Idees.Net

Gestió d'idees innovadores.

Roger Saladrigas Sitjà
Enginyeria tècnica en informàtica de gestió

Consultor: Jordi Ceballos Villach

8 de Juny de 2008

Idees.Net

El present document és el resultat final del estudi i desenvolupament d'un programari per a la gestió d'idees innovadores en el marc d'una empresa o col·lectiu. La gestió d'idees permet recollir, avaluar i organitzar les idees amb el objectiu de millorar el flux d'idees entre la direcció de l'empresa i la resta d'empleats. **Idees.Net** ofereix una solució senzilla que permet a qualsevol membre del col·lectiu aportar noves idees i participar activament en el procés de selecció d'idees.
Per al desenvolupament de l'aplicació s'ha utilitzat la tecnologia .NET de Microsoft intentant aprofitar al màxim les possibilitats que ofereix la aquesta plataforma en conjunció amb altres tecnologies de recent aparició.

Índex de continguts

Índex de figures

1 Introducció

1.1 Justificació i context.

Les idees, per normal general, sorgeixen en el moment menys oportú i si no s'anoten en el moment, és molt possible que s'oblidin. Per altra banda, no totes les idees que es tenen son prou bones, ni estan prou desenvolupades com per a poder-les dur a terme. Poder compartir les nostres idees amb un col·lectiu permetrà que les bones idees es desenvolupin correctament i arribin, en el seu moment, a materialitzar-se.

Per a evitar que les bones idees quedin en no res cal un sistema informàtic que permeti gestionar tota aquesta informació. És per això que en aquest treball de final de carrera proposa una solució senzilla i versàtil per a resoldre la problemàtica exposada.

1.2 Objectius.

La realització d'aquest projecte tenia uns objectius inicials molt clars des de el punt de vista de la solució del problema:

- Permetre als usuaris afegir idees al sistema de la forma més senzilla possible i des de qualsevol lloc/medi: correu electrònic, navegador web, dispositiu mòbil.
- Implementar un sistema de revisió que permeti obrir un canal de comunicació entre la persona que crea la idea i la persona que la revisa.
- Oferir als usuaris la possibilitat de votar per aquelles idees que els semblin millors.
- Generació d'informes recollint les idees més votades.

Però també existeixen uns objectius acadèmics:

- Consolidar els coneixements adquirits al llarg dels estudis en Enginyeria tècnica en informàtica de gestió.
- Familiaritzar-se amb el Visual Studio 2008.
- Aprendre i posar en pràctica algunes de les novetats que incorpora el llenguatge

C# en la seva versió 3, com per exemple el llenguatge de consultes integrat LINQ.

- Enriquir l'experiència del usuari mitjançant el ús de noves tecnologies com AJAX o Microsoft Silverlight.

1.3 Enfocament i mètode seguit.

Per a la realització del projecte s'ha seguit el cicle de vida clàssic o en cascada de manera que cada producte generat correspon amb el final d'una etapa. Cal tenir en compte, però, que la etapa de manteniment queda fora del abast del projecte.

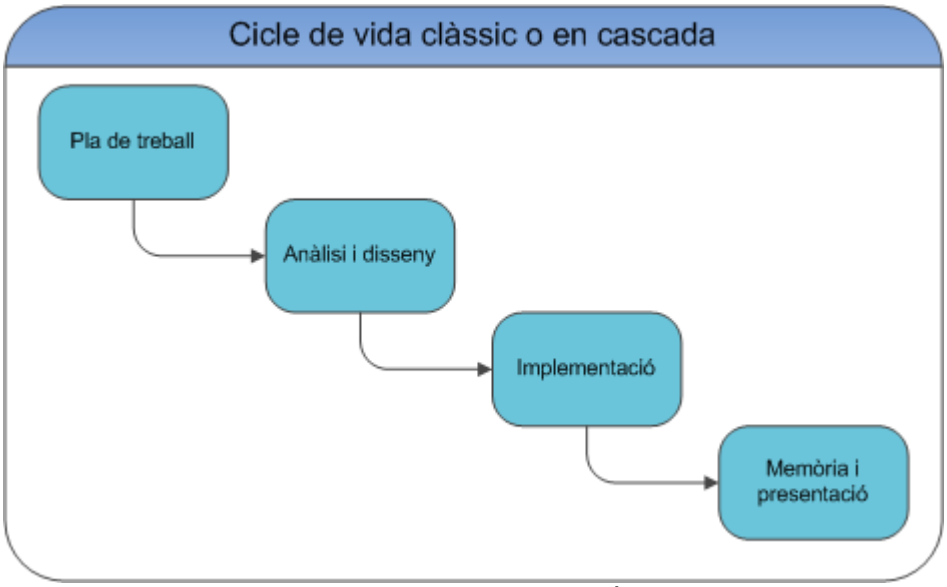


Figura 1: Cicle de vida clàssic.

