitzat el comentari.
- COMMENT_ITSRESP: Camp que indica si el comentari és resposta a un altre. Valor 0: no és resposta, valor 1: si és resposta.

- COMMENT_MNSJ: Text del comentari.

Relacions:

- COMENANT. Si el comentari és resposta d'un anterior. Relació que indica de quin comentari és resposta.
- ESCRITOR. Relació amb taula USUA que indica que un comentari és d'un usuari i un usuari pot fer varis comentaris.

Taula sequence_table: És la taula on s'emmagatzema els últims índexs donats per realitzar l'assignació d'índexs automàtics.

Atributs:

- SEQUENCE_NAME: Nom que identifica la taula de la qual és seqüència.
- SEQUENCE_COUNT: Valor d'índexs donats.

Relacions:

No hi ha.

4. Tecnologies.

4.1. Struts vs Php.

4.1.1. Struts.

Struts és una eina de suport per el desenvolupament d'aplicacions Web que utilitza el patró MVC (Model-Vista-Controlador) sota la plataforma Java EE (Java Enterprise Edition). Té caràcter de software lliure i és compatible amb totes les plataformes amb les que Java Enterprise és disponible.

Funcionament: El navegador genera una sol·licitud que és atesa pel controlador (un servlet especialitzat). Aquest mateix és l'encarregat de tractar la sol·licitud, continuar la configuració que se li ha assignat a l'arxiu de configuració xml i cridar a l'Action corresponent passant-li els paràmetres enviats. L'Action instanciarà o utilitzarà els objectes de negoci per concretar la tasca. Segons el resultat que retorni l'Action, el Controlador derivarà la generació de la interfície a una o més pàgines JSP's, las quals podran consultar els objectes del Model a fi de realitzar la seva tasca.

Tipus de llicència: Apache License**.**

Avantatges:

- Intenta simplificar la implementació d'una arquitectura segons el patró MVC. Gràcies a ell queda ben reflectida la separació entre la gestió del workflow de l'aplicació, del model d'objectes de negoci i de la generació de la interfície.
- Transporta automàticament les dades introduïdes al client (JSP) fins el controlador (Action) mitjançant formularis (ActionForm).
- Transporta automàticament les dades en el sentit contrari, des del controlador (Action) fins a la presentació (JSP) mitjançant els mateixos formularis (ActionForm).
- Implementa la part comú a totes les aplicacions a la part de controlador (ActionServlet).
- És codi lliure, tothom pot utilitzar les revisions del codi.

Inconvenients:

- Àmbit limitat. Struts és una solució web basada en MVC que és dirigida a ser implementada amb HTML, JSPs i Servlets.

- Suport de les aplicacions J2EE: Les possibilitats d'escollir entre diferents servidors es limiten ja que Struts requereix d'un servidor d'aplicacions que suporti especificacions JSP 1.1 i Servlet 2.2 com Apache Tomcat.
- Consumeix més recursos.

4.1.2. PHP.

PHP és un llenguatge de programació interpretat, dissenyat originalment per a la creació de pàgines web dinàmiques i, per tant, s'executa en el servidor. Té una gran semblança amb els llenguatges de programació estructurada més comuns com C o Perl i pot ser desplegat a la majoria de servidors web i a gairebé tots els sistemes operatius i plataformes, sense cap cost addicional.

Funcionament: Quan el client fa una petició al servidor demanant que aquest li enviï una pàgina web, el servidor executa l'intèrpret de PHP i processa l'script sol·licitat que generarà el contingut de forma dinàmica. El resultat és enviat per l'intèrpret al servidor, que al mateix temps, l'envia al client.

Tipus de llicència: GNU**.**

Avantatges:

- S'ha dissenyat orientat al desenvolupament d'aplicacions dinàmiques amb accés a informació emmagatzemada a una Base de Dades.
- Capacitat de connexió amb la majoria de motors de Base de Dades que s'utilitzen.
- És lliure, per tant és de fàcil accés per tothom.
- És possible la programació orientada a objectes.
- Inclou una extensa llibreria de funcions.

Inconvenients:

- No és precisament el llenguatge més ràpid ja que és un llenguatge interpretat.
- Si el desenvolupador no és conscient al programar, l'aplicació pot esdevenir vulnerable.
- S'ha de seguir una bona metodologia de programació ja que, de no fer-ho així, l'aplicació serà molt difícil de mantenir a causa de l'excés de codi.
- PHP és flexible als tipus de variables. Pot donar lloc a un comportament inesperat degut a un error del programador que altres idiomes no permeten.

4.2. Decisió.

Després d'estudiar els pros i contres de cada tecnologia, es va arribar a la conclusió de desistir de PHP i fer servir el patró Model Vista Controlador amb el framework Struts..

La raó principal ha estat la menor experiència en la tecnologia PHP i, per tant, la gran dificultat que suposa, en només 6 mesos, poder realitzar un projecte amb una correcta metodologia de la programació, amb totes les funcionalitats desitjades i amb uns patrons ben marcats. Molts usuaris coincideixen que una errònia pràctica de la seva programació comporta vulnerabilitats i codi "brut"/inutilitzat.

4.3. Tecnologies utilitzades.

4.3.1. JQuery.

JQuery és un framework de JavaScript, que permet simplificar la manera d'interactuar amb els documents HTML, manipular l'arbre del DOM, tractar events, desenvolupar animacions i afegit interacció amb la tècnica AJAX a pàgines web.

JQuery és un software lliure de codi obert.

Llicència: GPL i MIT.

Avantatges:

- Reducció de codi. Simplifica molt més qualsevol acció que s'hagués de realitzar sense utilitzar el framework.
- La reducció de codi implica la reducció de temps de codificació.
- Accés més simple al DOM.
- Gestió d'events més senzill.
- Realització de peticions AJAX al servidor.
- Plugins per les funcions més habituals:
    - JQuery UI: És un complement que permet implementar components diversos per generar interfícies d'usuari a pàgines web..[1]

- Uploadify: Plugin que integra de manera senzilla la pujada de fitxers al servidor. [2]
- Yoxview: Plugin que permet inserir un visor d'imatges a la web.[3]
- Json-js: Plugin que permet convertir variables a estructures Json.[4]
- jsCalendar: Plugin que insereix una agenda mensual d'aspecte similar a Google Calendar.[5]

Desavantatges:

- Alguns programadors consideren que carregar les seves llibreries i extensions carreguen massa espai a la pàgina web.

4.3.2. AJAX.

AJAX és una tècnica de desenvolupament web per crear aplicacions. Aquestes aplicacions s'executen al client, mentre es manté la comunicació asíncrona amb el servidor en segon pla. D'aquesta manera, i és la seva característica principal, és possible realitzar canvis sobre la pàgina sense haver de recarregar-la. Aquest fet, augmenta la interactivitat, velocitat i usabilitat a l'aplicació.

AJAX és una tècnica vàlida per múltiples plataformes i disponible en diferents sistemes operatius i navegadors ja que està basat en estàndards oberts com JavaScript i DOM.

Avantatges:

- Poder modificar la pàgina sense haver de recarregar-la completament.

Desavantatges:

- Navegadors molt antics no suporten aquesta tècnica.
- Una aplicació que utilitza AJAX obté més recursos del servidor. Per tant, s'ha d'analitzar bé la necessitat de peticions AJAX per no saturar el servidor.

4.3.3. JSON.

JSON és un format lleuger per l'intercanvi de dades. És un subconjunt de la notació literal d'objectes JavaScript que no requereix de l'ús d'XML.

JSON està constituït per dos estructures:

- Una col·lecció de parells de nom/valor. A diferents llenguatges conegut com: objecte, registre, estructura, diccionari, taula hash o llista de claus.
- Una llista ordenada de valors. A la majoria de llenguatges, s'implementa amb arrays, vectors, llistes o seqüències.

L'ús de JSON és necessari per a la comunicació de les peticions AJAX al servidor en les dos direccions.

- De client a servidor: Com ja s'ha comentat anteriorment, és necessària la llibreria JQuery "json-js" per la conversió de variables a la petició al servidor.
- De servidor a Client: És necessària la llibreria JAVA "json-lib" per retornar objectes JSON al client.

  Des del client, la implementació de la funció $.ajax de JQuery ja és capaç d'interpretar l'objecte que li és retornat a la petició.

4.3.4. CSS.

Fulls d'estil en cascada. És un llenguatge utilitzat per definir la presentació d'un document estructurat escrit en XHTML o XML. El W3C és l'encarregat de formular l'especificació de fulls d'estil que serviran d'estàndard per els navegadors o clients.

La idea del desenvolupament amb CSS és separar l'estructura d'un document de la seva presentació.

Hi ha gran varietat de pàgines web on s'ofereix plantilles CSS lliures disponibles, però per aprofundir més en el llenguatge, en aquest projecte s'ha començat des de zero. Partint d'un disseny gràfic, realitzat amb el software Adobe Fireworks, el qual posteriorment s'ha anat adaptant a codi CSS amb les imatges que així ho requerien.

4.3.5. JPA.

Java Persistence API. API de persistència desenvolupada per la plataforma Java EE. Es tracta d'un framework del llenguatge de programació Java que utilitza dades relacionals en aplicacions, utilitzant la Plataforma Java.

El seu objectiu és no perdre els avantatges de l'orientació a objectes a l'hora d'interactuar amb una Base de Dades (seguint el patró de mapeig objecte-relacional) i permet utilitzar objectes regulars.

Hi ha diferents implementacions:

- Hibernate.
- TopLink.
- CocoBase.
- EclipseLink.
- OpenJPA.

- Kodo.
- JPOX.
- Amber.

La utilitzada a aquest projecte és la implementació Toplink Essentials d'Oracle.

4.4. Patrons.

Com és evident, degut a l'ús del framework Struts, el patró que es farà servir és el de Model-Vista-Controlador MVC:

MVC: Aquest patró d'arquitectura software separa les dades d'una aplicació, la interfície d'usuari i la lògica de control en tres components diferents. És freqüentment utilitzat a pàgines web. En aquest cas la vista serà la pàgina JSP, el model serà el Sistema de Gestió de Base de Dades (en aquest cas realitzat amb el framework JPA) junt amb la lògica de negoci i el Controlador serà el Servlet.

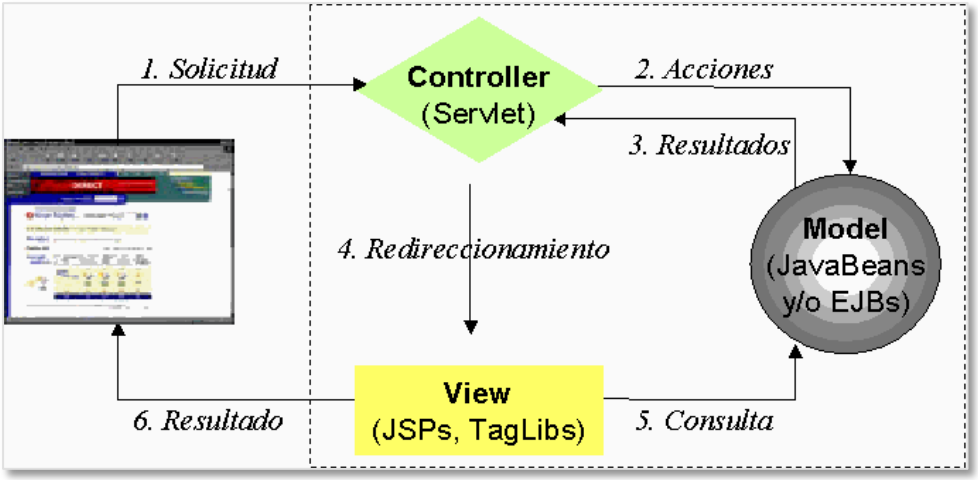

Figura 4.1 Patró Model-Vista-Controlador

Singleton (instància única): Dissenyat per restringir la creació d'objectes que pertanyen a una classe o el valor d'un tipus a un únic objecte. L'objectiu d'aquest patró és garantir un punt d'accés global a la classe. Aquest patró es farà servir per tractar la classe controladora de la Capa de Persistència, on s'accedeix a la Base de Dades.

5. Codificació.

El projecte s'ha desenvolupat d'acord amb l'esquema del patró de disseny Model-Vista-Controlador (MVC). A continuació, es descriurà el codi associat a cadascuna de les capes del patró.

5.1. Model.

A la capa model hi són inclosos els paquets:

- Domini: conté les classes que representen el model de dades. Com s'ha utilitzat el framework JPA, cadascuna d'aquestes classes serà reflectida com una taula a la Base de Dades.
- Beans: conté les classes que s'encarreguen de treballar amb les classes del Domini.
- Struts: conté les classes (Action i ActionForms) que són cridades per el Controlador. Aquestes són les encarregades de comunicar-se amb el paquet de Beans.

5.1.1. Domini.

Diagrama de classes del Domini:

S'explicarà només la configuració de notacions JPA d'una única classe ja que per totes es segueix el mateix procediment.

Agafem com exemple la classe Evento.java:

```
@Entity
@Table(name = "EVENTO")

public class Evento {

    @TableGenerator(name = "EVENTO_SEQ", table = "SEQUENCE_TABLE", allocationSize = 1,
    pkColumnName = "SEQUENCE_NAME",
    valueColumnName = "SEQUENCE_COUNT",
    pkColumnValue = "EVENTO")
    @Id
    @GeneratedValue(strategy = GenerationType.TABLE, generator = "EVENTO_SEQ")
    @Column(name = "agen_id")
    private int id;
    @OneToOne(cascade = CascadeType.ALL)
    private Act act;
    @OneToOne(cascade = CascadeType.ALL)
    private Concierto concierto;
    @OneToOne(cascade = CascadeType.ALL)
    private Pelicula pelicula;
    @OneToOne(cascade = CascadeType.ALL)
    private Cine cine;
    @OneToOne(cascade = CascadeType.ALL)
    private Pub_Bar pub_bar;
    @OneToOne(cascade = CascadeType.ALL)
    private Restaurante restaurante;
    @Column(name = "event_date")
    @Temporal(javax.persistence.TemporalType.TIMESTAMP)
    private Date fech;
    @Column(name = "event_fech_hora")
    @Temporal(javax.persistence.TemporalType.TIMESTAMP)
    private Date fech_creado;
```

Figura 5.1 Classe Evento.java

@Entity: Indica que aquesta taula serà una entitat

@Table(name = "x"): Nom que la classe tindrà a la Base de Dades.

@TableGenerator: Aquesta taula utilitzarà una taula auxiliar per obtenir els identificadors de manera automàtica i que aquests siguin consecutius.

- name: nom del registre a la taula d'identificadors (serà la primary key).
- table: taula auxiliar que s'utilitza per obtenir l'identificador.
- allocationSize: distància entre un identificador i el següent.
- pkColumnName: nom de la columna que conté el nom del registre (name).
- valueColumnName: nom de la columna que conté el valor dels identificadors.

@Id: indicació que l'atribut és l'identificador de la taula.

@GeneratedValue: tipus d'estratègia per generar automàticament l'identificador.

- strategy = GenerationType.TABLE: s'indica que utilitzarà la taula auxiliar, la qual s'ha definit anteriorment.
- generator = nom de la primary key a la taula que s'ha indicat abans.

@Column: l'atribut serà una columna de la taula.

@OneToOne: l'atribut és una relació de tipus 1-1 amb un altre taula.

@Temporal: l'atribut és de tipus data, s'indica quin tipus de data desitgem a la Base de Dades (TIMESTAMP).

Consultes a la Base de Dades:

```
@NamedQueries({
    @NamedQuery(name = "getEst_Music", query = "SELECT em FROM Est_Music em"),
    @NamedQuery(name = "getEst_MusicId", query = "SELECT em FROM Est_Music em WHERE em.id= ?1")
})
```

Figura 5.2 Consultes a la Base de Dades

Per altres casos, si és desitja indicar una relació d'un atribut de 1 a molts només s'ha d'utilitzar l'anotació @OneToMany.

@NamedQuery instanciarà una sentència de selecció a la Base de Dades d'aquesta taula. Seguint una bona programació orientada a objectes, només hauran de tenir sentències de selecció instanciades aquelles taules a les quals no es pot accedir des d'un nivell superior. Per exemple:

Si la taula usuari té relació amb la taula agenda i la taula perfil, a aquest dos últims no se'ls declararà una sentència de selecció ja que obtenint un usuari, s'obtenen totes les seves relacions.

Per que tot aquest muntatge d'anotacions tingui efecte i es creïn les taules a la Base de Dades, s'ha de configurar l'arxiu persistence.xml que es troba al paquet META-INF.

```
<?xml version="1.0" encoding="UTF-8"?>
<persistence version="1.0" xmlns="http://java.sun.com/xml/ns/persistence"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://java.sun.com/xml/ns/persistence http://java.sun
  <persistence-unit name="WeDugu" transaction-type="JTA">
<provider>oracle.toplink.essentials.ejb.cmp3.EntityManagerFactoryProvider</provider>
<class>Dominio.User</class>
    <exclude-unlisted-classes>false</exclude-unlisted-classes>
    <properties>
      <property name="toplink.cache.type.default" value="NONE"/>
      <property name="toplink.jdbc.user" value="root"/>
      <property name="toplink.jdbc.password" value="root"/>
      <property name="toplink.jdbc.url" value="jdbc:mysql://localhost:3306/WeDuguBBDD"/>
      <property name="toplink.jdbc.driver" value="com.mysql.jdbc.Driver"/>
      <property name="toplink.ddl-generation" value="create-tables"/>
```

Figura 5.3 Descripció de l'arxiu de configuració persistence.xml

Persistence-unit: Nom de l'entity manager.

Transaction-type: "JTA" si s'està utilitzant la plataforma de Java JavaEE o "RESOURCE LOCAL" si s'està utilitzant la plataforma JavaSE.

Provider: Indica la implementació de JPA que es farà servir, en aquest cas, TopLink.

Class: Les entitats que contindrà la base de Dades. S'ha de declarar un class per a cadascuna de les classes anotades com a entitat.

Exclude-unlisted-classes: Assignant un valor false a aquesta propietat els arxius amb extensió .jar que són adjunts no seran escanejats per les classes d'anotacions, mentre que si no s'indica, s'hauria de fer.

Property name="toplink.cache.type.default": Es desactiva la caché de la Base de Dades.

Property name="toplink.jdbc.user": usuari de connexió.

Property name="toplink.jdbc.password": contrasenya per la connexió.

Property name="toplink.jdbc.url": url per accedir a la Base de Dades.

Property name="toplink.jdbc.driver": driver que es fa servir com a connector. En aquest projecte es farà servir mysql.

Property name="toplink.ddl-generation": crea les taules de la base de la base de dades si aquestes no existeixen.

Totes les classes es corresponen en atributs, als mateixos que té la taula a la que representen a la Base de Dades. A l'hora, totes les classes tenen els mètodes get() i set() corresponents per atribut.

Tots els mètodes del tipus "getXString()" són retorns de muntatges en una única estructura de dades (sempre un ArrayList) de diferents informacions necessàries per la Vista.

La figura adjunta mostra un diagrama de classes a una mida ajustada a la pàgina si bé resulta farragós d'entendre, ja que no es pot percebre el detall:

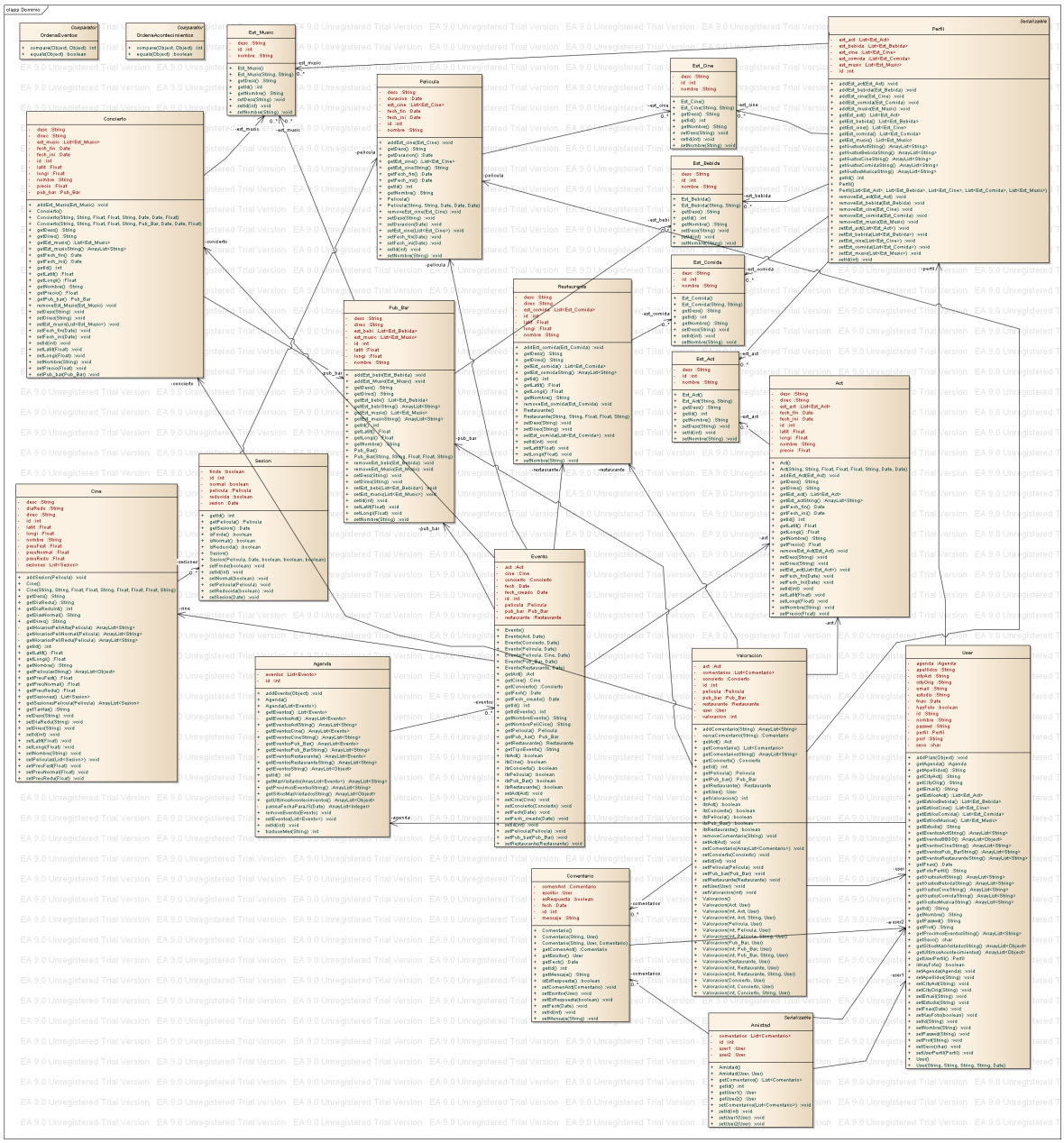

Figura 5.4 Diagrama de classes

A continuació es relaciona cada classe per separat indicant, només, els mètodes més característics:

- Act.java:
  - *public void addEst_Act(Est_Act est_act) throws Exception{}* i *public void removeEst_Act(Est_Act est_act) throws Exception{}*. Mètodes utilitzats per el manteniment de l'estil de l'activitat.

Figura 5.5 Diagrama classe Act.java

- Agenda.java:
  - *public ArrayList<String> getProximosEventosString() {}*: Retorna els propers esdeveniments que l'usuari té registrat a l'agenda.

    Per realitzar aquesta funció s'ordenen tots els esdeveniments de l'agenda d'antics a recents. Per aconseguir l'algoritme d'ordenació s'utilitza la funció sort() de la classe Collections de java. Es crea una nova classe que serà l'encarregada de marcar el criteri d'ordenació:

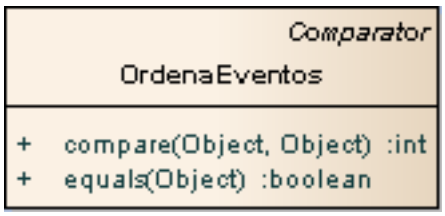

Figura 5.6 Diagrama classe OrdenaEventos

El criteri d'ordenació s'indicarà a la redefinició del mètode compare:

```
public int compare(Object o1, Object o2){
    Evento e1 = (Evento) o1;
    Evento e2 = (Evento) o2;
    /*Si e1 > e2 return positivo*/
    if (e1.getFech().after(e2.getFech())){
        return 1;
    }else if(e2.getFech().after(e1.getFech())){
        /*Si e2 > e1 return negativo*/
        return -1;
    }else return 0;/*Si e2 == e1 return 0*/
}
```

Figura 5.7 Mètode Compare

Un cop la llista d'esdeveniments és ordenada es selecciona només aquells que siguin superiors o igual a la data d'avui i es munta un ArrayList de tipus String que conté les dades necessàries per mostrar a la Vista.

Els retorns dels mètodes que envien dades per ser mostrades a la Vista sempre són ArrayList de tipus String. Si es dona el cas, com en aquest exemple, que d'un registre es volen vàries dades com és la url de la imatge i el nom, es munta l'ArrayList de manera intercalada (url-nom-url-nom-url-nom...). Quan es reculli aquest ArrayList caldrà tenir en compte que hi ha dues posicions per registre.

- *public ArrayList<Object> getSitiosMasVisitadosString () {}*: Retorna els llocs més freqüentats per l'usuari dintre de cada tipus (Restaurant, Pub/Bar, concert, activitat o cinema).
- *public ArrayList<Object> getUltimosAcontecimientos() {}*: Retorna els últims esdeveniments de l'usuari. S'entén per esdeveniment el registre d'activitats fetes i l'acceptació de propostes.

  És necessari ordenar la llista d'esdeveniments per data d'introducció a la Base de Dades, per tant, s'utilitzarà el mateix funcionament d'ordenació que en mètodes anteriors. Es crearà una classe nova que indicarà el criteri d'ordenació redefinint el mètode compare.

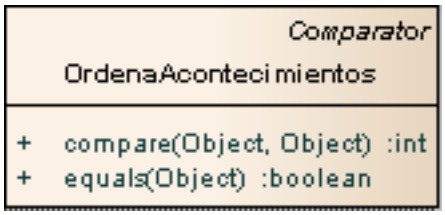

Figura 5.8 Classe OrdenaAcontecimientos

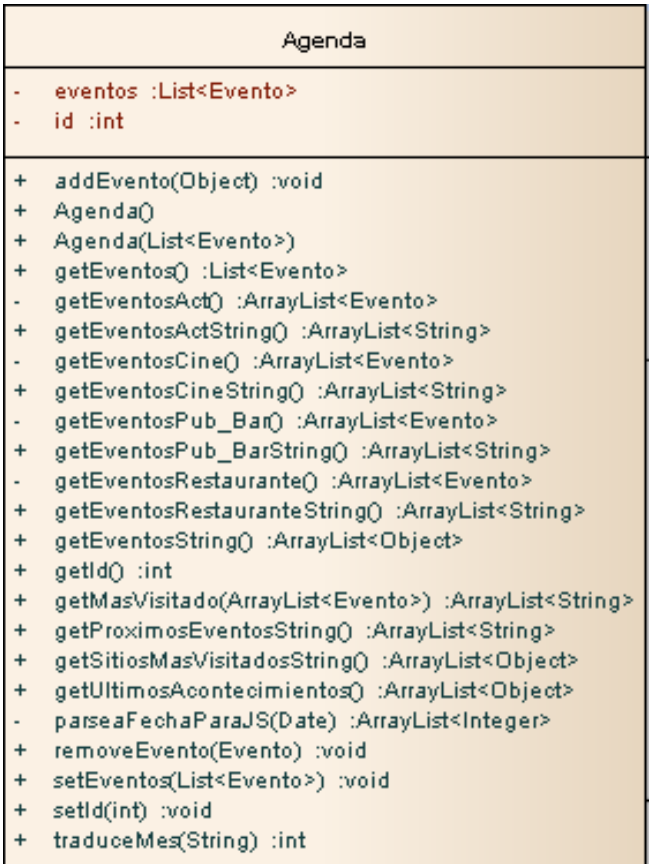

Figura 5.9 Classe Agenda

- Amistad.java. Diagrama de classe:

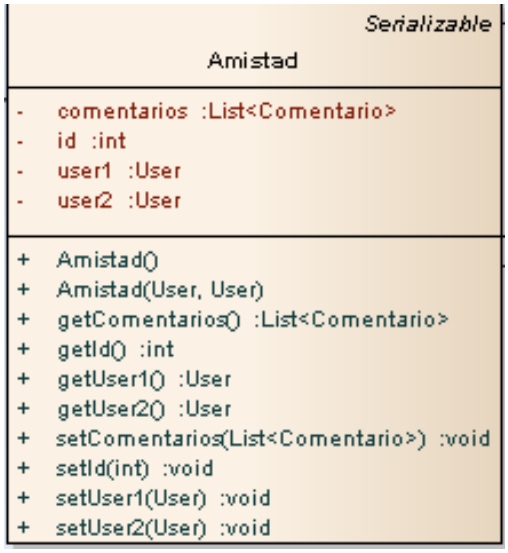

Figura 5.10 Classe Amistad

- Cine.java. Diagrama de classe:

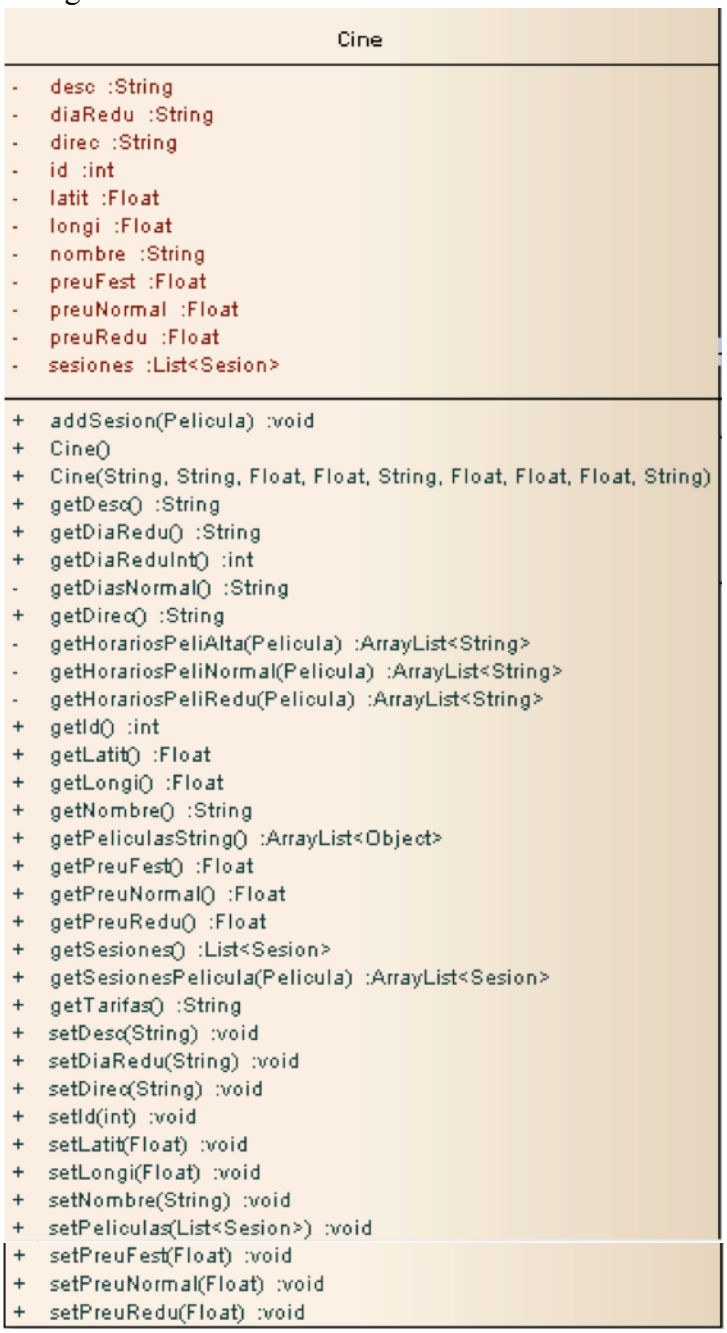

Figura 5.11 Classe Cine

- Comentario.java. Diagrama de classe:

Figura 5.12 Classe Comentario

- Concierto.java. Diagrama de classe:

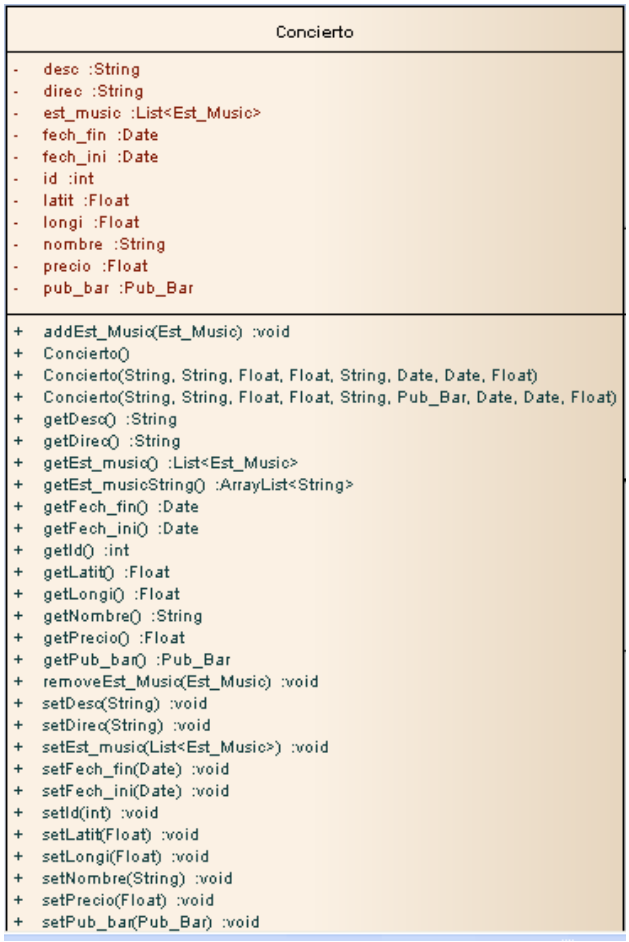

Figura 5.13 Classe Concierto

- Est_Act.java. Diagrama de classe:

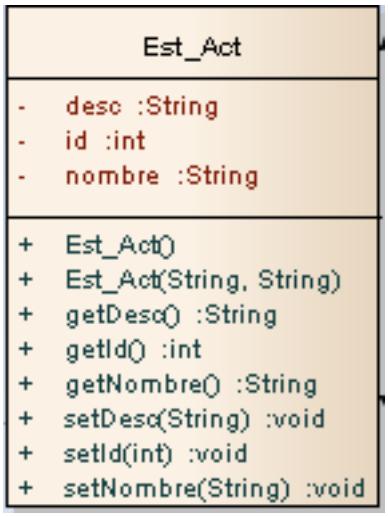

Figura 5.14 Classe Est_Act

- Est_Bebida.java. Diagrama de classe:

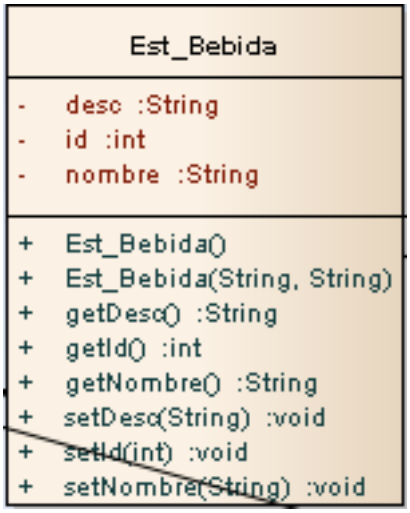

Figura 5.15 Classe Est_Bebida

- Est_Cine.java. Diagrama de classe:

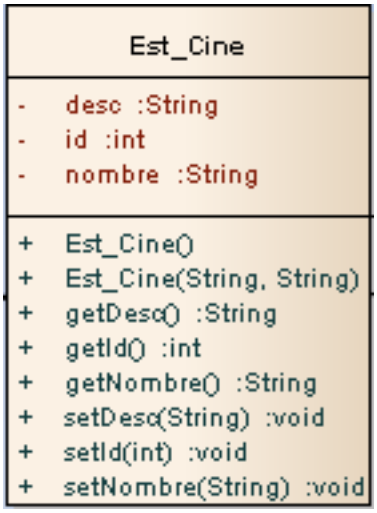

Figura 5.16 Classe Est_Cine

- Est_Comida.java. Diagrama de classe:

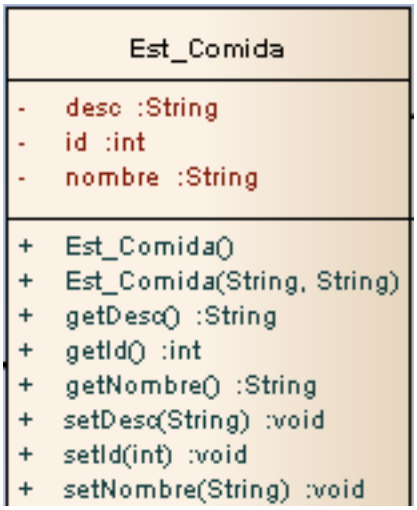

Figura 5.17 Classe Est_Comida

- Est_Music.java. Diagrama de classe:

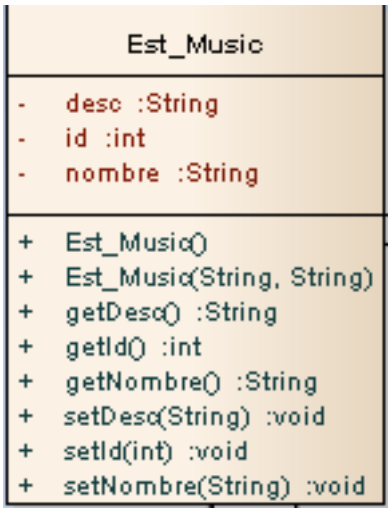

Figura 5.18 Classe Est_Music

- Evento.java.
    - *public String getTipoEvento() {}*: El nom del tipus d'esdeveniment. Els possibles són: "act", "concierto", "cine" o bé "pub_bar".

Diagrama de classe Evento:

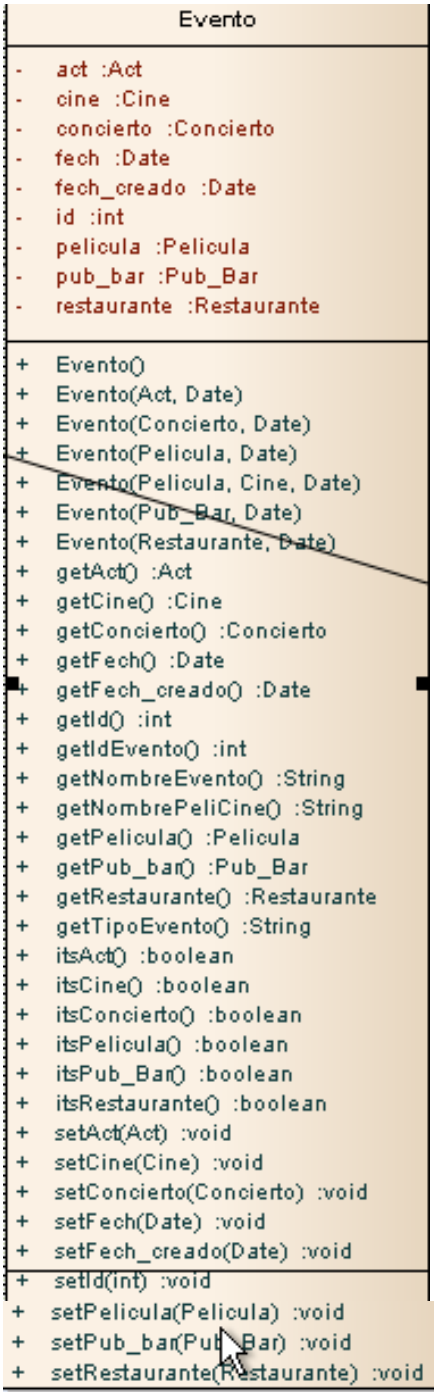

Figura 5.19 Classe Evento

- Pelicula.java. Diagrama de classe:

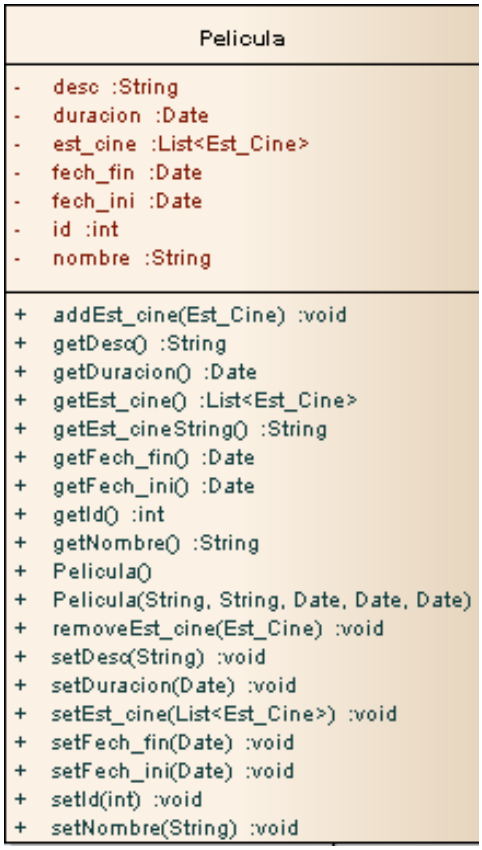

Figura 5.20 Classe Película

- Perfil.java.
  - *public String getGustosMusicaString(){}* - *public String getGustosBebidaString (){}* - *public String getGustosComidaString (){}*- *public String getGustosCineString(){}*- *public String getGustosActString (){}*: Totes aquestes funcions retornen les dades referents als estils d'aquest perfil. com es pot veure, per exemple, a la següent figura:

```
ArrayList<String> retorn = new ArrayList<String>();
if (this.est_bebida.isEmpty()) {
    retorn.add("FB_1");
    retorn.add("Ninguno.");
} else {
    for (int i = 0; i < this.est_bebida.size(); i++) {
        retorn.add("FB_" + String.valueOf(i + 1));
        retorn.add(this.est_bebida.get(i).getId()+"-"+
                this.est_bebida.get(i).getNombre());
    }
}

return retorn;
```

Figura 5.21 Mètode getGustos

getGustosBebidaString():

La funcionalitat dels índexs de l'ArrayList de retorn que contenen el text amb format "FB_numeroIndex" és identificar la fila de la taula a la qual es correspon.

Figura 5.22 Diagrama de classe Perfil

- Restaurante.java. Diagrama de classe:

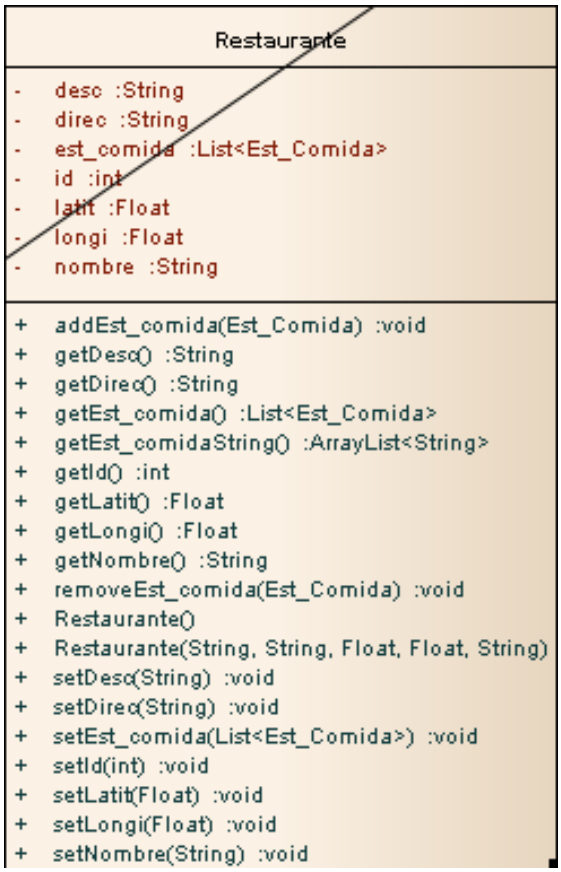

Figura 5.23 Diagrama Classe Restaurante

- Sesion.java. Diagrama de classe:

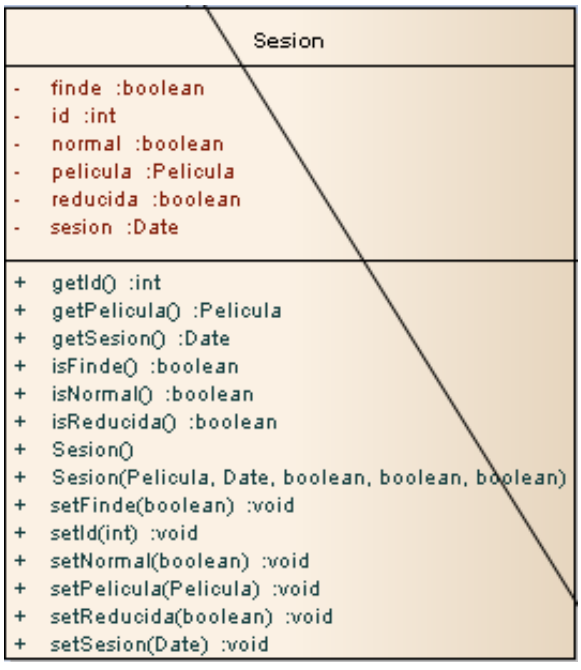

Figura 5.24 Diagrama Classe Sesion

- User.java. Diagrama de classe:

  - *public String getFotoPerfil(){}*: Retorna un String amb el path del servidor on es troba la imatge del perfil.