Al finalitzar cadascuna d'aquestes etapes s'obté un producte o material que serveix com a punt de partida per a la següent etapa. Tot i que aquest mètode no permet tornar enrere un cop finalitzada una etapa ha estat necessari fer-ho en alguna ocasió per tal de resoldre males decisions presses anteriorment.

1.4 Planificació del projecte

La primera etapa del cicle de vida fa referència al pla de treball del projecte. Així doncs, el primer pas ha estat elaborar un pla de treball del projecte que faciliti l'entrega puntual del producte final.

Tasca	Inici	Finalització
Pla de treball		
Pla de treball	25/02/2008	10/03/2008
Anàlisi i disseny		
Anàlisi	11/03/2008	19/03/2008
Prototip	19/03/2008	27/03/2008
Disseny	27/03/2008	06/04/2008
Implementació		
Implementació	07/04/2008	06/05/2008
Proves	06/05/2008	11/05/2008
Manual d'usuari i manual d'instal·lació	17/08/2008	18/05/2008
Memòria i presentació		
Memòria	19/05/2008	02/06/2008
Vídeo presentació	02/06/2008	08/06/2008

Figura 2: Planificació del projecte.

El projecte s'ha realitzat segons la planificació i s'ha pogut complir, gairebé al cent per cent, amb la planificació establerta al principi del mateix.

1.5 Productes obtinguts

Tal com s'ha mencionat en el apartat de enfocament i mètode seguit al finalitzar cadascuna de les etapes de cicle de vida en cascada s'ha obtingut un producte o material entregable. Aquesta taula resumeix els productes obtinguts al llarg del projecte:

Producte	Descripció
Pla de treball	Document inicial on es planifica el projecte i es fa una primera aproximació als requeriments funcionals del projecte.
Anàlisi i disseny	
Anàlisi	Especificació de les funcionalitats i especificació dels casos d'ús.
Disseny	El document de disseny inclou el disseny de la base de dades que donarà suport a l'aplicació així com el disseny de les classes.
Prototip	Primera aproximació a la interfície d'usuari. És una maqueta gràfica que permet fer-se una idea de com serà l'aplicació un cop estigui acabada.
Implementació	
Idees.Net	Aplicació web per a navegadors d'escriptori.
Idees.Net Mobile	Aplicació web per a dispositius mòbils.
Idees.Net Agent	Aplicació Win Forms. Es una aplicació auxiliar.
Documentació	Manual d'instal·lació i petit manual d'usuari.
Memòria	
Memòria	Document on es recull tota la informació sobre les tasques realitzades per tal de dur a terme el projecte.
Presentació	Vídeo demostratiu de l'aplicació. Permet veure el funcionament de l'aplicació.

Figura 3: Relació de productes obtinguts.

1.6 Breu descripció dels altres capítols de la memòria

El següent capítol recull els requisits, funcionals i no funcionals, identificats durant la part d'anàlisi del projecte. També es detallen els usuaris o rols que són reconeguts pel sistema.

El capítol tres està dedicat, íntegrament, al anàlisi. Això implica que hi podreu trobar diagrames de casos d'ús, una explicació detallada de cada cas d'ús i el model conceptual del domini.

Seguidament, s'ha inclòs un capítol de disseny on s'engloben totes aquelles qüestions que fan referència a la fase l'arquitectura de l'aplicació, persistència,diagrames de classes i decisions tecnològiques que s'han pres per poder desenvolupar el projecte.

S'ha inclòs un capítol que fa referència a la implementació del producte, on es detallen alguns dels aspectes més rellevants en quant a programació de la solució .

El capítol 6 inclou captures de pantalla de l'aplicació que destaquen les funcionalitats de l'aplicació.

Els últims capítols d'aquest document fan referència a les conclusions i les línies de desenvolupament futur.

2 Requisits

2.1 Usuaris a considerar

De la descripció inicial del problema és poden identificar fàcilment tres tipus d'usuari o rols. Per una banda existeixen els empleats – també anomenats ideòlegs – que són els encarregats d'aportar noves idees i de votar per aquelles idees que considerin més innovadores. En el nivell immediatament superior existeixen els caps de departament, que s'encarreguen de validar o refusar les idees proposades per els empleats. Finalment existeix el directiu que és qui sol·licita les idees i, per tant, és qui rep el informe final amb les idees més votades pels empleats. Així doncs cal considerar tres rols:

- **Ideòleg**: son els usuaris que poden afegir idees i votar.
- **Cap de departament**: la seva funció és la revisar les idees aportades per els ideòlegs.
- **Directiu**: són els encarregats de crear i gestionar els projectes i els receptors del informe final.

2.2 Requisits funcionals

2.2.1 Gestió d'usuaris

- Sistema d'identificació d'usuaris mitjançant nom d'usuari i contrasenya.
- Sistema d'identificació d'usuaris mitjançant un enllaç.
- Identificació del rol d'un usuari.