Figura 5.25 Diagrama classe Usuari

- Valoracion.java. Hi ha varis tipus de constructors ja que un usuari pot realitzar un comentari sobre un lloc però no fer una votació i viceversa.
  Diagrama de classe Valoracion:

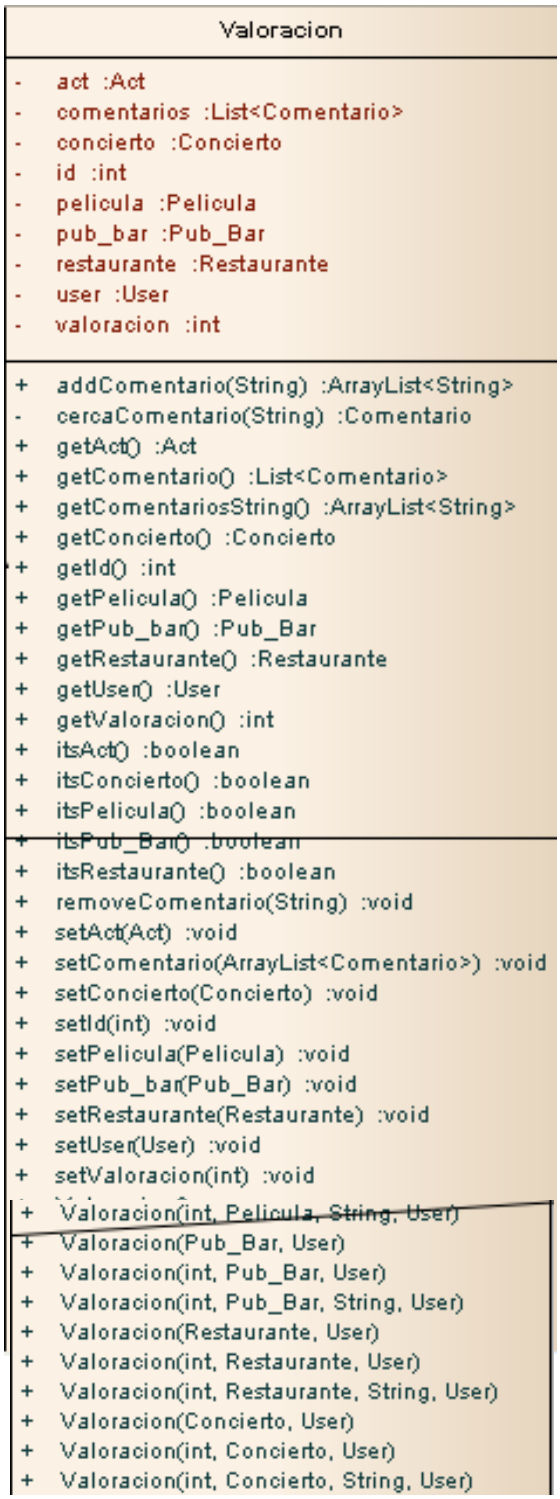

Figura 5.26 Diagrama classe Valoración

5.1.2. Persistència.

El següent paquet està format per les dues classes que s'encarreguen de treballar amb la Base de Dades.

Transaccion.java: Aquesta classe crea l'entity Manager declarat a l'arxiu persistence.xml i crea una transacció.

```
private EntityManagerFactory emf;
private EntityManager em;
private EntityTransaction tr;

public Transaccion(){
    this.emf = Persistence.createEntityManagerFactory("WeDugu");
    this.em = emf.createEntityManager();
    this.tr = em.getTransaction();
}
```

Figura 5.27 Classe Transacción

Aquesta transacció serà utilitzada a tots els mètodes que vulguin realitzar canvis a la Base de Dades. D'aquesta manera quan es vulgui començar una manipulació es cridarà al següent mètode:

```
public void begin(){
    tr.begin();
}
```

Figura 5.28 Mètode begin

Es realitzaran les operacions de manipulació pertinents i un cop s'hagi finalitzat es tancarà la transacció:

```
public void close(){
    tr.commit();
    em.close();
}
```

Figura 5.29 Mètode close

ControladorPersistence.java: A aquesta classe se li ha aplicat el patró singleton per que sempre es treballi sota la mateixa instància.

```
public synchronized static ControladorPersistence getInstancia() {
    if (instancia == null) {
        instancia = new ControladorPersistence();
    }
    return instancia;
}
```

Figura 5.30 Patró Singleton

El mètode més destacat d'aquesta classe és el mètode guardar().

```
public void guardar(Object o, Transaccion t) {
    t.getManager().merge(o);
}
```

Figura 5.31 Mètode guardar

Sempre que es realitzi un canvi sobre un registre de la base de dades s'haurà de cridar a aquest mètode que serà l'encarregat de realitzar el "commit". Cal destacar que sempre s'ha de treballar a favor del baix acoblament, per tant no es guardarà a la Base de Dades tots els objectes, si no aquells als quals no es pot accedir des d'un altre, ja que aquells que pengen d'un objecte superior seran emmagatzemats en cascada. Un exemple:
Si ha estat modificada una dada del perfil d'usuari, no es guarda la classe Perfil, ja que a la classe perfil s'accedeix a través de la classe Usuari. Per tant, es guardarà la classe Usuari i en cascada es guardaran les classes adjacents.
El resta de mètodes d'aquesta classe són consultes a la Base de Dades. Aquí hi han uns exemples:

Consulta amb retorn d'una sola fila:

```
public Est_Bebida cercaEst_Beb(int est_bebida, Transaccion tx) throws Exception {
    q = tx.getManager().createNamedQuery("getEst_BebidaId");
    q.setParameter(1, est_bebida);
    try {
        Est_Bebida eb = (Est_Bebida) q.getSingleResult();
        return eb;
    } catch (Exception e) {
        throw new Exception("No existe el estilo de bebida");
    }
}
```

Figura 5.32 Consulta retorn d'una fila

Aquesta consulta vol recuperar l'objecte corresponent a l'estil de beguda que s'identifica per l'identificador que es passa com a paràmetre.
Es munta la sentència de selecció que prèviament s'ha escrit amb l'anotació corresponent a la capçalera de la classe Est_Beb, utilitzant l'objecte transacció que arriba com a

paràmetre. Se li ha de passar el paràmetre 1 que és l'identificador (aquest paràmetre és l'utilitzat a la clàusula WHERE de la consulta).

S'executa la consulta com una consulta d'una sola fila (getSingleResul()), si funciona bé, es retorna l'objecte obtingut, en cas contrari, voldrà dir que no s'han trobat resultats, per tant s'envia una excepció informant-ho.

Consulta amb retorn de varies files:

```
public List<User> getAllUsers(Transaccion tx) {
    q = tx.getManager().createNamedQuery("getUsers");
    List result = (List) q.getResultList();
    return result;
}
```

Figura 5.33 Consulta retorn vàries files

L'única diferència que manté amb l'altre tipus de consultes resideix a la manera d'obtenir el resultat. En aquest cas la crida al mètode ha de ser getResultList().

5.1.3. Beans.

Totes les classes d'aquest paquet tenen una única instancia de la classe ControladorPersistence.java i un objecte Transaccion.java.

BeanAgregarAmigo.java: Aquest bean és l'encarregat de totes les operacions referents al cas d'ús "agregar una amistad".

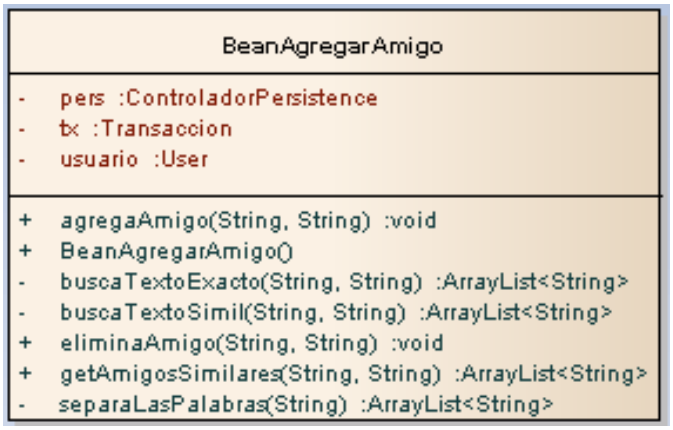

Figura 5.34 Classe BeanAgregarAmigo

Mètodes a comentar:

- getAmigosSimilares(). Aquest mètode serveix per buscar a persones que siguin usuàries i així, si es desitja, afegir-les com amistat.

Primer es realitza un crida al mètode buscaTextoExacto(...) que buscarà exactament el mateix text que ha introduït l'usuari com a nom de la persona que està buscant. Si per aquest text no s'aconsegueix cap resultat, es separa el text per paraules (tallant allà on hi hagi un espai) amb el mètode separaLasPalabras(), i es crida al mètode buscaTextoSimil(), que buscarà tots els usuaris que continguin cadascuna d'elles. Aquesta llista d'aproximació serà retornada a la vista per que l'usuari decideixi si finalment vol agregar a la persona o no.

- agregaAmigo(): Arribaran com a paràmetres els dos identificadors dels usuaris que passaran a tenir una amistat. S'accedeix a la Base de Dades per recuperar els objectes corresponents als dos usuaris. Cal comprovar, accedint a la llista d'amistats, que aquestes dos persones no siguin amigues actualment, en cas afirmatiu es llençarà una Excepció que serà recollida a la Vista indicant-li a l'usuari que no pot realitzar l'operació. Si no hi ha amistat prèvia, es crearà un objecte Amistad amb els dos usuaris i aquest objecte serà persistit amb el mètode guardar del ControladorPersistence:

```
User u1 = this.pers.cercaUserId(idUser,this.tx);
User u2 = this.pers.cercaUserId(idamigo, this.tx);

List<Amistad> amistadesUsuario = this.pers.getAmistadIdUser(u1, tx);
for (int i=0;i<amistadesUsuario.size();i++){
    if(amistadesUsuario.get(i).getUser1().equals(u2) ||
       amistadesUsuario.get(i).getUser2().equals(u2)){
        throw new Exception ("Ya eres amigo de este usuario.");
    }
}

Amistad nuevaAmistad = new Amistad(u1, u2);
this.tx.begin();
this.pers.guardar(nuevaAmistad,this.tx);
this.tx.close();
```

Figura 5.35 Mètode agregaAmigo

BeanDameDatos.java: Bean encarregat de realitzar diferents tipus de consultes normalment referents a la classe User.java.

- getEdad(): Aquest mètode calcula l'edat de l'usuari i l'envia a la Vista ja que el que hi ha emmagatzemat a la Base de Dades és la data de Naixement.

```
public String getEdad() {
    int edad = 0;
    Date hoy = new Date();
    edad = hoy.getYear() - this.usuario.getFnac().getYear();
    if (hoy.getMonth() < this.usuario.getFnac().getMonth()) {
        edad--;
    } else if (hoy.getMonth() == this.usuario.getFnac().getMonth()
            && hoy.getDate() < this.usuario.getFnac().getDate()) {
        edad--;

    }
    return String.valueOf(edad);
}
```

Figura 5.36 Mètode getEdad

- getInteresesQueMasCoinciden(): Mètode que cerca dintre de les amistats dels usuaris, els llocs visitats que més coincideixen entre ells.

  Primerament es recuperen les amistats de l'usuari cridant al mètode getAmistadIdUser(...) de la classe ControladorPersistence.java. Si el retorn d'aquesta crida és buit, voldrà dir que l'usuari encara no té cap amistat, per tant, directament es retornarà un missatge informant d'aquest fet. En cas contrari, si hi han amistats per analitzar, per cada una d'elles es recollirà els seus esdeveniments registrats a l'agenda separats per tipus (restaurant, pub/bar, cinema i activitat). Si després de la cerca la llista és buida, voldrà dir que cap de les amistats a registrat cap esdeveniment a la seva agenda, per tant, no es continua i s'informa del fet a l'usuari. En cas contrari, es cridarà al mètode calculaMasVisitados que s'encarregarà de recórrer aquesta llista de llocs visitats i mitjançant l'identificador de cadascun, muntarà una nova llista ordenada amb els 7 llocs més freqüents.

- getAcontecimientos(...): Mètode que mostrarà els últims esdeveniments realitzats per l'usuari i les seves amistats a la web, tant sigui l'acceptació d'una proposta com la introducció d'un pla ja realitzat.

  Primerament, es cridarà al mètode de la classe User getUltimosAcontecimientos(), que s'encarregarà de cridar al mètode amb el mateix nom de l'agenda (d'aquesta manera s'aconsegueix el baix acoblament desitjat, cada classe parla amb qui ha de parlar). Un cop realitzada la llista dels esdeveniments de l'usuari, es procedeix de la mateixa forma per cadascuna de les seves amistats. Quan s'ha realitzat la llista total, s'ordenarà en funció de la data d'inserció a la Base de Dades (el moment en que l'usuari va realitzar l'acció a la web). Per realitzar l'ordenació es cridarà a una nova classe que contindrà el criteri d'ordenació:

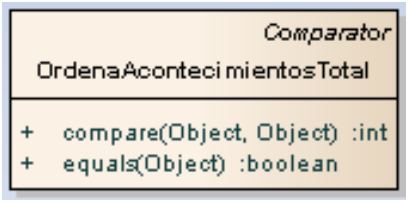

Figura 5.37 OrdenaAcontecimientosTotal

El retorn d'aquest mètode sempre serà de 10 registres. Per això és necessari el paràmetre vamosPor, que indicarà a partir de quina posició del llistat cal retornar.

Figura 5.38 Classe BeanDameDatos

BeanDamePlanes.java: Bean encarregat de realitzar propostes d'oci que poden interessar i estar a prop de l'usuari.

- damePlanes(...): Arribarà un paràmetre que indicarà si la proposta és filtrada (només es volen propostes d'un tipus de lloc) o bé es vol una proposta per tipus (funcionament estàndard).

  Sempre es retornaran 5 possibilitats com a màxim, en qualsevol dels dos casos.

  En funció del tipus de consulta (si és una consulta general o filtrada per tipus) es cridarà una vegada als mètodes:

  damePlanRestaurant(...), damePlanPub_Bar(...), damePlanAct(...), damePlanCinePeli(...) i damePlanConcierto(...) o 5 vegades, per obtenir 5 plans diferents d'un tipus de lloc.

  Tots aquests mètodes segueixen el mateix patró i, per tant, només s'explicarà un d'ells:

- damePlanRestaurant(...): S'agafen tots els restaurants de la Base de Dades i els estils de menjar que l'usuari ha informat al seu perfil. Per cada restaurant, per cadascun dels seus estils de menjar, es comprova si es troba dintre dels gustos de l'usuari. Si a més, es troba dintre d'un radi de 3km des de la ubicació on és l'usuari a la ubicació del restaurant (crida al mètode dentroDelRadio(...)), es considerarà que és una proposta vàlida i s'afegirà al contenidor de retorn.

  Un cop finalitzat l'anàlisi, si s'han trobat propostes, s'ordenarà aleatòriament aquest llistat (crida al mètode indicesAleatorios(...). Si no es realitza un ordre aleatori, cada vegada que l'usuari realitzi una proposta, li seran retornades les mateixes, en funció de l'ordre en que apareixen a la Base de Dades.

- dentroDelRadio(...): S'ha aplicat l'equació Haversine per conèixer la distància entre dos punts. Aquesta equació es pot trobar implementada en la majoria de llenguatges:

```
private boolean dentroDelRadio(double latitAct, double longiAct, double latitPlace, double longiPlace)
    double R = 6371; // km
    double dLat = Math.toRadians(latitPlace-latitAct);
    double dLon = Math.toRadians(longiPlace-longiAct);
    double a = Math.sin(dLat/2) * Math.sin(dLat/2) +
               Math.cos(Math.toRadians(latitAct)) * Math.cos(Math.toRadians(latitPlace)) *
               Math.sin(dLon/2) * Math.sin(dLon/2);
    double c = 2 * Math.atan2(Math.sqrt(a), Math.sqrt(1-a));
    double d = R * c;

    if(d<=3){
        return true;
    }else return false;
}
```

Figura 5.39 Mètode Haversine

Les posicions arriben com a paràmetres. Les del lloc han sigut obtingudes de la Base de Dades mentre que les de la ubicació de l'usuari les obté la Vista.

Un cop calculada la distància (km), si aquesta és inferior o igual a 3 voldrà dir que és vàlid, en cas contrari es retornarà un valor "false" per que aquesta proposta no sigui acceptada.

- indicesAleatorios(int cantidad, int size): El paràmetre "cantidad", indicarà quants números aleatoris es necessiten i el paràmetre "size" marcarà el rang d'on s'han d'extreure.

  Per tal que no es repeteixin números obtinguts s'ha de realitzar el següent muntatge:

```
if (size > 1) {
    if (cantidad == 1) {
        ale.add(this.aleatorio(size));
        metidos++;
    } else {
        ale.add(this.aleatorio(size));
        metidos++;
        for (int i = 0; i < cantidad; i++) {
            int num = this.aleatorio(size);
            do {
                for (int j = 0; j <= i; j++) {
                    if (ale.get(j) == num) {
                        num = this.aleatorio(size);
                        j = 0;
                        valid = false;
                    } else {
                        valid = true;
                    }
                }
            } while (!valid);
            if (metidos < cantidad) {
                ale.add(num);
                metidos++;
            }
        }
    }
} else {
    ale.add(0);
}
```

  Si el rang va de 0 a més gran de 1:

  - Si només es necessita un número aleatori: Es crida una única vegada a la funció que el dona (aleatorio(...)).
  - Si es necessita més d'un número: s'obté un número aleatori (crida a funció aleatorio(...)). Es suma el comptador de números obtinguts. Mentre no es tingui la quantitat de números necessària:
    - S'obté un número.

- Mentre no sigui vàlid: per tots els números ja obtinguts, si el número obtingut ja es troba a la llista de números escollits no serà vàlid, s'obté un altre i es torna a començar la comprovació a la llista de números escollits.

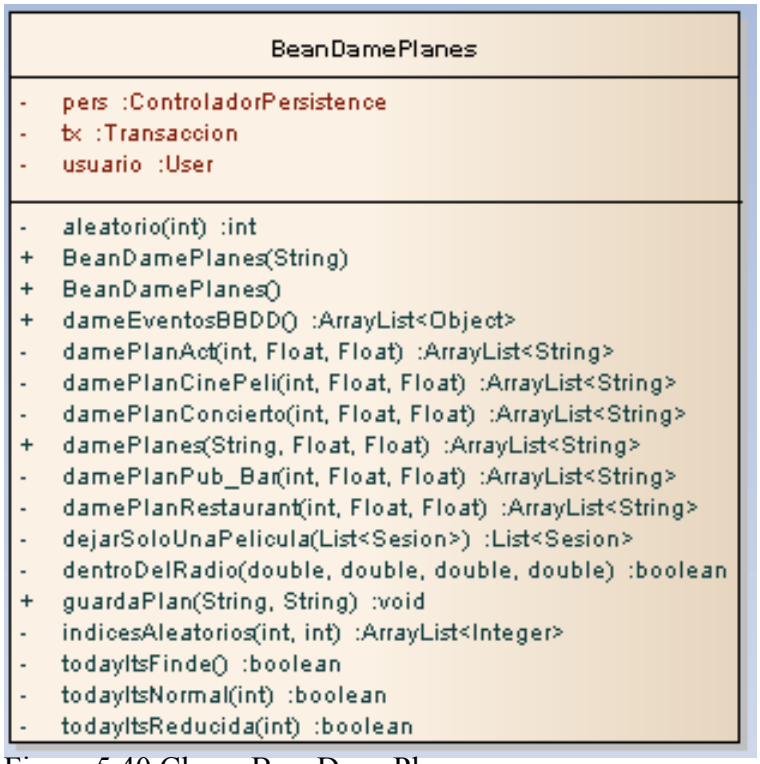

Figura 5.40 Classe BeanDamePlanes

BeanDatosLugar.java: Bean encarregat de realitzar diferents consultes enfocades en les dades dels llocs.

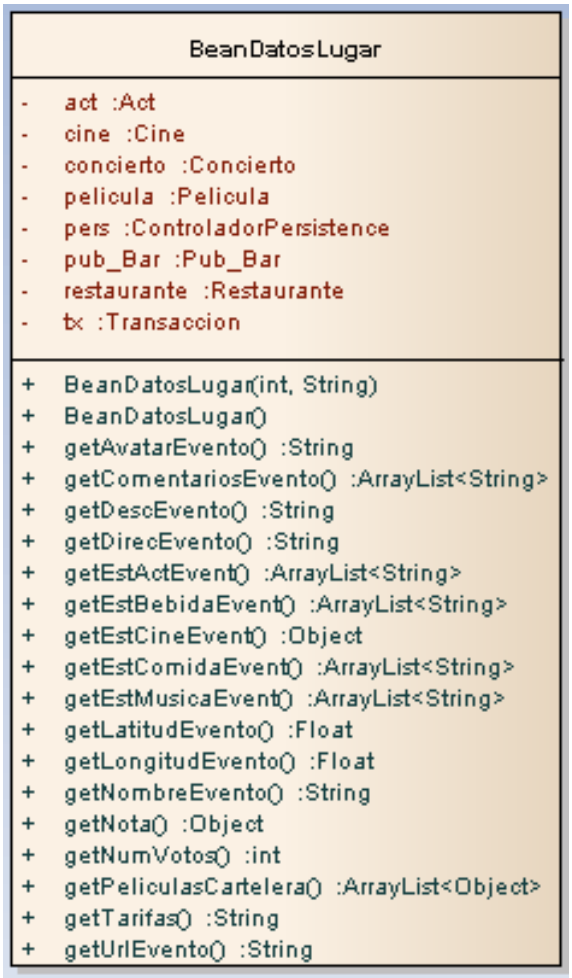

Figura 5.41 Classe BeanDatosLugar

BeanIntroHecho.java: Bean encarregat d'introduir a l'agenda de l'usuari un pla que ja ha realitzat i si el lloc on ha estat no existeix a la nostre Base de Dades, donar-ho d'alta.

- buscaSimilesNombre(...): Aquest mètode segueix el mateix funcionament que getAmigosSimilares() de la classe BeanAgregarAmigo(). Busca llocs els quals el seu nom sigui igual al que ha introduït l'usuari. O bé, s'hi aproximi. El criteri d'ordenació serà la quantitat de coincidències que té el text introduït per l'usuari amb el registre de la Base de Dades i és marcat per la classe OrdenaPorCoincidencias.java.
- addHechoHoy(…): Aquest mètode insereix a l'agenda de l'usuari l'event ja realitzat.

  El lloc on ha estat ja existeix a la nostre Base de Dades.

  Un cop inserit el nou registre a l'Agenda es comprova si l'estil d'aquest lloc no es troba dintre dels estils definits per l'usuari (crida a mètode

comprovaSiNuevosEstilos(…)), si no el té definit, se l'afegirà automàticament. D'aquesta manera, a pròximes propostes apareixeran llocs semblants al que ja ha estat o el mateix.

Els mètodes addHechoHoyActNueva(...), addHechoHoyConcNuevo(...), addHechoHoyPBNuevo(...), addHechoHoyPeliNueva(...), addHechoHoyRestNuevo(...) són definits per aquells plans d'usuari realitzats, dels quals el lloc no figura a la nostre Base de Dades. Per tant, aquests mètodes són els encarregats de crear el registre.

BeanLogin.java: Bean que inicia la sessió de l'usuari a l'aplicació.

Figura 5.42 Classe BeanIntroHecho

Figura 5.43 Classe BeanLogin

BeanPerfil.java: Bean utilitzat per la gestió del perfil d'usuari.

Figura 5.44 Classe BeanPerfil

BeanRegis.java: Bean que dona d'alta un usuari.

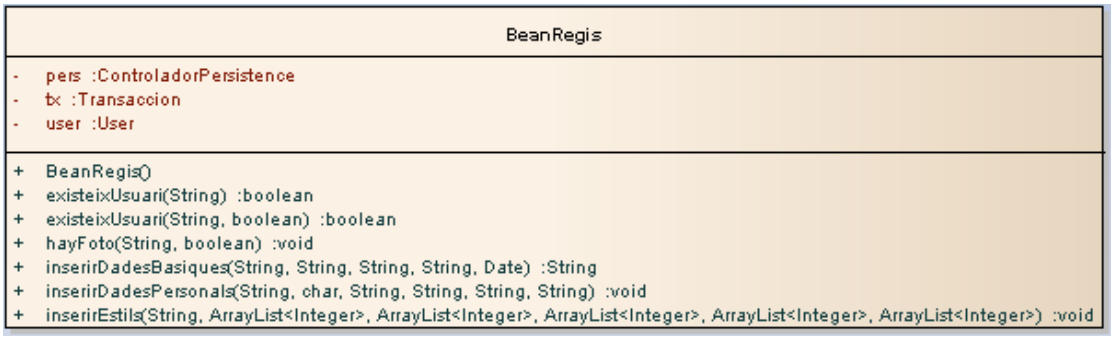

Figura 5.45 Classe BeanRegis

BeanValoraciones.java: Bean utilitzat per realitzar les valoracions dels usuaris, ja siguin puntuacions o comentaris.

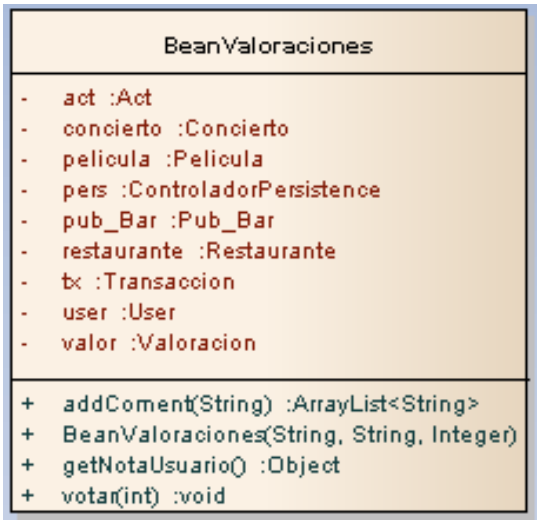

Figura 5.46 Classe BeanValoraciones

5.1.4. Struts (Actions + ActionForm).

- Classes ActionForm:

Totes les classes acabades amb nom "Form" són extensió de la classe ActionForm de Java. Per tant, és a les quals arriben les dades enviades a la petició del client, tal com es

configura al Controlador. La funció d'aquestes classes és rebre aquestes dades i passar-les a la classe "Action" amb la que està relacionada.

Aquestes classes també contenen un mètode destinat a la validació de les dades rebudes. A l'única classe a la que es troba implementat és a LogInForm.java. D'aquesta manera es crida a la classe BeanLogIn.java per saber si l'usuari i contrasenya rebuts a la petició són vàlids per poder realitzar l'inici de sessió. En cas negatiu, es recollirà l'excepció rebuda de BeanLogIn.java i es retornarà un objecte ActionErrors que contindrà l'identificador del missatge que s'ha de mostrar a la pàgina JSP de la Vista.

Per configurar els continguts d'aquests missatges s'ha d'anar al fitxer ApplicationResource.properties que es troba al paquet com.myapp.struts:

Aquest és el contingut afegit al fitxer per els errors que es contemplen a l'inici de sessió:

```
errors.log=Usuario incorrecto
errors.log2=Contraseña erronea
```

Figura 5.47 Contingut per fitxer errors

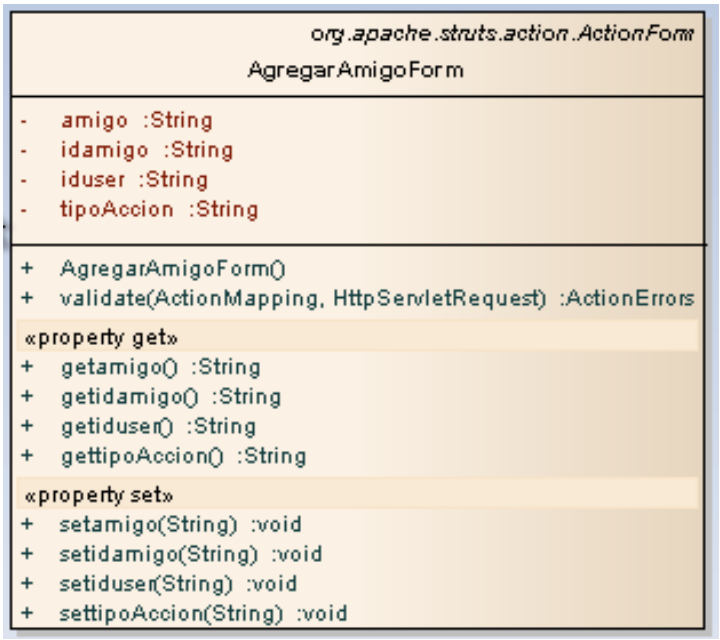

Figura 5.48 Classe AgregarAmigoForm

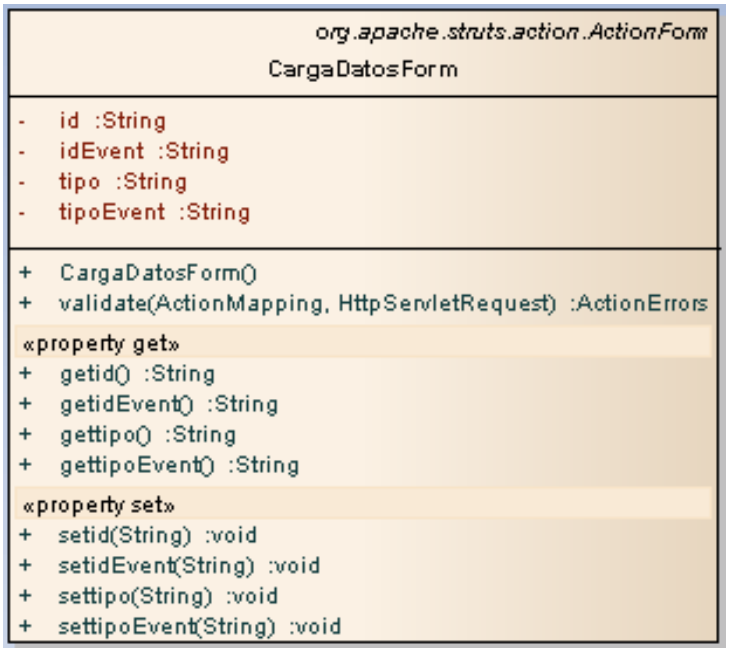

Figura 5.49 Classe CargaDatosForm

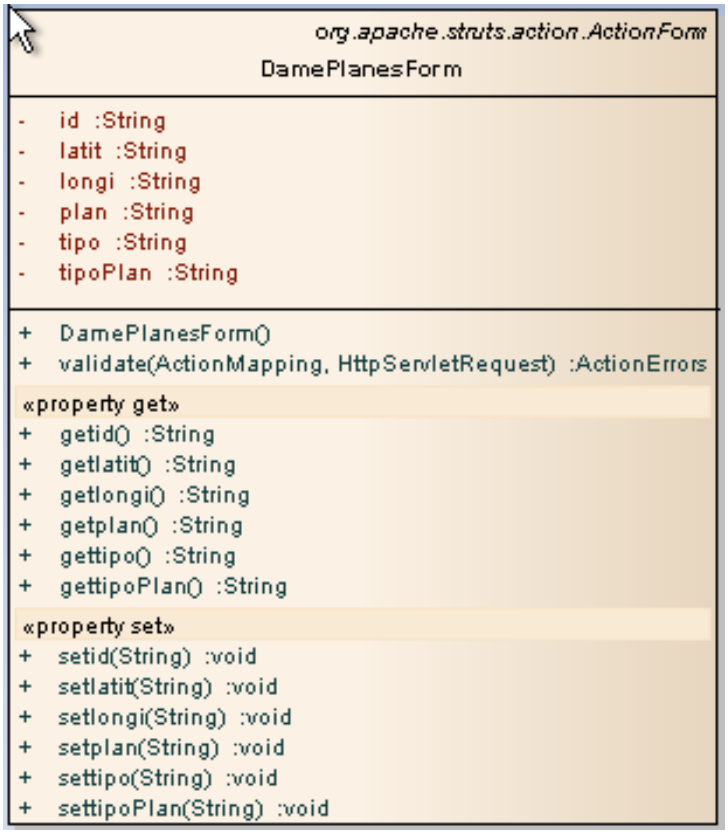

Figura 5.50 Casse DamePlanesForm

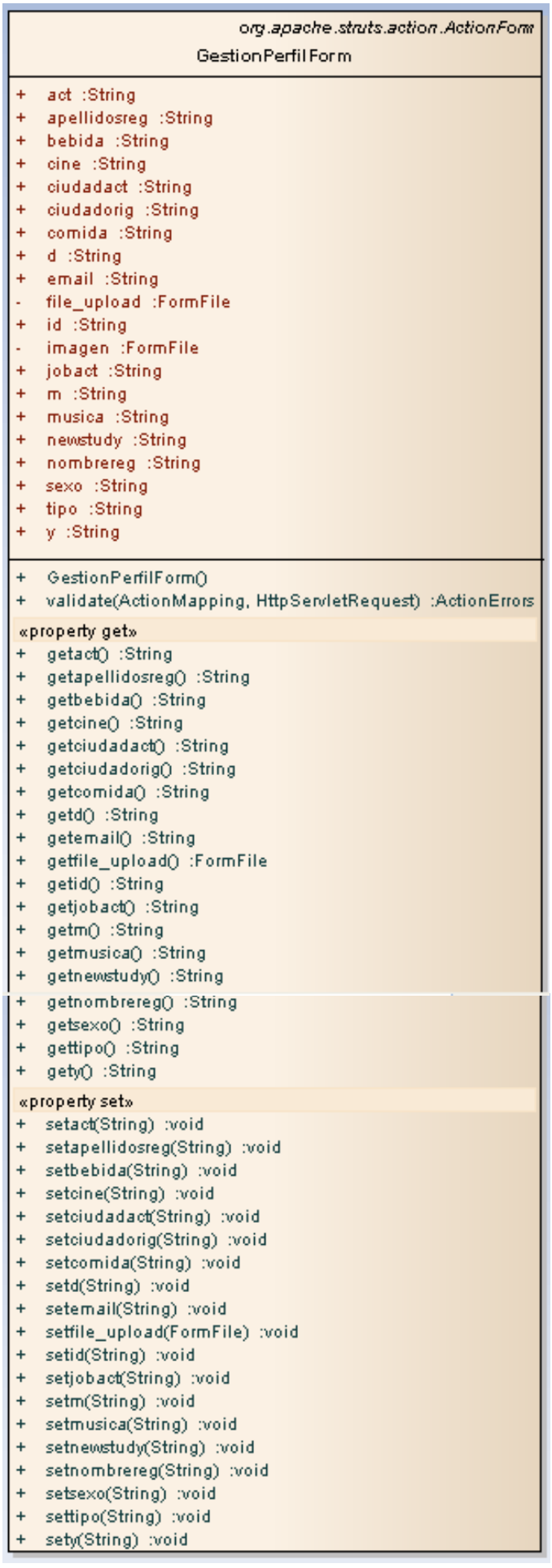

Figura 5.51 Classe GestionPerfilForm

Figura 5.52 Classe IntroduceHechoForm

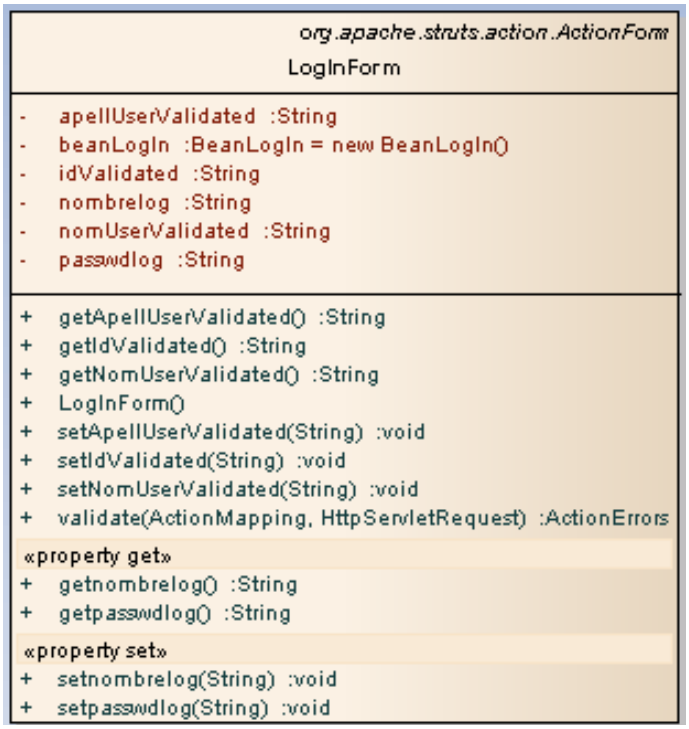

Figura 5.53 Classe LogInForm

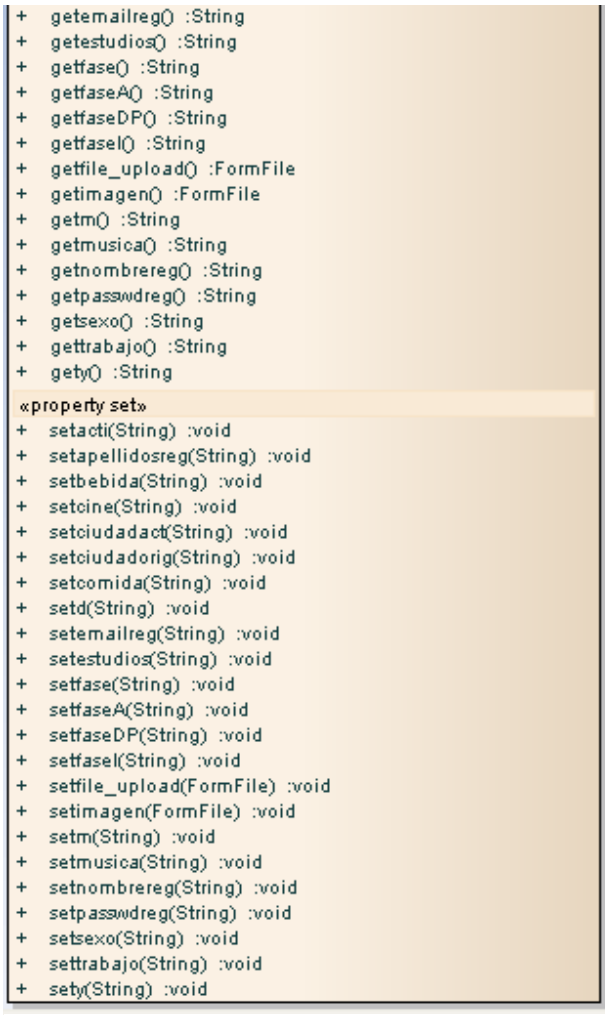

Figura 5.54 Classe RegistroForm

Figura 5.55 Classe ValorarForm

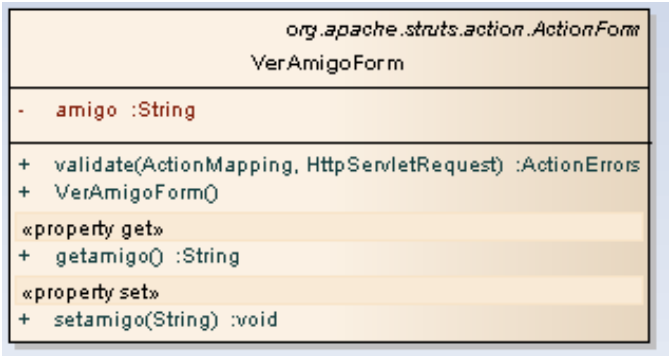

Figura 5.56 Classe VerAmigoForm

- **Classes Action**:

Aquestes classes manipulen les classes ActionForm i amb les dades obtingudes criden a la classe del paquet Bean corresponent per a portar a terme el cas d'ús.

Si s'han de mostrar dades a la vista, aquestes dades vindran de les crides a funcions del paquet Bean i seguidament, s'inseriran a la sessió a peticions de client fetes per formulari, o bé, es generarà un objecte JSON per peticions fetes via AJAX.

Inserir dades a la sessió:

```
HttpSession sessio = request.getSession();
sessio.setAttribute("ciudadAct", beanDameDatos.getCiudadAct());
sessio.setAttribute("email", beanDameDatos.getEmail());
sessio.setAttribute("edad", beanDameDatos.getEdad());
sessio.setAttribute("sexo", beanDameDatos.getSexo());
sessio.setAttribute("estudios", beanDameDatos.getEstudios());
sessio.setAttribute("trabajo", beanDameDatos.getTrabajo());
```

Figura 5.57 Exemple d'inserció de dades a la sessió

Retornar dades mitjançant objecte JSON:

És necessari utilizar la llibreria json-lib.jar.

```
ArrayList<String> amic = beanAgregarAmigo.getAmigosSimilares(
        agregarAmigo.getamigo(), agregarAmigo.getiduser());

HashMap hm = new HashMap();

hm.put("amic", amic.toArray());

JSONObject json = JSONObject.fromObject(hm);
if (json != null) {
    response.setContentType("text/x-json;charset=UTF-8");
    response.setHeader("Cache-Control", "no-cache");
    try {
        json.write(response.getWriter());
    } catch (IOException e) {
        throw new Exception("IOException in populateWithJSON", e);
    }
}
```

Figura 5.58 Retorn de dades amb la llibreria json-lib

A aquest exemple, la crida del mètode de l'objecte beanAgregarAmigo retorna un ArrayList de tipus String necessari per ser rebut a la Vista.

S'introdueix aquest ArrayList dintre d'un HashMap i es crea un objecte JSONObject passant com a paràmetre aquest HashMap.

A la resposta s'indica el ContentType com "text/x-json;charset=UTF-8" i s'escriu a la resposta l'objecte json.

Redirecció a pàgina:

La classe action és l'encarregada d'indicar al Controlador a quina pàgina de la vista s'ha de dirigir.

A retorns de variables a sessió s'indicarà el nom que identifica la pàgina tal com s'ha configurat al Controlador.

```
return mapping.findForward("event");
```

Figura 5.59 Redirecció a vista estàndard

En canvi, a retorns JSON s'indicarà un retorn null ja que no es vol tornar a carregar cap pàgina per que és una petició AJAX.

```
return mapping.findForward(null);
```

Figura 5.60 Redirecció a vista utilitzant objectes Json

5.2. Controlador.

El Controlador de l'aplicació és l'arxiu de configuració d'struts. Al qual s'indica quina classe ha de rebre els paràmetres que arriben de la petició del client i quina classe s'ha d'executar per continuar amb la lògica del programa. Per tant, és l'encarregat de distribuir la feina.