Per a la identificació dels usuaris i rols s'ha fet ús del sistema que proporciona el propi framework del .NET: les classes *MembershipProvider* i *RoleProvider* proporcionen els mètodes necessaris per a validar un usuari i obtenir el seu rol. Ha estat necessari implementar dues classes que deriven de les mencionades anteriorment per a cobrir les necessitats específiques del projecte.

En quant a la validació mitjançant un enllaç s'ha implementat un sistema que genera un codi o tiquet que es vàlid durant 24 hores i que està associat a un usuari en concret. En utilitzar el codi en un enllaç el sistema genera automàticament una *cookie* o galeta idèntica a la que es genera el *MembershipProvider*. Aquest sistema de tiquets permet als usuaris validar-se de la mateixa manera que ho farien en introduir el seu nom d'usuari i contrasenya.

2.2.2 Gestió de projectes

- Afegir projectes a la base de dades.
- Gestió dels estats d'un projecte: etapa inicial, etapa de votació i finalització del projecte.
- Generar un informe al finalitzar un projecte.
- Oferir gràfics estadístics sobre els projectes i usuaris.

2.2.3 Gestió d'idees

- Afegir idees a un projecte.
- Revisar les idees proposades. Cal especificar un motiu al refusar una idea.
- Votar les millors idees d'un determinat projecte.

2.3 Requisits no funcionals

2.3.1 Ús del correu electrònic.

El principal requisit no funcional de l'aplicació era la possibilitat d'interactuar amb l'aplicació des del correu electrònic. Aquest era, per tant, un requisit que no es podia deixar de banda. Concretament, calia oferir la possibilitat d'afegir idees mitjançant el correu electrònic.

2.3.2 Suport per a dispositius mòbils.

Calia oferir la possibilitat d'utilitzar dispositius mòbils per a realitzar les tasques de gestió d'idees. Els usuaris d'un dispositiu mòbil també poden utilitzar el client de correu del seu dispositiu per a afegir idees a un projecte. Així doncs, només calia oferir una solució que cobrís la resta de requisits funcionals de la gestió d'idees: revisió i votació d'idees. Es va crear una nova aplicació fent ús de la tecnologia ASP.NET Mobile que permet crear aplicacions web per a qualsevol tipus de dispositiu mòbil.

2.3.4 Directori actiu de Windows.

L'aplicació havia d'oferir alguna mena de suport que permetés obtenir informació dels usuaris des del directori actiu de Windows. Al mateix temps calia que l'aplicació pogués obtenir les dades dels usuaris des d'una taula de la base de dades. Utilitzant les classes que la plataforma .NET ofereix s'ha implementat el suport per al directori actiu.

3 Anàlisi

3.1 Gestió de projectes

CDU001 Afegir projecte	
Resum	Crear projecte i iniciar el procés de recollida d'idees.
Actors	**Directiu**
Casos relacionats	
Precondició	El usuari s'ha autentificat.
Postcondició	Es registra la informació sobre el nou projecte.
Flux normal	■ El usuari demana crear un nou projecte. ■ El sistema presenta un formulari per omplir amb les següents dades: *nom*, *motiu*, *etiquetes per defecte*. ■ El directiu introdueix les dades. ■ El sistema emmagatzema el projecte i executa el cas d'ús Enviar Notificació indicant quins empleats rebran la notificació.

Figura 4: CDU - Afegir projecte.

CDU002 Iniciar votació	
Resum	Inicia la fase de votacions d'un projecte.
Actors	Directiu
Casos relacionats	Llistar projectes
Precondició	El usuari s'ha autentificat.
Postcondició	El projecte canvia al estat *en votació*.
Flux normal	■ El sistema mostra una llista de projectes que estan en estat *iniciat*. ■ El directiu selecciona un projecte i demana que s'iniciï la fase de votacions. ■ El sistema canvia el estat del projecte.

Figura 5: CDU - Iniciar votació.

CDU003 Finalitzar votació	
Resum	Finalitza la fase de votacions d'un projecte.
Actors	**Directiu**
Casos relacionats	Enviar Notificació
Precondició	El usuari s'ha autentificat.
Postcondició	El projecte canvia al estat *finalitzat.*
Flux normal	■ El sistema mostra una llista de projectes que estan en estat *en votació*. ■ El directiu selecciona un projecte i demana que finalitzi la fase de votacions. ■ El sistema canvia l'estat del projecte. ■ El sistema genera un informe en format PDF amb les deu idees més votades i l'envia als directius mitjançant el cas d'us Enviar Notificació.

Figura 6: CDU - Finalitzar votació.

DU004 Monitoritzar projectes	
Resum	Envia notificacions automàtiques als usuaris.
Actors	**Agent**
Casos relacionats	Enviar Notificació
Precondició	Cap
Postcondició	Cap
Flux normal	■ El sistema comprova si ja han votat tots els empleats assignats a un projecte concret. ■ S'envia una notificació a aquells usuaris que no han emès encara el seu vot. ■ El sistema comprova si hi ha noves idees pendents de revisió. ■ S'envia una notificació als caps de departament informant-los que hi ha idees pendents de revisar.
Observacions	Les notificacions s'envien fent una crida al cas d'ús Enviar Notificació.