5.2.1. Struts-config.xml.

L'arxiu de configuració es troba a la carpeta WEB-INF dintre del directori WEB.

```
<?xml version="1.0" encoding="UTF-8" ?>

<!DOCTYPE struts-config PUBLIC
          "-//Apache Software Foundation//DTD Struts Configuration 1.3//EN"
          "http://jakarta.apache.org/struts/dtds/struts-config_1_3.dtd">

<struts-config>

    <form-beans>
       <form-bean name="LogInForm" type="Struts.LogInForm"/>
       <form-bean name="RegistroForm" type="Struts.RegistroForm"/>
       <form-bean name="CargaDatosForm" type="Struts.CargaDatosForm"/>
       <form-bean name="GestionPerfilForm" type="Struts.GestionPerfilForm"/>
       <form-bean name="VerAmigoForm" type="Struts.VerAmigoForm"/>
       <form-bean name="DamePlanesForm" type="Struts.DamePlanesForm"/>
       <form-bean name="ValorarForm" type="Struts.ValorarForm"/>
       <form-bean name="IntroduceHechoForm" type="Struts.IntroduceHechoForm"/>
       <form-bean name="AgregarAmigoForm" type="Struts.AgregarAmigoForm"/>
    </form-beans>
```

Figura 5.61 Arxiu de configuració d'Struts

Dintre del tag form-beans s'afegiren totes les classes ActionForm que hauran de recollir els paràmetres de la petició.

```
<action-mappings>
    <action path="/logeando" type="Struts.LogInAction" name="LogInForm"
             scope="session" validate="true" input="/index.jsp">
            <forward name="inicio" path="/inicio.jsp"/>
    </action>
```

Figura 5.62 Definició d'un action

Dintre del tag action-mappings es configura l'strut.

Path: la url per cridar a l'strut.

Type: L'Action encarregat.

Name: ActionForm encarregat.

Input: La pàgina JSP d'on s'és cridat

Forward: Un cop finalitzada la lògica, a on es redirigirà.

Name: nom identificador que s'utilitza a la crida mapping.findForward(nom) de l'Action.

Path: la url de la pàgina jsp a la qual ens dirigim.

5.3. Vista.

5.3.1. Pàgines JSP.

La web està formada per 6 pàgines diferents:

agenda.jsp: Pàgina on es visualitza l'agenda de l'usuari amb tots els esdeveniments registrats fins el moment.

evento.jsp: Pàgina on es troba la informació sobre un lloc, la seva valoració (puntuació més comentaris) i la ubicació, amb un mapa de Google Maps.

index.jsp: Pàgina principal de l'aplicació. Des d'aquí es pot iniciar sessió o bé registrar-se com un usuari nou aportant les dades bàsiques.

inicio.jsp: Pàgina principal un cop s'ha iniciat la sessió. Aquí es trobarà: la llista d'amistats, els últims esdeveniments ocorreguts, els llocs més visitats per les amistats, la possibilitat d'introduir on s'ha estat avui i el servei de propostes.

perfil.jsp: Pàgina on es veuen les dades d'un usuari, el seu perfil i els llocs que més visita. Aquesta pàgina és compartida per el perfil propi de l'usuari (que té l'opció d'editar les dades) i la pàgina de dades d'una amistat (amb l'opció d'edició amagat).

registro.jsp: Pàgina del registre extens de l'usuari. Aquí indicarà els seus gustos i si ho desitja, una imatge de perfil.

Per mostrar un paràmetre de la sessió a una pàgina ".jsp" cal indicar la sentència:

```
title="Tus amigos (${sessionScope.numAmigos})"
```

Figura 5.63 Com escriure una dada de la sessió desde JSP

Amb ${sessionScope.nomDelParàmetre} s'escriurà el valor d'aquest paràmetre a la pàgina.

Per mostrar els muntatges d'ArrayList tipus String intercalats que s'han fet a alguns mètodes del Model, s'explica el següent exemple:

```
<div id="pantVerTodosAmigos" title="Tus amigos (${sessionScope.numAmigos})" style="display:none;">
    <table border="0" cellpadding="1" cellspacing="0" align="center">
        <form id="cargaAmigo2" name="cargaAmigo2" action="verAmigo.do" method="post">
            <c:set var="it_A" value="0"/>
            <c:forEach items="${sessionScope.todosAmigos}" var="t_ami">
                <c:if test="${it_A==0}">
                    <tr style="cursor:pointer" onclick="cargaAmigos2('${t_ami}')">
                    </c:if>
                    <c:if test="${it_A==1}">
                        <td>
                            <img src='${t_ami}' id="noimage" width="25"
                                 height="25" alt="" />
                        </td>
                    </c:if>
                    <c:if test="${it_A==2}">
                        <td id="textoamigo" class="hoverGris">${t_ami}</td>
                        <input type="hidden" id="amigo2" name="amigo" value=""/>
                    </tr>
                </c:if>
                <c:set var="it_A" value='${it_A+1}'/>
                <c:if test="${it_A==3}">
                    <c:set var="it_A" value="0"/>
                </c:if>
            </c:forEach>
        </form>
    </table>
</div>
```

Figura 5.64 Exemple d'impressió de muntatge ArrayList a pàgina JSP

Aquest contenidor, mostra tots els amics d'un usuari. Per cada ítem de l'atribut "todosAmigos" es mostra un tros de codi HTML. Com se sap que les diferents dades estan intercalades, es controla amb índexs a quina dada es correspon cada ítem. A aquest cas, a la posició 1 sempre vindrà l'identificador de l'amic (necessari per redigir-se a la seva pàgina de perfil si es prem a ell), a la posició 2 vindrà la direcció on es troba la seva imatge de perfil i finalment, al tercer lloc ve el seu nom. Quan ja s'han mostrat les tres dades, s'inicialitza el comptador d'índexs per continuar amb un nou amic.

5.3.2. JavaScript.

S'ha guanyat molta dinàmica gràcies al framework JQuery. Per poder utilitzar-ho cal afegir a la capçalera de la pàgina la seva referència:

<script src="jquery/jquery-1.4.1.min.js" type="text/javascript"></script>

Principals funcionalitats implementades amb Javascript:

- Accés a components del DOM senzillament amb la sentència: $("#nomId") o $(".nomClase").
- Afegir codi html a la pàgina amb el mètode append(). Per exemple:

```
$("#listaPlanes").append(
    "<div class=\"propGifSinLetras\" "+
    " id=\"cargandoProp\" style=\"margin-left:25%;margin-right:25%\"></div>");
```

Figura 5.65 Mètode append de JQuery

S'està afegint al component amb identificador "listaPlanes" un etiqueta "div" que conté un arxiu animat amb extensió ".gif". Aquest exemple és útil per que l'arxiu animat s'afegeix abans d'una petició AJAX al servidor que podrà mantenir a l'espera a l'usuari. D'aquesta manera l'usuari percebrà que ha d'esperar.

- Petició al servidor via AJAX amb la funció $.ajax(). Exemple:

```
$.ajax({ //Comunicación jQuery hacia Strut
    type: "POST",
    url: "damePlanes.do",
    data: "tipo="+tipo+"&"+
    "id="+$("#idUser").val()+"&"+
    "latit="+latit+"&"+
    "longi="+longi,
    dataType: "json",
    success: function(jqXHR){
        $("#cargandoProp").remove();
        for (var j=0;j<5;j++){
            $("#tablaPlanes").html("");
            $("#textoAfinar").remove();
        }
        //alert("");
        if (jqXHR.planes.length==0){
            $("#tablaPlanes").append(
                "<tr>"+
                "<td>"+"No hay resultados que se afinen a tu situaci&oacuten y"+
                " gustos. Lo sentimos."+
                "</td>"+
                "</tr>"+
                "<tr>"+
                "<td>Si lo deseas, ve a tu perfil y danos mas informacion "+
                "sobre tus intereses!</td>"+
                "</tr>");
        }
        var metidos = 0;
        for (var i=0;i<jqXHR.planes.length;i+=7){
            //alert("");
            if (metidos > 0){
                $("#tablaPlanes").append(
                    "<tr>"+
                    "<td width=\"400px\" colspan=\"6\" class=\"linea_discontinua\" "+
                    "height=\"15\" >"+
                    "</td>"+
                    "</tr>");
            }
            $("#tablaPlanes").append(
                "<tr>"+
                "<td onclick=\"guardaTipo('"+jqXHR.planes[i+6]+"',"+
                jqXHR.planes[i+2]+")\">"+
                "<input type=\"radio\" name=\"seleccion\" "+
                "value=\""+jqXHR.planes[i+2]+"\" /></td>"+
                "<td class=\"texto\">"+
                "<img src="+jqXHR.planes[i] + " id=\"noimage\" "+
                "width=\"25\" height=\"25\" alt=\"\"/>"+
                "</td>"+
                "<td class=\"texto\"> </td>"+
                "<td class=\"texto hoverGris\" colspan=\"2\" "+
                "title=\"Mas informacion\" "+
                "style=\"cursor:pointer\" "+
                "onclick=\"goToEventoOpenWindow('"+jqXHR.planes[i+6]+"', "+
                jqXHR.planes[i+2]+")\">"+
                jqXHR.planes[i+1]+"</td>"+
                "</tr>"+
                "<tr>"+
                "<td class=\"texto\" colspan=\"3\"></td>"+
                "<td colspan=\"2\"> "+jqXHR.planes[i+3]+"</td>"+
                "</tr>"+
                "<tr>"+
                "<td class=\"texto\" colspan=\"3\"></td>"+
                "<td colspan=\"2\"> "+jqXHR.planes[i+4]+"</td>"+
```

```
                "</tr>"+
                "<tr>"+
                "<td class=\"texto\" colspan=\"3\"></td>"+
                "<td colspan=\"2\"> "+jqXHR.planes[i+5]+"</td>"+
                "</tr>");
            metidos++;
        }

        $("#listaPlanes").append(
            "<table id=\"textoAfinar\">"+
            "<tr><td><br></br></td></tr>"+
            "<tr id=\"afinar\" style=\"cursor:pointer\">"+
            "<td class=\"texto\" colspan=\"3\">" +
            "<input type=\"hidden\" id=\"tipoSelec\" value=\"\"/>"+
            "<input type=\"hidden\" id=\"elegido\" value=\"\"/>"+
            "</td>"+
            "<td class=\"negrita\" onclick=\"sacaParam()\">Afinar el resultado</td>"+
            "</tr>"+
            "</table>");
    },
    error: function(xml,msg){
        alert("error");
        $("#cargandoProp").css("display", "none");
    }
});
```

Figura 5.66 Exemple de crida a servidor via AJAX

Atributs:

type: Indica el mètode de comunicació (GET o POST).

url: path de l'action al qual volem cridar.

data: paràmetres que es volen passar al servidor i que rebrà l'ActionForm. Si hi ha varis, s'han de separar per "&".

Si alguna de les dades a enviar es tracta d'una taula, s'utilitza la funció JSON.stringify(nom del camp a convertir) per convertir-la a cadena.

datatype: es tracta amb objectes Json.

success: funció que es vol que s'executi una vegada el servidor ha acabat amb la seva feina. Aquesta funció recull l'objecte JSON que el servidor ha creat i el tracta. A aquest exemple s'ha recollit les propostes d'oci que el servidor ha generat.

En primer lloc, s'esborra l'arxiu animat que s'havia afegit a mode d'espera amb el mètode remove(). S'inicialitza la taula on s'afegeixen els plans amb el mètode html(). D'aquesta manera s'han esborrat totes les seves files i columnes. Si l'objecte JSON "jqXHR.planes" té mida zero voldrà dir que no s'han trobat propostes, per tant, s'afegirà amb el mètode append() una fila a la taula amb identificador "tablaPlanes" que informi del succés. En cas contrari, per cada ítem de l'array de l'objecte JSON s'afegirà la informació corresponent amb el mateix muntatge de dades intercalades que s'utilitza a les pàgines JSP.

error: si el servidor no ha finalitzat la seva feina amb èxit i s'ha produït una excepció, s'executarà aquesta funció on s'hi pot afegir la lògica desitjada. En aquest cas, simplement

es mostrarà un missatge d'error per que es suposa que al servidor ha d'estar tot controlat per que sempre funcioni.

- Control d'events. Per exemple: si un usuari prem sobre un element, es pot definir una funció a realitzar. En comptes de marcar a la pàgina JSP la sentència onClick= "".

```
$("#cines").click(function () {
    obtenPlanes("cines");
});
```

Figura 5.67 Exemple de capturació d'esdeveniments amb JQuery

- Finestres de diàleg amb estil utilitzant l'extensió ui de JQuery. El component s'anomena **dialog**.

  Per utilitzar aquesta extensió de JQuery s'ha d'afegir a la capçalera de la pàgina i també l'arxiu CSS.

```
<script type="text/javascript" src="jsCalendar/jquery-ui/smoothness/jquery-ui-1.8.1.custom.min.js"></script>
<link rel="stylesheet" type="text/css" href="cssCalendar/jquery-ui/smoothness/jquery-ui-1.8.1.custom.css" />
```

  Per declarar que una etiqueta serà un diàleg és necessari el següent codi JavaScript:

```
$("#planes").dialog({
    autoOpen: false,
    width:500,
    modal: true,
    buttons: {
        'Go!': function() {
            aceptoPlan();
            $(this).dialog('close');
            $("#parametros").fadeOut();
        },
        'No me interesa ninguno...': function() {
            $(this).dialog('close');
            $("#parametros").fadeOut();
        }
    },
    close: function() {
        // reset form elements when we close so they
        $("#parametros").fadeOut();
    }
});
$('#planes').dialog('open');
```

Figura 5.68 Declaració de diàlegs amb JQuery

Atributs:

autoOpen: el valor false, diu que s'obrirà quan s'indiqui.

Width: amplada de la finestra.

Modal: el valor true farà que la resta de la pàgina s'enfosqui, donant així més visibilitat al dialog.

Buttons: botons que tindrà la finestra i les funcions que realitzarà cadascun.

Open: funció que s'ha de realitzar en el moment que s'obri el diàleg. No indicar aquest paràmetre voldrà dir que no s'ha de fer res.

Close: funció que s'ha de realitzar quan es tanqui el diàleg.

Tancar un diàleg automàticament: $(this).dialog('close') (si estem dintre del mateix dialog) o bé $("#identificador").dialog('close').

Per obrir un diàleg s'ha de procedir exactament igual que per tancar-ho, simplement canviant el text "close" per "open".

- Efectes. Es pot fer aparèixer o desaparèixer elements amb un efecte de fos utilitzant les funcions fadeIn() i fadeOut(). Si no es vol utilitzar cap efecte i simplement es vol fer desaparèixer o aparèixer un element, s'utilitzen les funcions show() i hide().
- Assignar atributs a components. Útil per assignar valors a camps ocults que s'utilitzan per enviar informació al servidor. Amb el mètode attr(), es pot modificar qualsevol atribut d'un component del DOM indicant el nom de l'atribut i el seu valor.

```
function guardaTipo(tipoSel, seleccion){
    $("#tipoSelec").attr("value", tipoSel);
```

Figura 5.69 Exemple d'assignació de valor a camps del DOM amb JQuery

- Obtenir valors de components. La funció val() retorna el valor d'un component.

```
var tipoSelec = $("#tipoSelec").val();
var elegido = $("#elegido").val();
```

Figura 5.70 Exemple de consulta de valor a camps del DOM amb JQuery

- Obtenir ubicació de l'usuari. Navigator és un atribut de l'API que ha implementat l'W3C per la versió d'HTML 5.

```
$(document).ready(function(){
    navigator.geolocation.getCurrentPosition(function(position) {
        latit=position.coords.latitude;
        longi=position.coords.longitude;
    });
});
```

Figura 5.71 Exemple de consulta de la ubicació de l'usuari

- Mostrar un mapa de Google Maps amb l'API de Google. Aquest exemple mostra el mapa posicionat amb les coordenades obtingudes per l'ubicació actual de l'usuari. Es genera un objecte anomenat coordenades de la classe google.maps.LatLng.

```
var latlng = new google.maps.LatLng(latit,longi);
var myOptions = {
    zoom: 12,
    center: latlng,
    mapTypeId: google.maps.MapTypeId.ROADMAP
};

var map = new google.maps.Map(document.getElementById("map"), myOptions);

var oMarker = new google.maps.Marker({
    'position': latlng,
    'map': map,
    'title': 'Here'
});

var texto="<span style='margin-left:40%;>"+
"¡Est&aacutes aqu&iacute!"+
"<br/></span>";

var infoWindow = new google.maps.InfoWindow({
    content: texto
});
google.maps.event.addListener(oMarker, 'click', function () {
    infoWindow.open(map,oMarker);
});
google.maps.event.addListener(map, 'click', function(event) {
    placeMarker(event.latLng, map);
});
```

Figura 5.72 Exemple d'inserció de mapa amb l'API de Google Maps

A continuació s'indiquen les opcions del mapa: que comenci amb un zoom de 12, posicionat a les coordenades creades i que el tipus de mapa sigui el tipus de mapa 2D predeterminat de Google Maps.

Hi ha els següent tipus de mapa possibles:

ROADMAP: ja explicat.

SATELLITE: imatges de satèl·lit.

HYBRID: barreja de mosaics fotogràfics i una capa de mosaics per els elements del mapa més destacats.

TERRAIN: mosaics de relleu físic per indicar les elevacions del terreny i les fonts d'aigua.

Un cop creades totes les funcions es crea el mapa passant-li com a paràmetre l'etiqueta "div" que farà de contenidor i les opcions definides.

També s'afegirà un marcador indicant l'ubicació actual de l'usuari creant un objecte de la classe google.maps.Marker. A aquest se li passarà les coordenades on ha d'aparèixer, el mapa i si es desitja, un títol.

Per donar més informació, es crea una finestra que apareixerà al clickar al marcador generat, creant un objecte de la classe google.maps.InfoWindow.

A aquest exemple és necessari l'activació d'events sobre el mapa, per que el que es desitja és que l'usuari premi a un punt del mapa i obtenir aquestes coordenades. Per fer això es crida a la funció addListener, a la qual se li passa el mapa que reconeixerà els events i el tipus d'event necessari (en aquest cas: click) i la funció que s'ha de realitzar quan es premi: en el nostre cas, generar un nou marker amb la posició indicada per l'usuari, la qual s'obté gràcies a l'event.

Per afegir aquest nou marker s'ha creat una nova funció que abans esborrarà els markers que hagi afegit l'usuari anteriorment, per que només es desitja un. Això es pot fer, assignant-li un mapa null al marcador que es vol esborrar.

- Visor d'imatges. S'ha utilitzat un plugin de JQuery anomenat "yoxview", el qual només necessita afegir la ruta de l'arxiu a la capçalera de la pàgina.

<script type="text/javascript" src="yoxview/yoxview-init.js"></script>

```
$("#imageperfil").click(function(){
    $("#thumbnails").yoxview({
        backgroundColor: '#333333',
        playDelay: 5000,
        lang:'es'
    });
});
```

Figura 5.73 Exemple d'utilització del plugin YoxView de JQuery

Per activar el plugin sobre la imatge només cal definir el seu mètode click (prémer) i quan es produeixi l'event, es generarà un visor amb el color, idioma i opacitat que s'indiqui.

- Obrir nova finestra. Quan l'usuari es troba dintre d'un dialog i té la possibilitat de navegar a altres pàgines, és necessari que es generi una nova finestra, ja que si no, no es podrà recuperar el diàleg que estava visualitzant, si així es desitja.

```
window.open("nueva.html?tipo="+tipo+"&tipoEvent="+tipoEvent+"&idEvent="+sitio);
```

Figura 5.74 Exemple de generació de nova finestra

Per fer-ho es crida al mètode window.open que redirigirà a una pàgina en blanc (amb els paràmetres necessaris) per mètode GET. Aquesta pàgina en blanc serà l'encarregada de cridar (via formulari) al servidor i aquest, mostrar la pàgina demanada.

- Plugin agenda: Els arxius necessaris per la seva utilització són:

  CSS:

```
<link rel="stylesheet" type="text/css" href="cssCalendar/frontierCalendar/jquery-frontier-cal-1.3.2.css" />
```

  JavaScript:

```
<script type="text/javascript" src="jsCalendar/lib/jshashtable-2.1.js"></script>
<script type="text/javascript" src="jsCalendar/frontierCalendar/jquery-frontier-cal-1.3.2.js"></script>
```

- Plugin pujada de fitxers: Els arxius necessaris per la seva utilització són:

  CSS:

```
<link rel="stylesheet" type="text/css" href="./uploadify/uploadify.css" media="all" />
```

  JavaScript:

```
<script src="uploadify/swfobject.js" type="text/javascript"></script>
<script src="uploadify/jquery.uploadify.v2.1.4.min.js" type="text/javascript"></script>
```

S'anomena al camp fitxer "file_upload" i mitjançant aquest identificador se li assignarà el plugin yoxviex.

```
$(document).ready(function(){
    $("#file_upload").uploadify({
        "uploader":"uploadify/uploadify.swf",
        "script":"registro.do",
        "cancelImg":"uploadify/cancel.png",
        "fileDataName":"file_upload",
        "buttonText":"Selecciona la foto",
        "sizeLimit":"512000",
        "folder":"imagenes",
        "auto":true,
        'onProgress':function(){
            $('#loader').show;
        },
        'onComplete':function(event, queueID, fileObj, response, data){
            var fichero=parseaResponse(response);
            $("#imageperfil").fadeOut();
            $("#filesUploaded").append("<a href="+"imagenes/"+fichero+">"+
                "<img src="+"imagenes/"+fichero+" width=\"200\" height=\"233\" />"+
                fileObj.name+"</a><br>");
            $("#filesUploaded").fadeIn("slow");
            $('#loader').hide;
        }

    })
})
```

Figura 5.75 Exemple d'utilització del plugin Uploadify de JQuery

Opcions:

Uploader: indica l'animació flash que s'executarà mentrestant s'està pujant al servidor el fitxer.

Script: l'strut al qual s'ha de cridar.

cancelImg: imatge per cancel·lar la pujada del fitxer.

fileDataName: nom del fitxer

buttonText: text que es desitja que aparegui al botó que obre l'explorador d'arxius.

sizeLimit: tamany màxim que es tolera per fitxer.

Folder: el directori on s'emmagatzemarà el fitxer.

Auto: true indicarà que un cop seleccionat el fitxer de pujada, la pujada començarà automàticament.

onProgress: funció es vol que es realitzi mentrestant s'està pujant el fitxer.

onComplete: funció que es vol que es realitzi una vegada ha finalitzat el treball del servidor.

5.3.3. CSS.

Hi ha una gran quantitat d'estils aplicats a totes les pàgines, per tant, només es passarà a comentar l'estructura compartida per totes elles.

Totes les pàgines JSP comparteixen la mateixa estructura:

- Una capçalera amb el logotip de l'aplicació i un menú de navegació.

La capçalera estarà fixada a la part superior i tindrà un fons que es tracta d'una imatge petita que s'anirà repetint per tota l'amplada de la pantalla de manera que l'ompli.

```
#cabe {
    width:100%;
    height:44px;
    background-image: url(img/png/bag2.png);
    background-repeat: repeat-x;
    background-position: center;
    z-index:4;
    top:0%;
    position:fixed;
}
```

Figura 5.76 Definició de capçalera amb CSS

- El contingut de la pàgina. El contingut de la pàgina es troba dintre d'una taula que fa de contenidor i a la qual se li ha aplicat un estil segons la columna que es tracti, de manera que les columnes del cantó dret i esquerre i tota la fila inferior, tenen aplicat un estil d'imatge que engloba tot el contingut:

- El peu de pàgina. Seguint el mateix patró que a la capçalera de la pàgina, s'utilitzarà una imatge petita com a fons que s'anirà repetint per tot l'ample de la pantalla. En aquest cas es vol posicionar a la part baixa de la finestra i que no sigui fix, és a dir, si la finestra creix i apareix l'scroll de pàgina, el peu haurà d'aparèixer quan s'arriba al final de l'scroll.

```
#pie {
    width:100%;
    height:23px;
    text-align:center;
    background-image: url(img/png/pie.png);
    background-repeat: repeat-x;
    position:relative;
    bottom:0%;
    z-index:4;
    font-weight: bold;
}

#pie td{
    padding-top:6%;
}
```

Figura 5.77 Definició de peu de pàgina amb CSS

6. Manual d'usuari.

A continuació es passarà a descriure com fer ús de totes les funcionalitats implementades a l'aplicació.

6.1. Accés a l'aplicació.

WeDugu es pot trobar a la direcció:

http://wedugu.no-ip.org

Si l'usuari no havia estat identificat prèviament, apareixerà la següent pàgina:

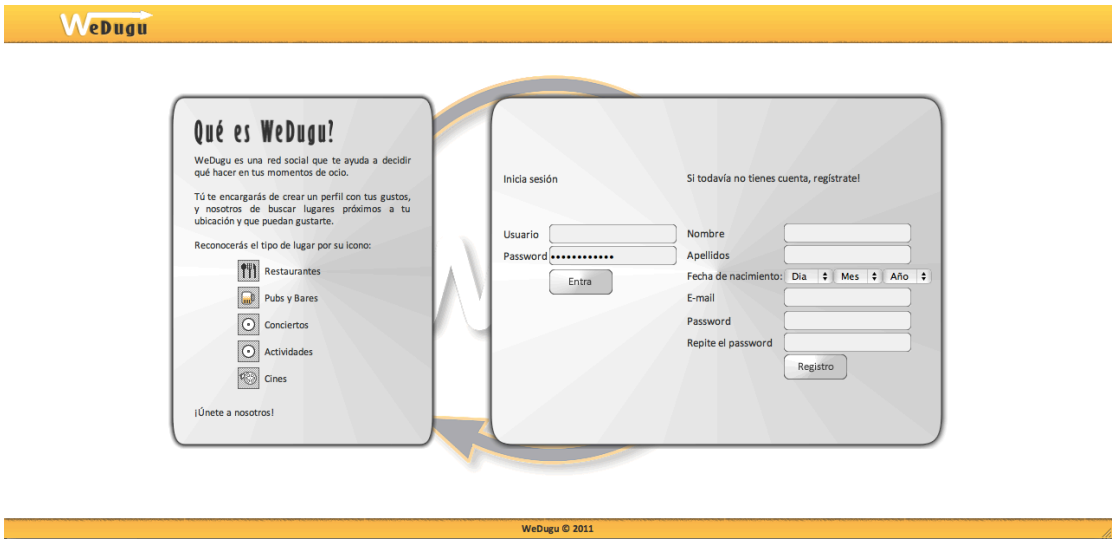

Figura 6.1 Pàgina inici

6.1.1. Iniciar sessió.

Per iniciar sessió s'haurà d'introduir l'e-mail amb el que es va fer el registre i la contrasenya corresponent. Si s'han utilitzat unes credencials correctes es redirigirà a la següent pàgina:

Figura 6.2 Pàgina amb sessió iniciada

6.1.2. Alta d'usuari.

El procés de registre de nous usuaris consta de tres fases. La primera és obligatòria mentre que les dos següents són voluntàries i permetent completar el perfil del nou usuari.

Fase 1:

Si un usuari nou vol registrar-se al web, haurà d'introduir totes les dades demanades a la pàgina inicial (figura 6.1).

Requeriments:

- L'usuari ha de ser major d'edat.
- El correu introduït no pot ser un e-mail utilitzat anteriorment per un altre usuari.

Amb aquestes dades l'usuari ja serà donat d'alta i es passarà a la següent fase, on es demanaran dades complementàries a la creació del perfil.

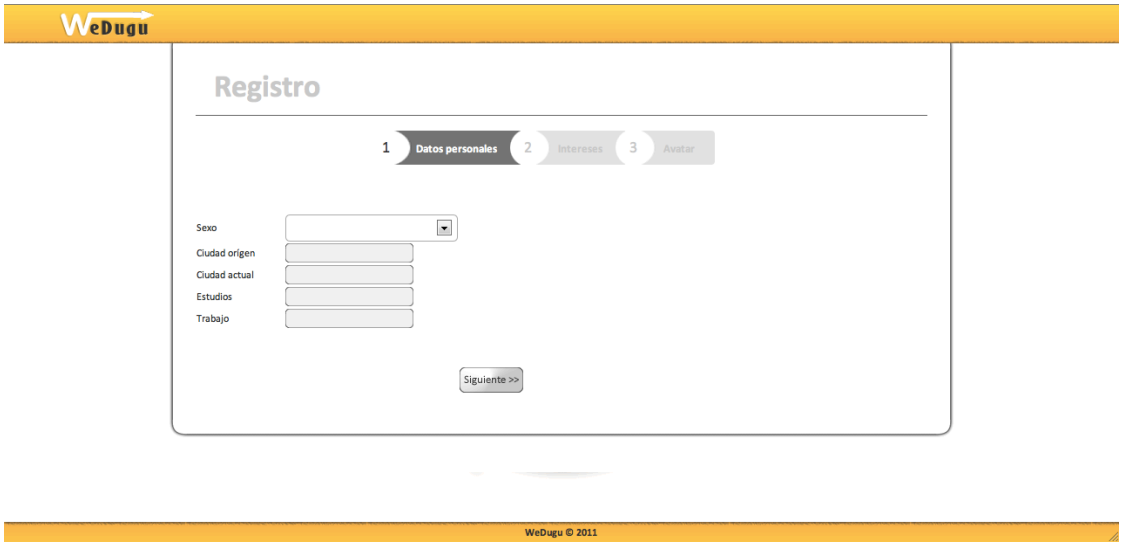

Figura 6.3 Fase 1 del registre

Fase 2:

Les dades següents no són obligatòries. L'usuari només les informarà si desitja que figurin al seu perfil.

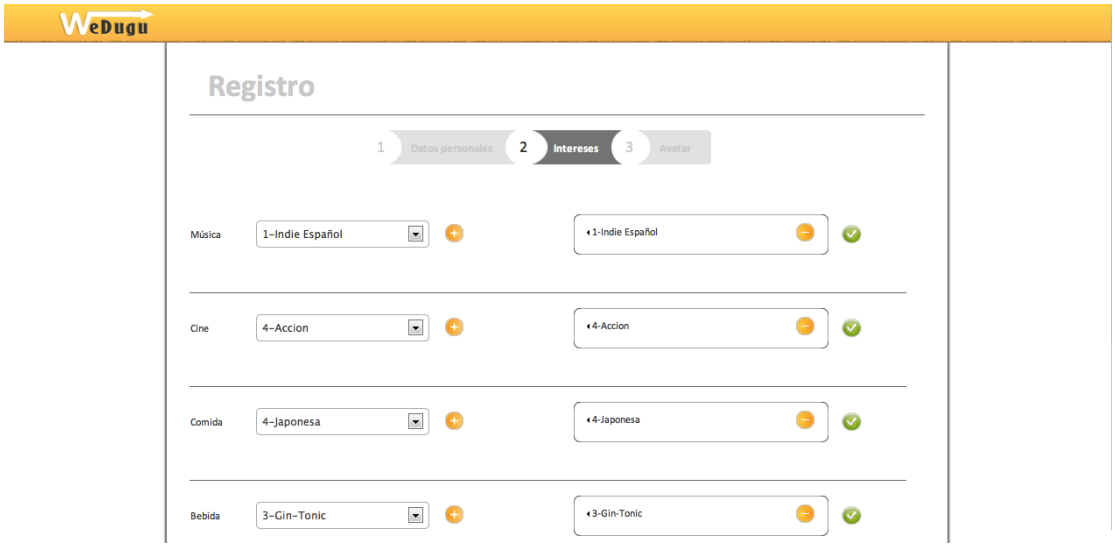

Figura 6.4 Fase 2 del registre

Aquesta fase és on l'usuari indica els seus gustos que, posteriorment, farà servir el servei de propostes. Les seves preferències es concentren en 5 blocs:

- Música
- Cinema
- Menjar

- Beguda
- Activitats

Dintre de cadascuna de les llistes, l'usuari trobarà un conjunt d'estils. Si s'identifica amb algun d'ells haurà de seleccionar-lo i afegir-ho amb el botó afegir. Si vol editar el conjunt d'estils que s'està creant, podrà esborrar aquell que desitgi amb el botó esborrar. Finalment, quan l'usuari hagi finalitzat un grup d'estils passarà al següent prement el botó "Ya está":

Un cop hagi editat els cinc grups apareixerà el botó "Siguiente" per que pugui arribar a la tercera i última fase.

Figura 6.5 Botons de navegació

Fase 3:

A l'última fase del registre es dóna la possibilitat a l'usuari de pujar al servidor una imatge personal que figuri com a imatge de perfil. Si així ho desitja, prement al botó "Selecciona la foto" s'obrirà l'explorador d'arxius del seu disc dur on podrà escollir la fotografia desitjada i l'aplicació s'encarregarà de pujar la imatge.

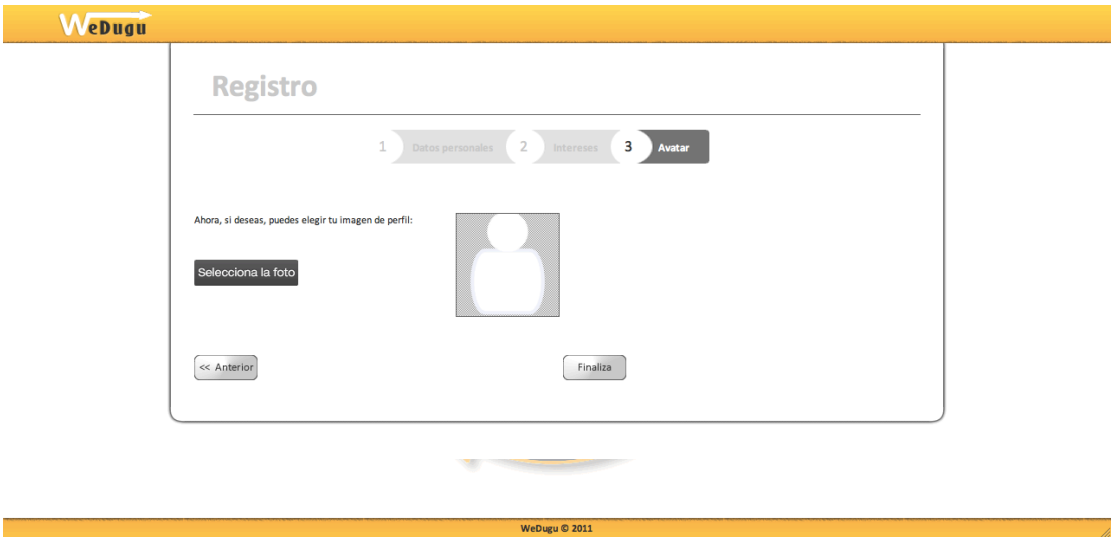

Figura 6.6 Fase 3 del registre

En acabar, si l'usuari prem al botó "Finaliza" es redirigirà a la pàgina principal amb sessió iniciada:

Figura 6.7 Pantalla finalització registre

Com es tracta d'un usuari nou, les consultes disponibles no obtindran cap resultat ja que serà necessari que l'usuari interactuï amb la pàgina.

6.1.3. Serveis disponibles.

Gestió d'amistats: El servei d'amistats és útil perquè permet a un usuari veure quins llocs visiten els seus amics, interessar-se per ells i visitar-els.

Com afegir una amistat:

A la pàgina principal, es troba la caixa que conté la llista d'amistats de l'usuari.

Figura 6.8 Caixa d'amistats

Si es desitja agregar-ne una caldrà prémer sobre el text "Agregar amigos" i apareixerà la finestra:

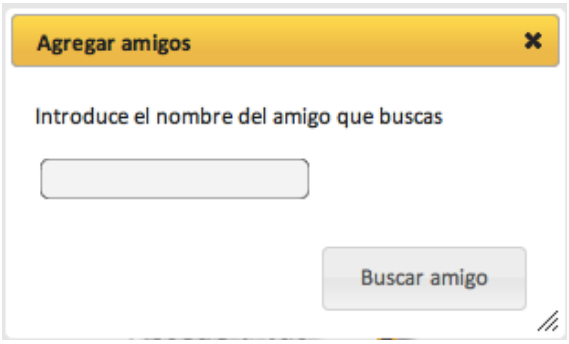

Figura 6.9 Finestra agregar amistat

L'usuari haurà de teclejar el nom de la persona que està cercant i prémer al botó "Buscar amigo". Seguidament, podrà visualitzar una llista amb les persones que s'aproximen a la seva cerca:

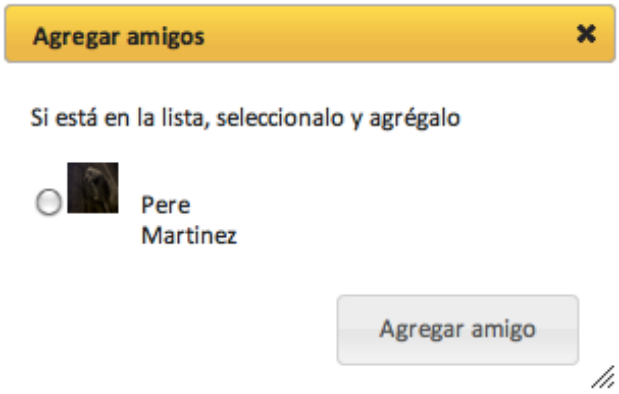

Figura 6.10 Finestra usuaris semblants

Abans d'afegir com a amic a la persona desitjada, podrà accedir al seu perfil prement al seu nom. El perfil serà mostrat a una nova finestra del navegador per que no es perdi la cerca que s'està utilitzant. Des d'aquesta mateixa pàgina es dóna la possibilitat d'afegir a la persona com amistat.

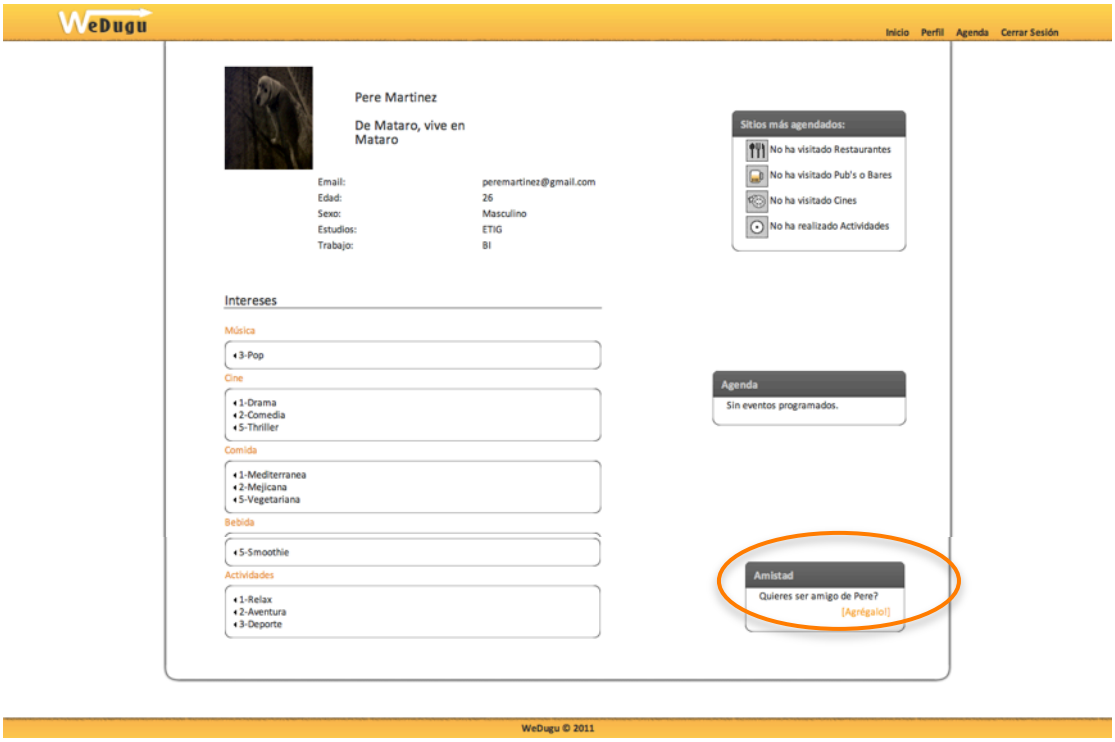

Figura 6.11 Pàgina perfil usuari

Si no s'afegeix a la persona des de la mateixa pàgina de perfil, l'usuari l'haurà de seleccionar de la llista obtinguda a la seva cerca i prémer al botó "Agregar amigo".
Finalment, aquesta persona apareixerà a la caixa d'amistats:

Figura 6.12 Amistat agregada

Visualitzar una amistat: A la caixa d'amistats només es visualitzaran cinc persones escollides aleatòriament cada vegada per no veure sempre les mateixes. Si l'usuari té més de cinc amistats agregades apareixerà el text "+Muestra todos" on l'usuari haurà de prémer per veure la llista sencera.

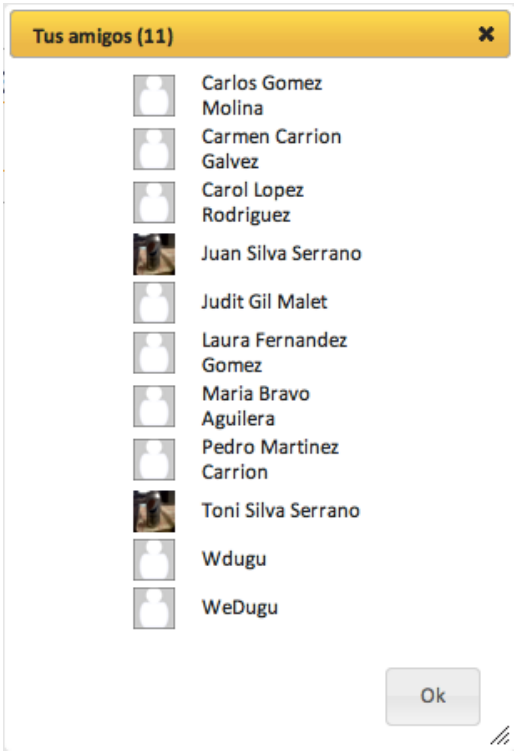

Figura 6.13 Llistat amistats total

Esborrar una amistat: Quan l'usuari vulgui que una persona deixi de formar part de la seva llista d'amistats només haurà d'accedir al seu perfil i a la part inferior trobarà:

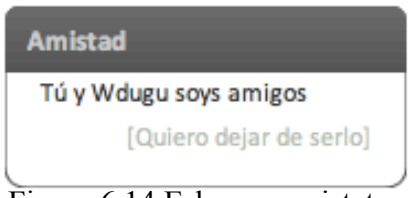

Figura 6.14 Esborrar amistat

Prement al text "Quiero dejar de serlo" aquesta persona serà esborrada de la seva llista d'amistats.

Llocs que més visiten els teus amics: Aquesta consulta li serveix a l'usuari per veure de tota la seva llista d'amistats quins són els llocs més freqüentats:

Figura 6.15 Llocs més freqüentats

Prement al nom del lloc es redirigirà a la pàgina del perfil del lloc.

Informació sobre llocs: La pàgina de perfil d'un lloc conté tota la informació referent al mateix: dades, estil, valoració i ubicació.

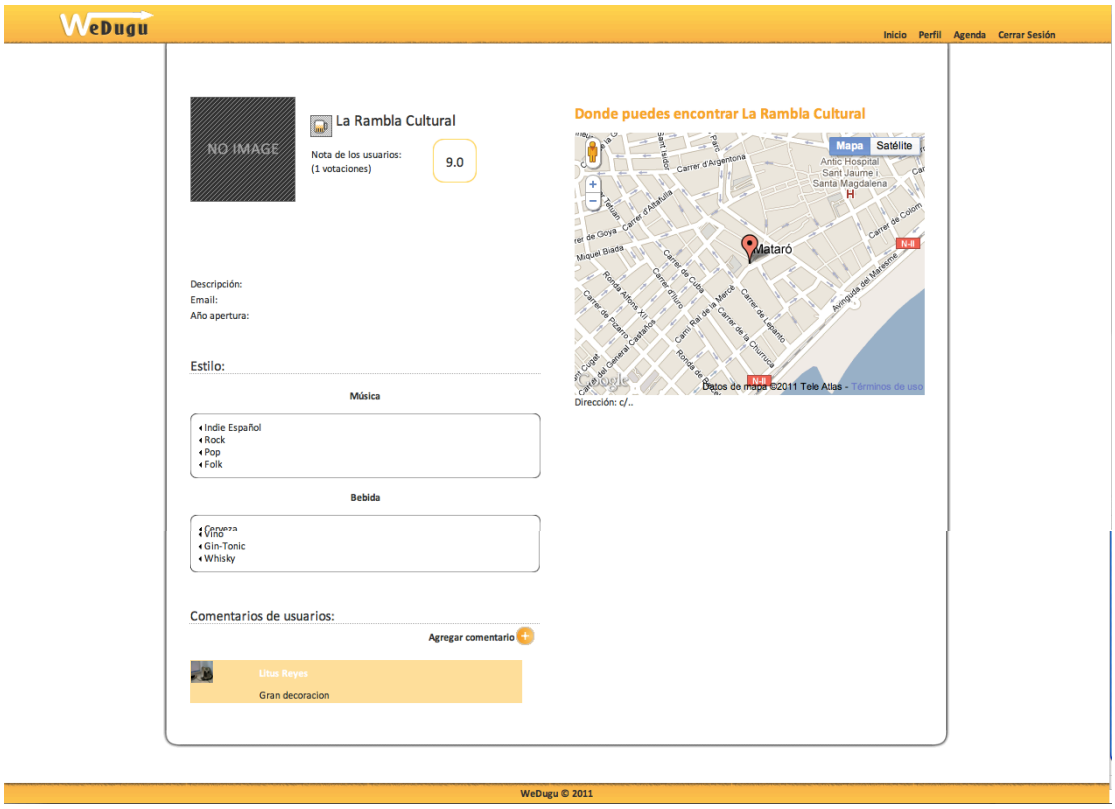

Figura 6.16 Pàgina d'un lloc

Valorar un lloc: Els llocs poden ser puntuats i comentats pels usuaris.