Figura 7: CDU - Monitoritzar projectes.

CDU005 Enviar notificació	
Resum	Envia un missatge a un empleat.
Actors	**Directiu, Cap departament, Agent**
Casos relacionats	Iniciar Votació, Finalitzar Votació, Processar idees, Monitoritzar Projectes
Precondició	S'ha indicat un destinatari o destinataris.
Postcondició	S'envia un missatge al destinatari.
Flux normal	▪ El usuari introdueix el text de la notificació i especifica els arxius adjunts (si n'hi ha). ▪ El sistema obté del directori actiu (o d'una base de dades) l'adreça de correu del empleat i envia el missatge.

Figura 8: CDU - Enviar notificació.

CDU006 Llistar projectes	
Resum	Mostra una llista de tots els projectes actius.
Actors	**Directiu**
Casos relacionats	Iniciar votació, Finalitzar votació
Precondició	El usuari està autentificat.
Postcondició	Cap
Flux normal	▪ El sistema mostra una llista amb tots els projectes que estan *iniciats* o *en votació*, correctament agrupats per estat i ordenats per data. ▪ Per a cada projecte del llistat, el usuari podrà sol·licitar l'execució dels casos d'ús Iniciar votació i Finalitzar Votació.

Figura 10: CDU - Llistar projectes.

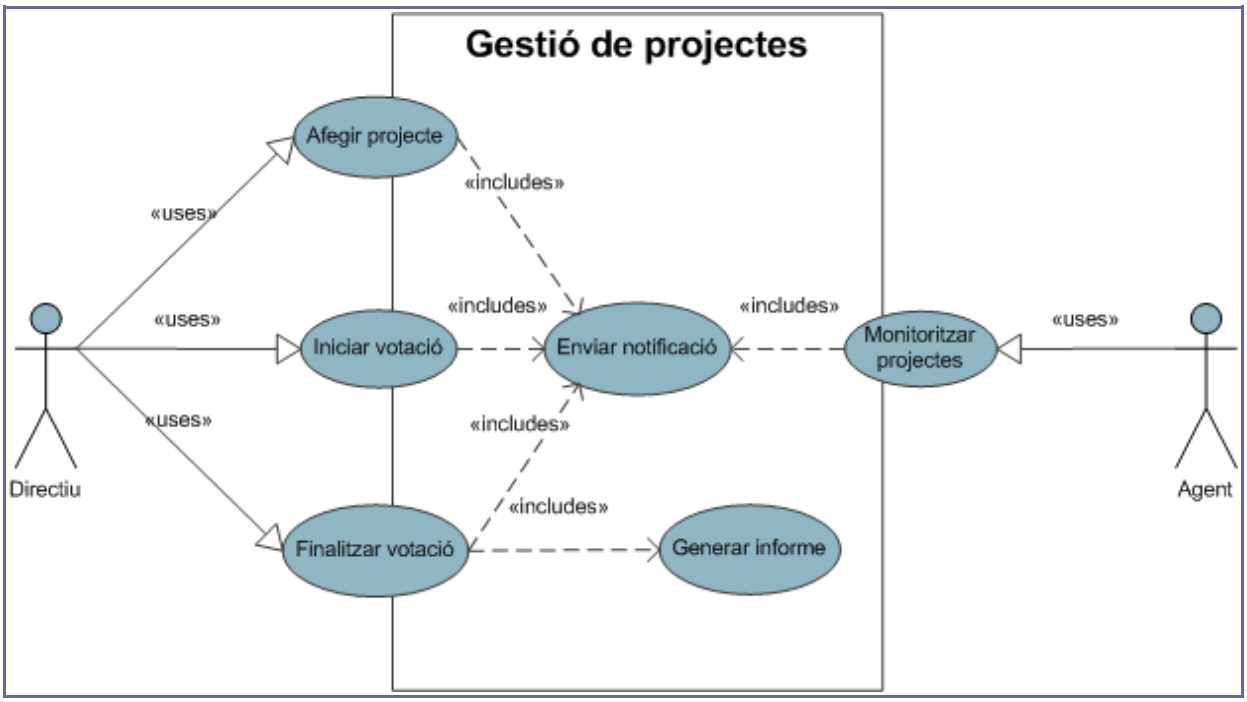


Figura 9: Casos d'ús - Gestió de projectes

3.2 Gestió d'idees

CDU007 Afegir idea	
Resum	Afegeix una nova idea a un projecte.
Actors	**Ideòleg**, **Agent**
Casos relacionats	Etiquetar idea
Precondició	El usuari està autentificat.
Postcondició	L'idea queda associada al projecte.
Flux normal	■ El sistema mostra una llista de projectes iniciats. ■ El usuari selecciona un projecte. ■ El sistema mostra el formulari per afegir idees. ■ El introdueix la descripció de la idea. Es possible afegir etiquetes a la idea mitjançant el cas d'ús Etiquetar idea. ■ El sistema emmagatzema la informació introduïda.
Observacions	El agent executa aquest cas d'ús quan processa les idees rebudes per correu electrònic.