- Comentar: L'usuari que desitgi afegir un comentari sobre un lloc haurà de prémer sobre el botó "Agregar Comentario" situat a la part inferior de la pantalla

Seguidament, apareixerà una finestra amb una caixa de text on introduirà el seu comentari.

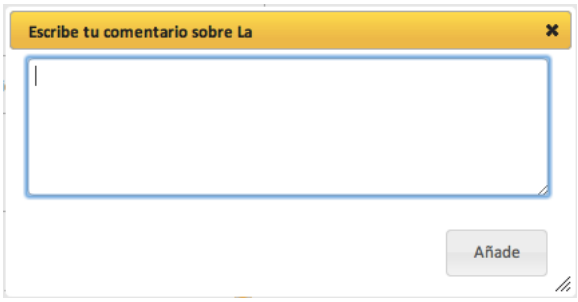

Figura 6.17 Comentar un lloc

Prement al botó "Añade" s'afegirà el comentari a la llista de comentaris.

- Puntuar: Per puntuar un lloc l'usuari haurà de prémer a sobre de "Nota de los usuarios" o bé sobre la nota mitja del lloc.

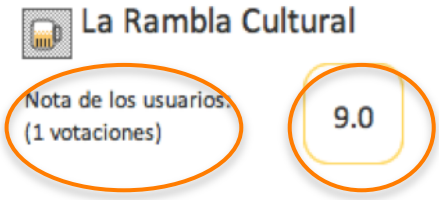

Figura 6.18 Com puntuar un lloc

Un cop hagi premut apareixerà un rang de notes.

Figura 6.19 Rang de notes per puntuar un lloc

Escollirà la nota desitjada i premerà al botó inferior per tal de tancar el desplegable.

La nota mitja del lloc s'actualitzarà amb l'última nota obtinguda.

Si l'usuari ja havia votat anteriorment aquest lloc, al anar a puntuar-lo li apareixerà la nota que anteriorment havia donat:

Figura 6.19 Nota que anteriorment s'havia donat a un lloc

Si el lloc en qüestió encara no havia sigut puntuat per ningú, apareixerà el text que es mostra a la figura adjunta:

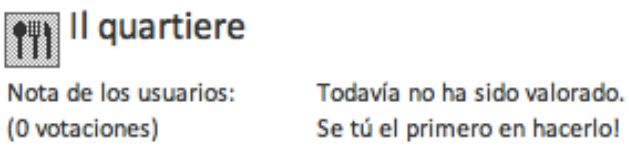

Figura 6.20 Lloc no puntuat

Gestió del perfil d'usuari: La diferència entre la pàgina de perfil del propi usuari i la dels altres usuaris a la web, és l'aparició dels botons d'edició de dades:

Figura 6.20 Botó edició

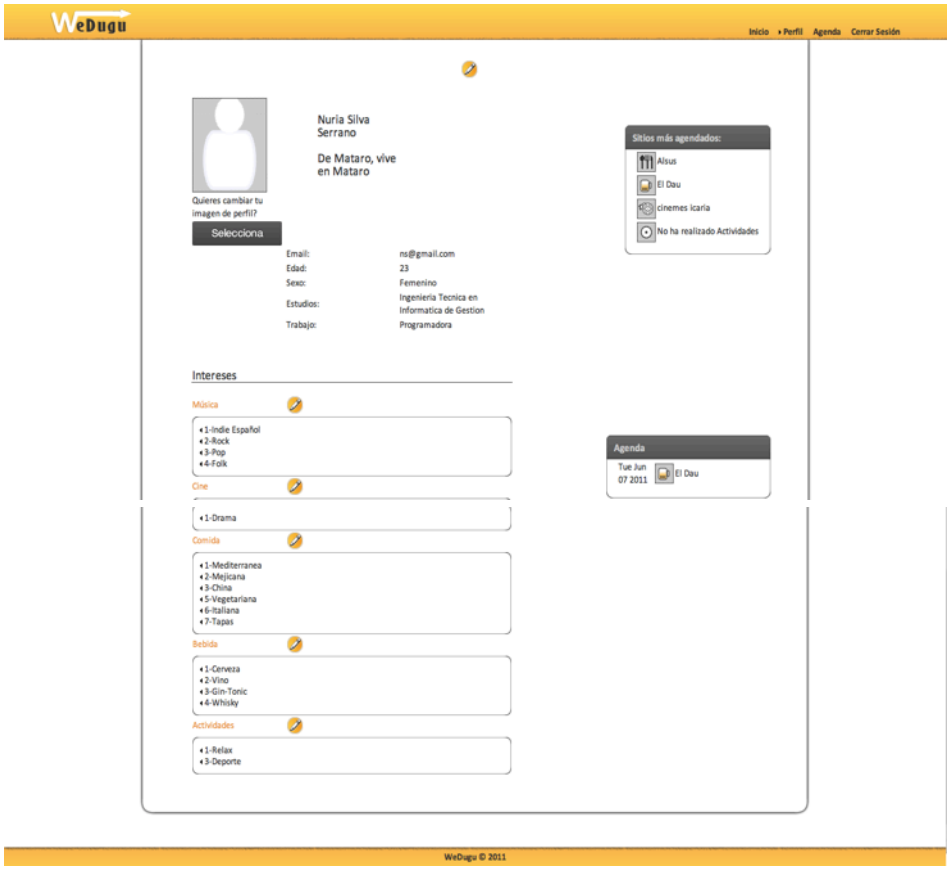

Figura 6.21 Perfil editable

El botó d'edició superior activa la modificació de les dades personals:

Figura 6.22 Dades activades per l'edició

Un cop l'usuari hagi modificat les dades desitjades, premerà al botó de guardar i la pantalla tornarà a la situació inicial.

Cada botó d'edició d'estils activarà la modificació de l'estil corresponent:

Figura 6.23 Edició d'estils

Aquesta pantalla és idèntica a la de gestió d'estils del registre d'usuari. L'usuari podrà afegir i eliminar aquells que cregui pertinents i seguidament premerà al botó de guardar per emmagatzemar el canvi.

A la pantalla del perfil d'usuari també hi apareixen dues informacions d'interès:

1. Llocs més visitats:

Figura 6.24 Llocs més visitats

Per cada tipus d'esdeveniment, mostrarà aquell lloc al que hi acudeix més sovint.

2. Agenda. Des d'aquesta caixa es pot dirigir a l'agenda personal on es visualitzen tots els esdeveniments realitzats i per realitzar i es mostraran només que estan planificats per fer pròximament.

Figura 6.25 Esdeveniments planificats

La mateixa caixa es troba a la pàgina principal de l'usuari.

Visualització dels últims esdeveniments: Aquest llistat fa referència als darrers fets que han succeït a la web. Activitats fetes o planificades per amics o per l'usuari propi.

Figura 6.26 Últims successos

Introducció d'esdeveniments ja realitzats: L'usuari introduirà a la següent caixa de text el nom del lloc on ha estat:

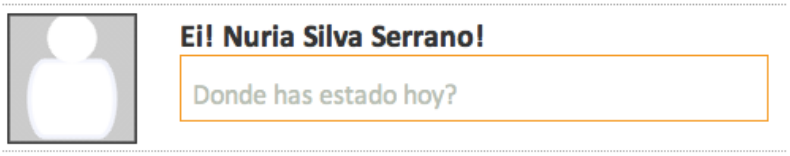

Figura 6.27 Com introduir el lloc on s'ha estat avui

Un cop premi la tecla ENTER es mostrarà una llista de llocs que s'aproximen al que l'usuari ha indicat:

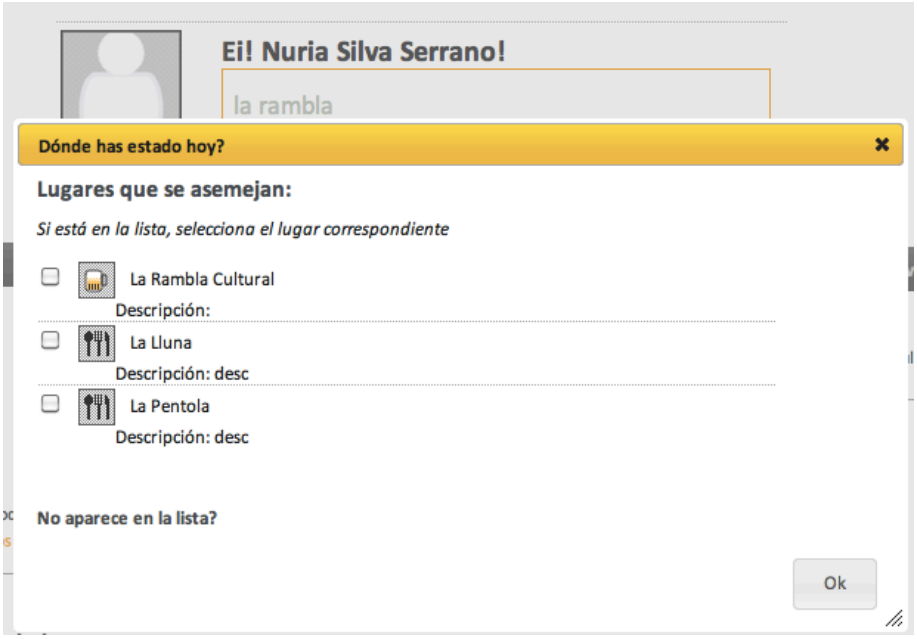

Figura 6.28 Llistat de llocs que s'assemblen per nom

Si vol veure la pàgina de perfil dels llocs que apareixen a la llista només haurà de prémer sobre el nom del mateix per que s'obri. Si el lloc on ha estat coincideix amb algun de la llista, l'haurà de seleccionar i prémer al botó "Ok".

L'esdeveniment es donarà d'alta a l'agenda de l'usuari amb data d'avui i apareixerà una finestra preguntant a l'usuari cap on vol dirigir-se:

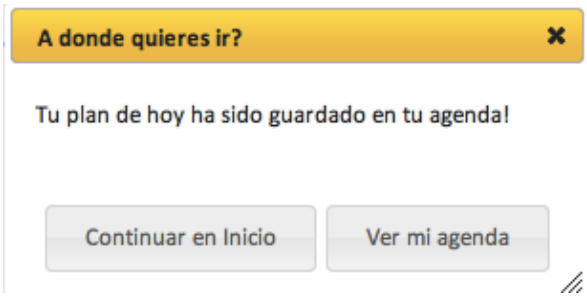

Figura 6.29 Pregunta a on es vol dirigir l'usuari

Si el lloc que l'usuari vol registrar no es trobés a la llista de possibilitats mostrades, haurà de prémer sobre el text "No aparece en la lista?" per donar-lo d'alta i apareixerà la finestra de registre del lloc:

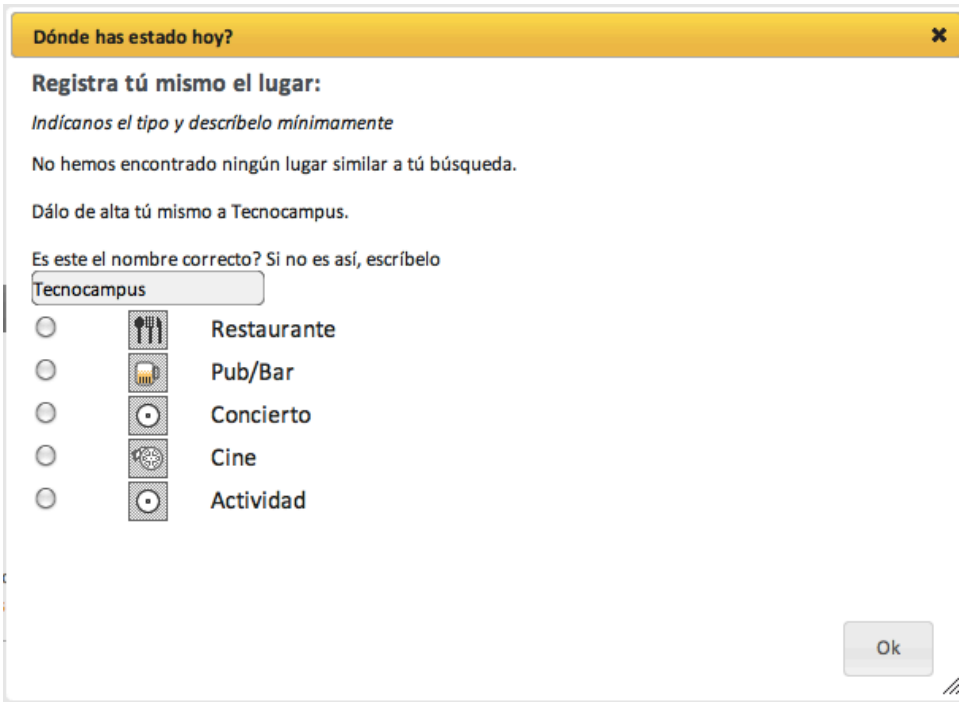

Figura 6.30 Lloc introduït no existeix a la Base de Dades

L'usuari haurà d'indicar el tipus de lloc que es tracta seleccionant-lo i apareixeran les llistes d'estils disponibles per el tipus en qüestió:

Figura 6.31 Dades per donar d'alta un lloc

El mètode d'afegir estils és idèntic al descrit en casos anteriors. Un cop indicades les dades del lloc s'ha de prémer al botó "Ok" per passar a la següent pantalla que es tracta d'ubicar el lloc.

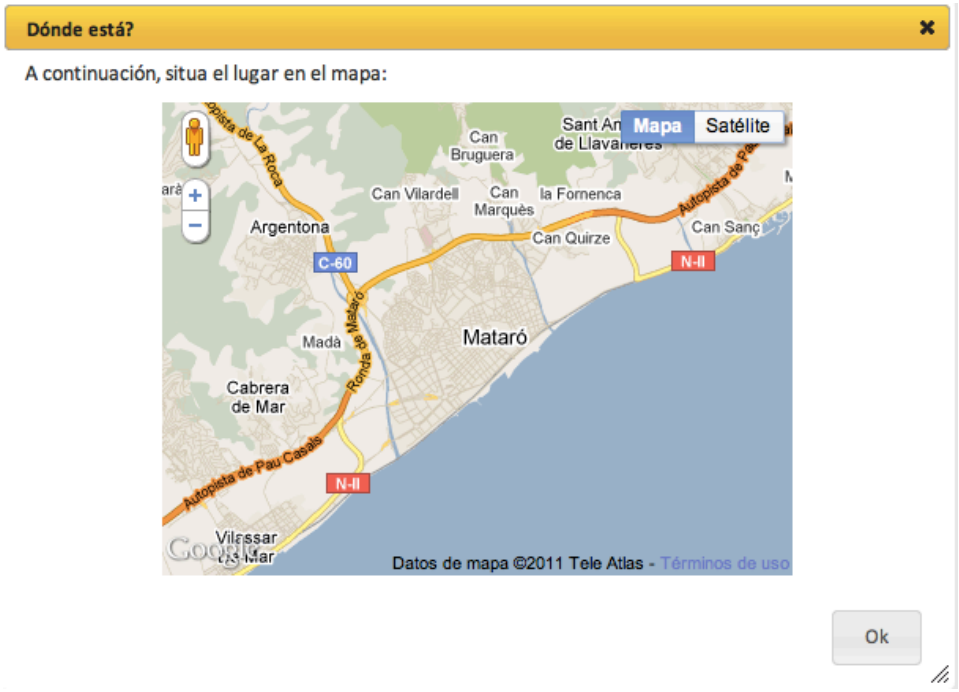

Figura 6.32 Pantalla per ubicar un lloc

L'usuari ha de prémer al mapa el punt on es troba el lloc. Pot desplaçar el mapa i també pot fer-hi zoom sobre ell.

Un cop ha marcat el lloc ha de prémer el botó "Ok" i es donarà d'alta el lloc i l'esdeveniment.

Finalment, apareixerà la mateixa finestra explicada anteriorment preguntant cap on es vol dirigir l'usuari.

Proposta d'esdeveniments: Per accedir a la proposta d'esdeveniments l'usuari ha de prémer a la següent imatge:

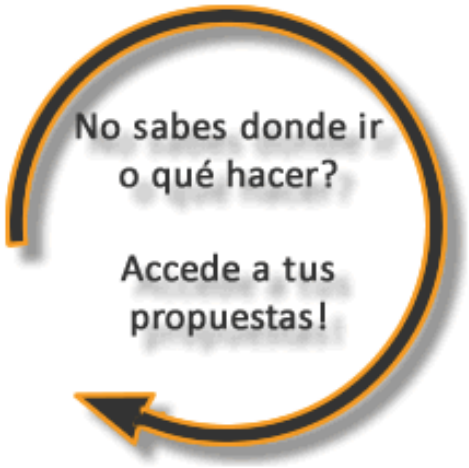

Figura 6.33 Accés a la proposta d'esdeveniments

Per cada tipus d'activitat (restaurant, pub o bar, concert, activitat o cinema) es donarà una proposta que s'adeqüi als gustos de l'usuari i a la seva ubicació actual, en un radi de tres km. Si d'algun dels tipus no es troben disponibles simplement no apareixeran a la llista:

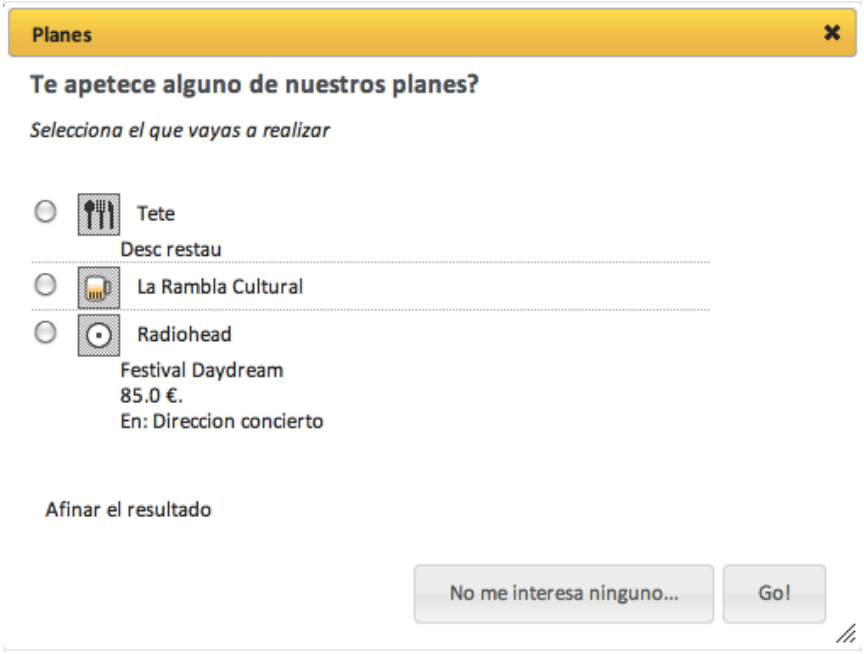

Figura 6.34 Proposta de plans

Prement a sobre del lloc es podrà navegar a la pàgina de perfil del mateix.

Si només es volen propostes d'un tipus d'esdeveniment l'usuari premerà a "Afinar el resultado" i apareixerà la selecció del tipus que desitja.

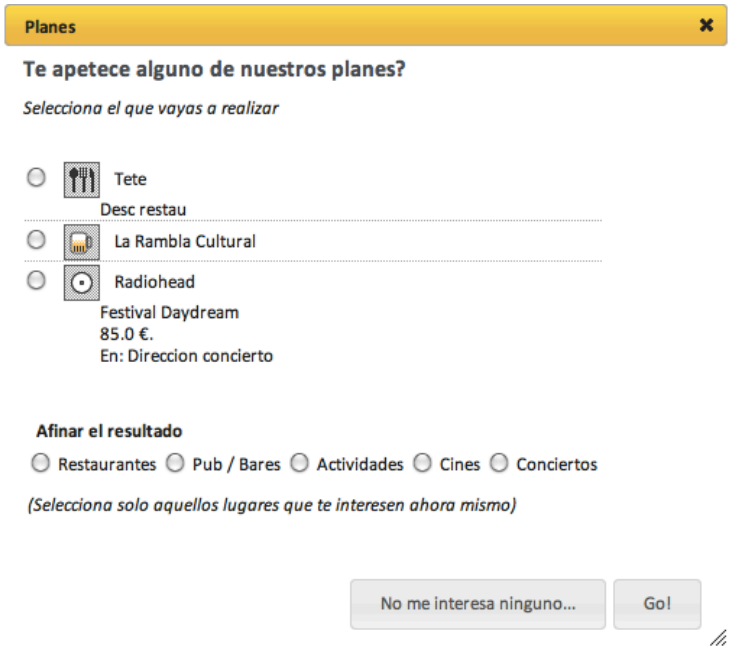

Figura 6.35 Afinar el resultat de la proposta

Al seleccionar un tipus, la llista canviarà per mostrar només cinc propostes d'aquest tipus:

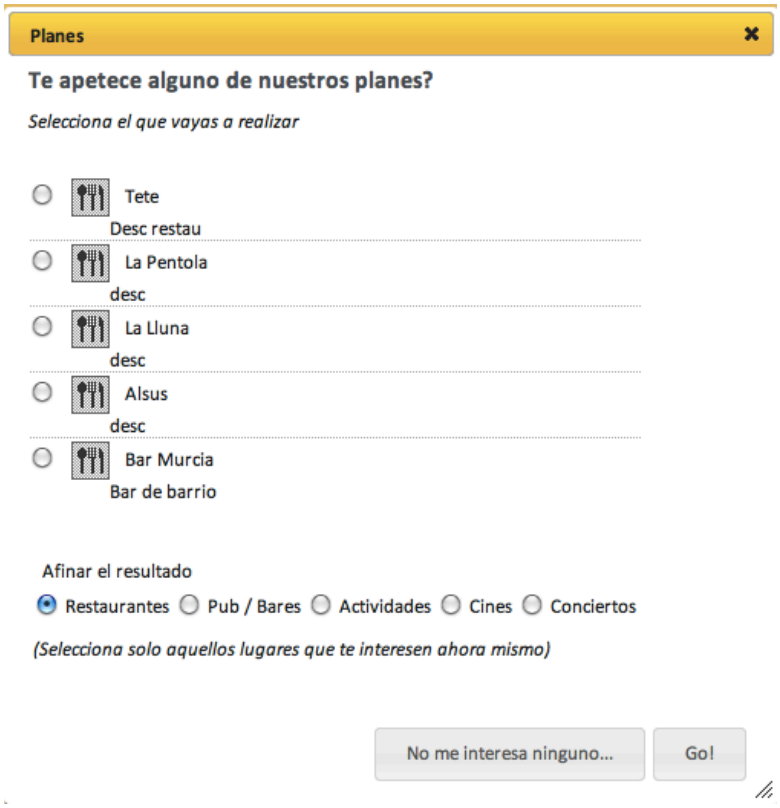

Figura 6.36 Proposta parametritzada per tipus de lloc

Si a l'usuari no l'interessa cap proposta premerà al botó "No me interesa ninguno..." i es tancarà la finestra, en cas contrari, si accepta una proposta, haurà de seleccionar-la i prémer al botó "Go!". Seguidament, apareixerà la finestra que li preguntarà cap on es voldrà dirigir, si continuar a la pàgina d'inici o bé, visualitzar l'agenda.

Agenda: L'usuari es podrà dirigir a la pàgina de l'agenda per qualsevol de les opcions abans esmentades o bé per la capçalera de totes les pàgines on figura el text Agenda.

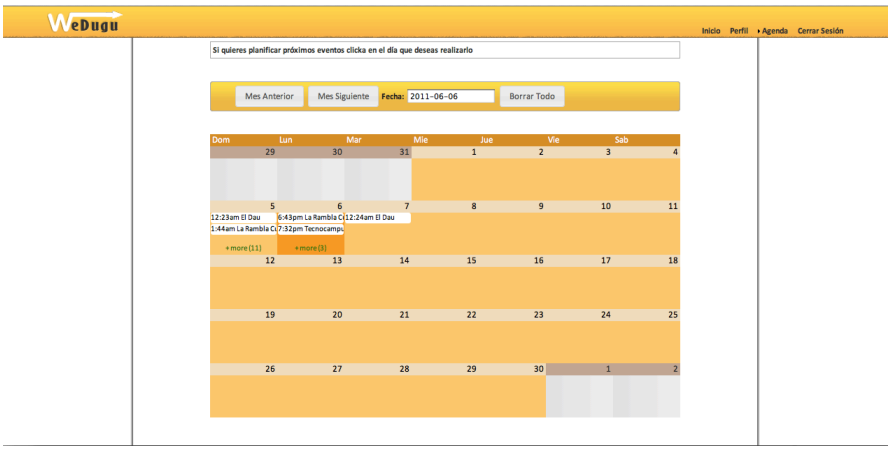

Figura 6.37 Pantalla agenda

Mitjançant la barra superior es dona l'opció de navegar dintre de l'agenda i anar al dia desitjat. Dintre d'aquesta barra, prement a la data apareix un calendari petit per anar més directe al dia.

Figura 6.38 Accedir a un dia del calendari en concret

Si l'usuari activa un dia del calendari s'obrirà la finestra per planificar futurs esdeveniments.

Planificació de futurs esdeveniments:

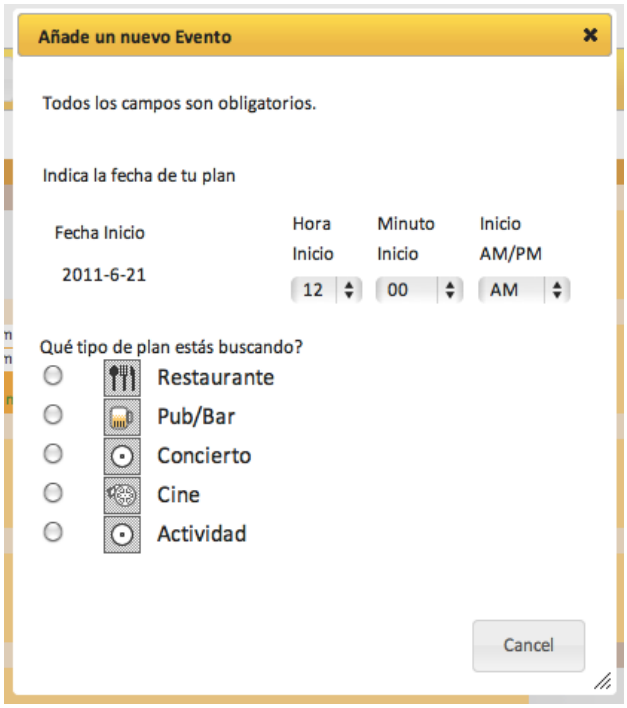

Figura 6.39 Finestra planificació futurs esdeveniments

És necessari indicar totes les dades: la data que es realitzarà l'esdeveniment i el tipus d'esdeveniment que s'està cercant.

Al prémer al tipus d'esdeveniment, apareixerà una llista de possibles llocs que podrien agradar a l'usuari (sense tenir en compte la ubicació).

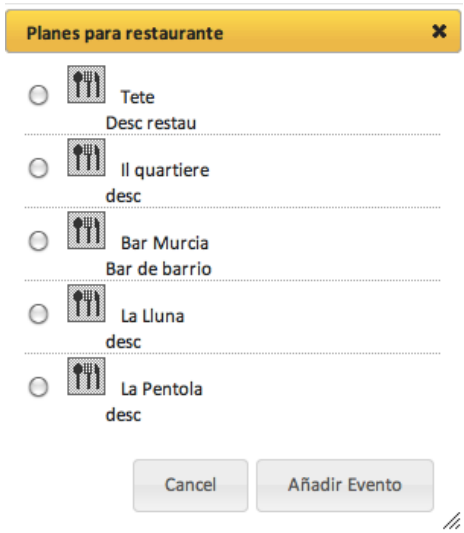

Figura 6.40 Proposta d'esdeveniments a planificar

Si algun dels llocs proposats l'interessa, l'haurà de seleccionar i prémer al botó "Añadir Evento". L'esdeveniment s'afegirà a l'agenda i apareixerà la finestra que preguntarà a l'usuari cap on es vol dirigir.

Si no interessa cap esdeveniment, es premerà al botó "Cancel" i es sortirà de la finestra.

7. Proves.

Per comprovar el correcte funcionament de l'aplicació s'ha seguit la següent matriu de proves.

7.1. Login.

Acció	**Resultat esperat**	**Prova**	**Resultat obtingut**
Introduir un usuari no existent a la base de dades	Mostrar missatge advertint que l'usuari no és vàlid.	Usuari: pppp@gmail.com Contrasenya: aaa	Ok
Introduir un usuari existent a la base de dades amb la contrasenya errònia	Mostrar missatge advertint que la contrasenya indicada no és correcte.	Usuari:ns@gmail.com Contrasenya: aaa	Ok
Introduir un usuari correcte i una contrasenya correcte	S'inicia la sessió i es redirigeix a la pàgina principal de l'aplicació.	Usuari:ns@gmail.com Contrasenya: nspasswd	Ok

Taula 7.1 Proves Login

7.2. Registre.

Acció	**Resultat esperat**	**Prova**	**Resultat obtingut**
Registrar l'usuari amb algun dels camps buits	Mostrar missatge advertint que totes les dades són	Camp nom buit i prémer el botó registre	Ok

	obligatòries		
Introduir una data de naixement, el càlcul de l'edat de la qual no fa 18 anys	Mostrar missatge advertint que l'edat mínima per utilitzar l'aplicació són 18 anys.	Data de naixement: 6 de Maig de 1994	Ok
Introduir una direcció d'e-mail que ja figura a la nostre Base de Dades per un altre usuari.	Mostrar missatge advertint que s'ha d'utilitzar un altre e-mail per que aquest ja existeix.	E-mail:ns@gmail.com	Ok
Introduir una direcció d'e-mail mal formulada	Mostrar missatge advertint que la direcció no és correcta.	ns	Ok
Introduir una repetició de contrasenya que no coincideix amb la contrasenya indicada primerament.	Mostrar missatge advertint que les contrasenyes no coincideixen.	Contrasenya: password Repetició contrasenya : contrasenya	Ok
Si no es produeix cap dels casos anteriors	Passar a la fase 1 del registre	Prémer el botó "Registro"	Ok

Taula 7.2 Proves Registre

7.2. Pantalla principal.

Acció	**Resultat esperat**	**Prova**	**Resultat obtingut**
Visualització de pantalla d'un usuari que no ha interactuat amb l'aplicació	Visualització de missatges que indiquin que per veure resultats s'ha d'interactuar amb la pàgina	Iniciar sessió amb un usuari que s'acaba de registrar	Ok
Agregar amistat: no introduir text al nom de l'usuari a cercar	Mostrar missatge advertint que s'ha d'introduir el nom de la persona que es busca.	Prémer el botó "Buscar amigo" sense afegir text.	Ok
Agregar amistat: No seleccionar l'amic que es vol agregar	Mostrar missatge advertint que s'ha de seleccionar la persona que s'agrega.	Prémer el botó "Agregar amigo" sense seleccionar la persona desitjada	Ok
Agregar amistat: seleccionar una persona a afegir que ja tenim a la nostra llista d'amistats.	Mostrar missatge advertint que aquesta persona ja figura com amistat a la nostre llista.	Seleccionar una persona que ja és a la nostre llista	Ok
Propostes: No seleccionar el pla que s'està acceptant	Mostrar missatge advertint que s'ha de seleccionar el pla desitjat	Prémer el botó "Go!" sense seleccionar el pla desitjat	Ok

Introducció lloc on s'ha estat avui: introduir el nom d'un lloc que no existeix a la Base de Dades	Formulari d'alta del lloc a la nostre Base de Dades	Lloc: Mezzaluna	Ok
Introducció fet avui lloc nou: donar d'alta el lloc sense marcar al mapa de Google Maps la seva ubicació	Mostrar missatge que indiqui que s'ha de marcar on es troba el lloc.	Prémer el botó "Ok" a la finestra del mapa sense marcar la ubicació	Ok
Introducció fet avui lloc existent: Acceptar un lloc dels semblant sense indicar-ho	Mostrar missatge que indiqui que s'ha de seleccionar el lloc on s'ha estat.	Prémer el botó "Ok" a la finestra "Dónde has estado hoy?" sense seleccionar el lloc	Ok
Introducció fet avui lloc existent: El lloc es tracta d'un cinema.	Mostrar formulari per introduir la pel·lícula que s'ha vist	Introduir "Cinemes Icaria" al camp de text "Dónde has estado hoy?" i seleccionar-l'ho	Ok

Taula 7.3 Proves pantalla principal

7.2. Perfil.

Acció	**Resultat esperat**	**Prova**	**Resultat obtingut**
Eliminació de tots	S'afegeix el text	Prémer el botó "-" per esborrar	Ok

els estils afegits a un grup d'estils	"Ninguno"	tots els estils d'un grup	
Agregar un estil quan a un grup no hi ha cap (text "Ninguno")	S'esborra el text "Ninguno" per afegir l'estil seleccionat.	Afegir un estil amb el botó "+"	Ok
Canviar la imatge del perfil per una que supera la mida permesa	Mostrar missatge advertint que la imatge és massa gran	Escollir una imatge del nostre sistema més gran de 3 megabytes.	Ok
Entrar a la pàgina de perfil d'un usuari que no és amic nostre	Veure el text "Agregar" que ens dona la possibilitat d'agregar-ho	La mateixa acció	Ok
Entrar a la pàgina de perfil d'un usuari que sí és amic nostre	Veure el text "Quiero dejar de serlo" per eliminar aquesta persona de la llista d'amistats.	La mateixa acció	

Taula 7.4 Proves perfil

7.3. Agenda.

Acció	**Resultat esperat**	**Prova**	**Resultat obtingut**
Prémer sobre un dia del calendari	Aparició de la finestra per planificar	Prémer sobre un dia qualsevol del calendari	Ok

	esdeveniments		
Planificació d' esdeveniments: Seleccionar tipus esdeveniment	Aparició de finestra amb llistat de propostes per planificar	La mateixa acció	Ok
Planificació d' esdeveniments: Afegir esdeveniment sense seleccionar	Mostrar missatge advertint que s'ha de seleccionar l' esdeveniment del llistat	Prémer el botó "Añadir Evento" sense seleccionar l' esdeveniment desitjat	Ok

Taula 7.5 Proves agenda

7.3. Sessió.

Acció	**Resultat esperat**	**Prova**	**Resultat obtingut**
Tancar el navegador sense tancar la sessió i entrar a l'aplicació	Redirigir-se a la pàgina d'inici amb la sessió iniciada.	La mateixa acció	Ok
Tancar el navegador després de tancar la sessió d'usuari	Redirigir-se a la pàgina d'inici de sessió o registre	La mateixa acció	Ok

Taula 7.6 Proves sessió

8. Valoració econòmica.

A continuació es detallarà la despesa econòmica que comportaria aplicar aquest projecte a la realitat.

El preu estipulat de dedicació per persona és de 20€ l'hora a la fase d'anàlisi del projecte i 10€ l'hora a la fase de desenvolupament. S'ha de tenir en compte que en aquest projecte han participat dues persones, per tant, alguns casos es multiplicaran per dos.

8.1 Anàlisi del projecte.

Tasca	Duració
Planificació	10h / persona
Anàlisis(Casos d'ús)	40h / persona
Estudi Tecnologies	20h / persona
Total anàlisi del projecte per persona	1400,00 €/persona
Total anàlisi del projecte	**2.800,00 €**

Taula 8.1 Valoració anàlisi

8.2 Eines utilitzades.

Eina	Cost
Netbeans 6.8 (2 unitats)	Gratuït
Sistema Operatiu MAC OS X (2 unitats)	169,00 € x 2
Eclipse SDK 3.5 (1 unitat)	Gratuït
Adobe Fireworks CS5 (1 unitat)	149,00 €
Enterprise Architect 9 (Desktop Edition Standard License) (1 unitat)	93,42 € (135 $)
Windows 2003 Server (1 unitat)	150,00 €
MySQL	Gratuït
Tomcat	Gratuït
Apache 2	Gratuït
Microsoft Office para Mac (2 unitats)	139,00 € x 2
Total eines utilitzades	**1.008,42 €**

Taula 8.2 Eines utilitzades

- Netbeans i Eclipse han estat utilitzats per tota la fase de codificació.

- El sistema operatiu MAC OS X, és l'instal·lat als ordinadors de treball.
- Adobe Fireworks s'ha fet servir per dissenyar les interfícies.
- Enterprise Architect s'ha utilitzat per realitzar els diagrames d'anàlisi de l'aplicació.
- Windows 2003 Server, MySQL, Tomcat i Apache 2 son les eines necessàries per el servidor de l'aplicació.
- Microsoft Office for Mac ha sigut necessari per escriure la documentació d'aquest projecte.

Es considera que aquestes eines seran utilitzades a pròxims projectes, per tant només s'ha de calcular la part proporcional a aquest. Considerant que les eines seran renovades cada any i que s'ha utilitzat durant 2 mesos el total serà de **168,07 €**.

8.3 Codificació.

Tasca	Duració
Disseny web	80h (1 persona)
Creació de capa domini amb JPA	20h (1 persona)
Formació Android	80h (1 persona)
Codificació casos d'ús generals Web	80h (1 persona)
Codificació casos d'ús general Android	80h (1 persona)
Codificació Casos d'ús socials Web	40h (1 persona)
Codificació Casos d'ús socials Android	40h (1 persona)
Codificació Casos d'ús agenda Web	100h (1 persona)
Codificació Casos d'ús agenda Android	100h (1 persona)
Revisió final Web	20h (1 persona)
Revisió final Android	20h (1 persona)
Total codificació del projecte	**6.600,00 €**

Taula 8.3 Cost de la codificació

A les hores de codificació ja hi són incloses les hores dedicades a les proves de l'aplicació.

8.4 Documentació.

Tasca	Duració
Redacció	60h / persona
Total documentació del projecte	**1.200,00 €**

Taula 8.4 Valoració documentació

El preu establert per temps de dedicació a la redacció de la documentació d'aquest projecte es considera el mateix que per la codificació, és a dir, 10€ l'hora.

8.5 Hardware utilitzat.

Equip	Cost
Macbook Pro	1.400,00 €
Macbook	840,00 €
Portàtil Ahtec Singal zw6	300,00 €
Total documentació del projecte	**2.540,00 €**

Taula 8.5 Cost hardware

Considerem que la vida útil dels ordinadors és de 3 anys, comptant que la durada d'aquest projecte és de 4 mesos, apliquem el factor d'amortització al total obtingut i obtenim el total de **282,22 €**.

8.6 Suma total cost del projecte.

Concepte	Cost
Anàlisi del projecte	2.800,00 €
Eines utilitzades	168,07 €
Codificació	6.600,00 €
Documentació	1.200,00 €
Hardware utilitzat	282,22 €
Costos indirectes	1.657,54
Total cost del projecte	**12.707,83 €**

Taula 8.6 Suma total cost projecte

Els costos indirectes són els corresponents a energia, comunicacions, consumibles, lloguer, amortització de mobiliari, etc. Es considera un 15% del total del projecte.

Tenint en compte que aquest projecte engloba 2 projectes (cadascun realitzat per una persona), aquest projecte no es pot considerar excessivament car per que s'ha d'apreciar el cost de les eines que han sigut necessàries i que l'esforç ha sigut multiplicat per dos.

9. Conclusions.

9.1. Errades sense solucionar.

- **Obtenir posició actual de l'usuari:** No hi ha hagut manera d'aconseguir que funcioni l'API d'HTML5 per obtenir la ubicació de l'usuari amb el navegador Safari. Hi ha vegades que amb el navegador Chrome no funciona correctament i amb el navegador Firefox funciona a la perfecció.

9.2. Possibles ampliacions a tenir en compte.

Degut al temps disponible per realitzar aquest projecte, aquesta aplicació s'ha fet com una maqueta del que realment s'havia pensat que seria. És a dir, hi ha moltes funcionalitats pensades que no s'han pogut implementar ja que no hi ha havia temps material. Les funcionalitats pensades són:

- **Aplicació de paginació a les visualitzacions de resultats.** Els retorns del servidor a la vista de l'aplicació estan implementats per retorns petits. En un entorn real, amb gran densitat de dades a la Base de Dades aquesta implementació no seria vistosa ja que s'obligaria a l'usuari a recórrer amb l'scroll la llista de dalt a baix. Seria ideal implementar una paginació de entre 5 i 10 resultats per fer més agradable la visualització dels resultats.
- **Acceptació de condicions d'ús i privacitat.** Redactar un text explicant les condicions d'ús i privacitat d'aquesta aplicació i aportar de més seguretat a la mateixa.
- **Enviament d'e-mail per confirmar el registre.** Un cop l'usuari introdueix les dades bàsiques per el registre, seria correcte enviar a la direcció indicada un e-mail de confirmació per aportar més seguretat. Des d'aquest e-mail de confirmació es navegaria a la fase de creació del perfil i no directament, com s'està fent ara.
- **Control de seguretat a la contrasenya de l'usuari.** S'hauria de realitzar un script que valorés el nivell de seguretat de la contrasenya que introdueix l'usuari. Actualment, només s'està validant que realment sigui el que tenia pensat escriure, obligant-lo a introduir-ho dues vegades i comprovant que coincideixen les dos entrades.

- **Recordar l'usuari i la contrasenya de l'inici de sessió.** S'hauria d'afegir la possibilitat que l'usuari vulgui guardar les seves credencials al navegador. S'implementaria afegint dues cookies (informació que el navegador de l'usuari recorda) per que encara que es tanqui la sessió, el navegador consulti les cookies per mostrar les credencials a utilitzar.
- **Renovació de la contrasenya oblidada.** Si l'usuari no recorda la seva contrasenya s'hauria d'implementar una funcionalitat per que si així ho desitja, s'enviï per e-mail una nova contrasenya generada automàticament.
- **Propostes de llocs que podrien agradar.** S'implementaria una consulta per la pàgina principal de l'aplicació on és retornés una llista de llocs que encaixarien amb la persona segons el seu perfil i que es trobessin al voltant de les ubicacions per on sol estar l'usuari.
- **Propostes d'amistats semblants.** S'implementaria també una proposta de persones que podrien formar part de les nostres amistats si aquestes comparteixen gustos amb nosaltres i visiten llocs semblants als que nosaltres hem visitat.
- **Control de privacitat.** Donar la opció a l'usuari d'escollir qui pot veure i qui no el seu perfil i la seva agenda.
- **Control de la distància de propostes.** Deixar que l'usuari indiqui quina distància màxima vol que hi hagi entre la seva ubicació i les propostes que l'aplicació li farà.
- **Paràmetres desitjats a les propostes.** Deixar que l'usuari filtri propostes per horari i preu.
- **Implementar una aplicació per els administradors.** Seria necessari implementar una aplicació per gestionar estils i llocs.

10. Referències.

JQuery

[1] http://jquery.com/ Pàgina oficial del framework.

[2] http://jqueryui.com/ Pàgina oficial de l'extensió del framework.

[3] http://www.desarrolloweb.com/manuales/manual-jquery.html Manual de JQuery por desarrollo web.

[4] http://www.json.org/js.html Plugin JSON per JQuery.

[5] http://www.yoxigen.com/yoxview/ Plugin visor d'imatges per JQuery.

[6] http://www.uploadify.com/ Plugin de pujada de fitxer per JQuery.

[7] http://code.google.com/p/jquery-frontier-calendar/ Plugin calendari per JQuery.

Disseny web

[8] http://www.netdreams.co.uk/ Pàgina d'on es van agafar idees per el disseny.

[9] http://www.ajaxload.info/ Generació d'arxiu animat per la simul·lació d'espera amb AJAX.

Json

[10] http://json-lib.sourceforge.net/ Llibreria per enviar objectes Json des del servidor.

API Google

[11] http://www.desarrolloweb.com/manuales/desarrollo-con-api-de-google-maps.html Manual per utilitzar l'API de Google Maps.

[12] http://code.google.com/intl/es-ES/apis/maps/index.html Pàgina oficial.

API Geolocalització HTML5

[13] http://dev.w3.org/geo/api/spec-source.html Pàgina oficial.

[14] http://www.pedroventura.com/utilidades/cual-es-mi-ip/ Exemple de geolocalització amb HTML5.