Figura 11: CDU - Afegir idea.

CDU008 Processar idees	
Resum	Permet al cap del departament validar o rebutjar les idees rebudes.
Actors	**Cap de departament**
Casos relacionats	Enviar Notificació
Precondició	El usuari s'ha autentificat.
Postcondició	La idea quedarà acceptada o rebutjada.
Flux normal	■ El sistema mostra una llista de les idees pendents de processar. ■ L'usuari selecciona una idea. ■ El sistema mostra les etiquetes associades a la idea. ■ L'usuari valida o rebutja la idea.
Alternativa	■ Si l'usuari rebutja la idea s'ha d'especificar un motiu, que s'enviarà al ideòleg.

Figura 12: CDU - Processar idees.

CDU009 Votar	
Resum	Permet votar una idea.
Actors	**Ideòleg**
Precondició	L'usuari està autentificat.
Precondició	L'usuari no pot votar les seves pròpies idees.
Precondició	L'usuari només pot votar un cop per projecte.
Postcondició	L'idea seleccionada suma un vot.
Flux normal	▪ El sistema mostra una llista amb totes les idees que han estat acceptades per a un projecte determinat. ▪ El usuari en pot seleccionar una i votar-la. ▪ El sistema afegeix el vot.

Figura 13: CDU - Votar.

CDU010 Processar correu electrònic	
Resum	Processa les idees afegides mitjançant el correu electrònic.
Actors	**Agent**
Casos relacionats	Enviar Notificació
Precondició	
Postcondició	Es processen tots els correus pendents.
Flux normal	▪ L'usuari revisa el compte de correu i processa tots els missatges pendents de llegir. ▪ Per cada missatge trobat: ➔ Es comprova que el projecte al qual fa referència encara permet afegir idees. ➔ S'afegeix la idea al projecte i s'envia una notificació al usuari.
Alternativa	▪ Si el projecte al qual fa referència un missatge ja no admet noves idees es descarta el missatge sense cap notificació.
Alternativa	▪ Si no es possible identificar a quin projecte fa referència un missatge es notificarà al usuari.

Figura 14: CDU - Processar correu electrònic

CDU011 Cercar	
Resum	Cerca d'idees utilitzant les etiquetes com a criteri de cerca.
Actors	**Ideòleg, Cap de departament, Directiu**
Precondició	El usuari està autentificat.
Postcondició	S'obté una llista de les idees que compleixen el criteri de cerca.
Flux normal	▪ L'usuari introdueix les etiquetes per les quals vol realitzar la cerca. ▪ El sistema retorna una llista amb totes les idees que tenían alguna de les etiquetes indicades pel usuari.

Figura 15: CDU - Cercar

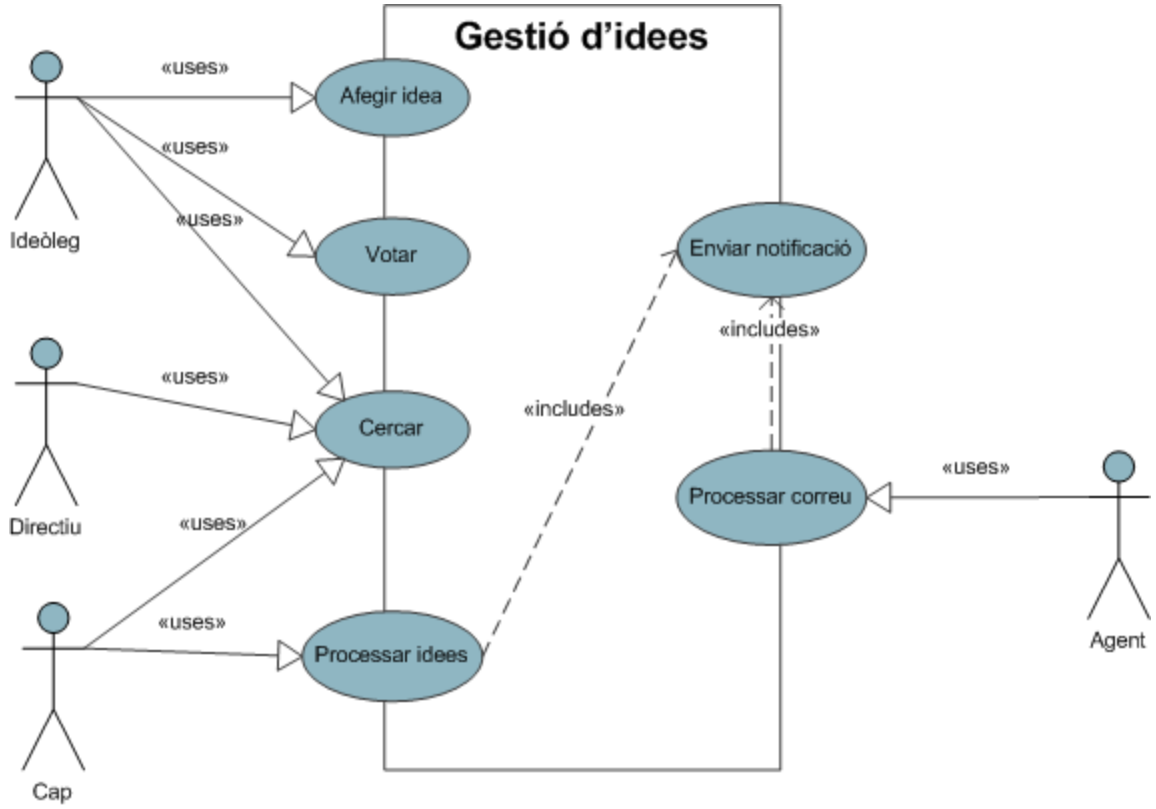


Figura 16: Casos d'ús - Gestió d'idees.

3.3 Mòdul d'estadístiques

CDU011 Consultar estadístiques projecte	
Resum	Permet realitzar un anàlisi estadístic d'un determinat projecte.
Actors	**Directiu**
Casos relacionats	Consultar estadístiques empleat
Precondició	L'usuari està autentificat.
Postcondició	El sistema mostra les estadístiques del projecte seleccionat.
Flux normal	▪ El sistema mostra una llista de tots els projectes finalitzats. ▪ El usuari en selecciona un. ▪ El sistema mostra un gràfic que representa la proporció d'idees acceptades i idees rebutjades. .

Figura 17: CDU - Consultar estadístques projecte.

CDU012 Consultar estadístiques ideòleg	
Resum	Permet realitzar un anàlisi estadístic d'un determinat ideòleg.
Actors	**Directiu**
Casos relacionats	
Precondició	L'usuari està autentificat.
Postcondició	El sistema mostra les estadístiques del ideòleg seleccionat.
Flux normal	■ El sistema mostra una llista de tots els ideòlegs. ■ El usuari en selecciona un. ■ El sistema mostra un gràfic que representa el total d'idees per projecte.

Figura 18: CDU - Consultar estadístiques ideòleg.

3.4 Model conceptual

El model conceptual, que representa el espai del problema, es la base per a la identificació de les classes. Les relacions del model conceptual són, per normal general, relacions entre els elements del món real.

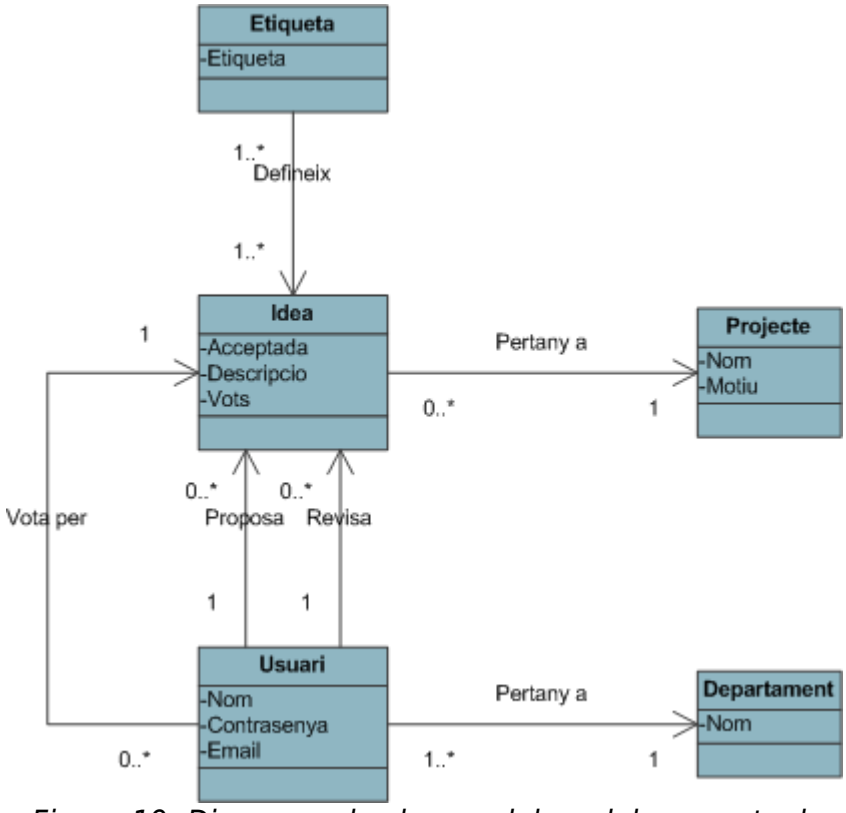


Figura 19: Diagrama de classes del model conceptual.

En aquest projecte, gràcies al ús de LINQ to SQL, les classes entitat es generen automàticament a partir de les taules de la base de dades.

4 Disseny

4.1 Arquitectura global

L'aplicació està basada en una arquitectura en tres capes. Les arquitectures en tres capes permeten delimitar les responsabilitats de cadascuna d'elles; al mateix temps una capa en concret desconeix com funcionen la resta de capes. Això vol dir que la capa de presentació desconeix completament d'on provenen les dades que està mostrant i, el més important, no té cap necessitat de saber-ho.

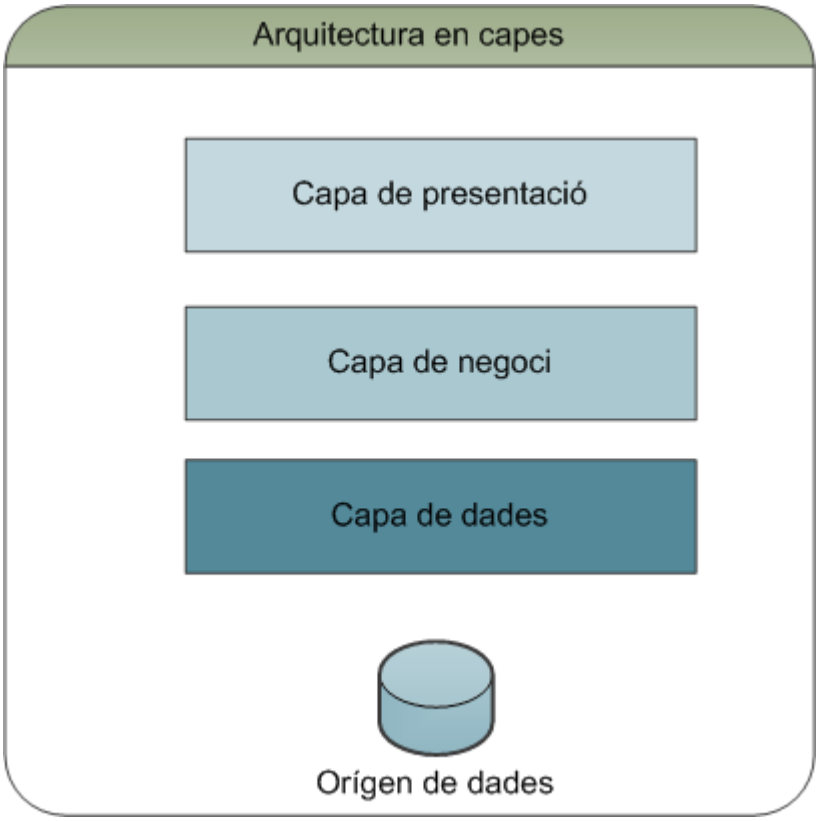


Figura 20: Arquitectura en capes.

4.1.1 La capa de presentació

És la capa amb la que interacciona el usuari i està formada pels formularis web. La capa de presentació utilitza el llenguatge de marques XHTML i les fulles d'estil CSS per mostrar la informació.

El ús de XHTML i CSS permet separar per complert la presentació de les dades, s'utilitza el llenguatge XHTML per estructurar les dades i les fulles d'estil per decidir com s'han de mostrar.

4.1.2 La capa de negoci

Aquest nivell està format per una sèrie de classes que comuniquen la capa de presentació amb la capa de dades i que defineixen la lògica del negoci. En aquest cas la capa de negoci estaria implementada en les classes que hem anomenat Gestor. La capa dels objectes del negoci respon als events de la capa de presentació, obté les dades necessàries, les transforma i les retorna a la capa de presentació.

4.1.2 La capa de dades

Finalment trobem la capa de dades que, evidentment, és la capa que treballa directament amb les dades. En aquesta capa es realitzen les consultes a la font de dades i es retornen les dades obtingudes a la capa de negoci.

La capa de dades de l'aplicació s'ha creat utilitzant el component LINQ to SQL. Aquest component, novetat del Visual Studio 2008, permet treballar directament amb objectes o entitats en lloc de tenir que fer-ho directament sobre la base de dades. És a dir, permet modelar una base de dades utilitzant objectes classes del .NET.

Així doncs, la capa de dades tampoc treballa directament sobre les dades, sinó que treballa sobre les entitats modelades (objectes) i és el component de LINQ to SQL qui s'encarrega de traduir a SQL les operacions que realitzem sobre els objectes.

A títol personal, el llenguatge de consultes integrat LINQ i les seva aplicació a les bases de dades SQL m'ha semblat, sens dubte, el punt més fort de la última versió del llenguatge C#. El ús de LINQ to SQL facilita la feina amb les bases de dades i ens allibera de tenir que implementar classes per la persistència.

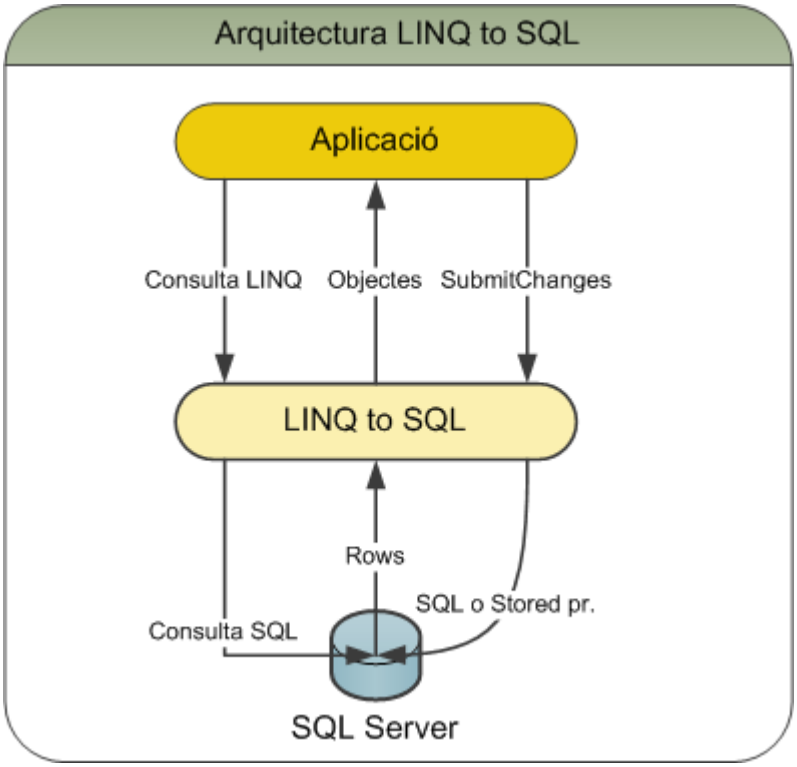


Figura 21: Arquitectura LINQ to SQL.

Cal tenir en compte que el ús de LINQ no és limita al SQL, podem emprar consultes LINQ sobre qualsevol font de dades, com per exemple aquesta sentencia que podem trobar al fitxer GestorIdees.cs:

```
var tags = from t in Etiquetes.Split(SEPARADOR_ETIQUETES) select t;
```

La sentencia anterior obté les etiquetes d'una cadena de text de manera que després podem utilitzar un iterador per obtenir cada una de les etiquetes. Tot i que l'exemple es senzill ja permet entreveure la potència i versatilitat del llenguatge.

4.2 Decisions tecnològiques

La primera decisió que calia prendre era si decantar-se per el nou framework MVC (Model – Vista – Controlador) o si utilitzar el, ja clàssic, *Web Forms*. Personalment m'interessava molt més el *framework* MVC, una implementació del patró de disseny Model – Vista - Controlador, però en la fase inicial d'investigació em vaig adonar que, de moment, no s'oferia suport per a ASP.NET AJAX.

Posteriorment vaig poder comprovar que les crides asíncrones es podrien haver implementat utilitzant llibreries de JavaScript com JQuery, però el projecte ja estava en marxa i era massa tard per tornar enrere.

Un dels objectius del projecte és aprofundir, com més millor, en les novetats que incorpora el llenguatge C#. Això implica que no es podia deixar escapar l'oportunitat de descobrir el apassionant món del llenguatge de consultes integrat LINQ. Com ja hem comentat en el apartat d'arquitectura de l'aplicació l'accés a dades és realitzarà utilitzant el component LINQ to SQL.

Per a complir amb el requeriment d'oferir suport per a dispositius mòbils s'ha utilitzat la tecnologia ASP.NET *Mobile Forms* que s'encarrega automàticament de generar planes web que siguin compatibles amb el dispositiu.

Finalment la interfície gràfica s'ha realitzat utilitzant els estàndards XHTML i CSS que, com ja s'ha comentat abans, permeten separar completament les dades de la presentació. Amb la finalitat d'enriquir la interfície d'usuari s'ha afegit el ús de ASP.NET AJAX en totes les planes mitjançant el component *UpdatePanel*. Aquest component permet que tots els controls que es col·loquen dins del panell siguin capaços, automàticament, de fer crides asíncrones al servidor. D'aquesta manera es simplifica el procés de convertir qualsevol plana al model d'aplicació web Ajax.

El cicle normal d'una plana web implica que cada cop que cal processar un event (com ara prémer un botó o seleccionar un element d'una llista) cal enviar la plana sencera al servidor. Aquest processarà la petició del client i tornarà a enviar una nova plana amb informació actualitzada. No és difícil veure que aquest procés es costós en quan a temps.

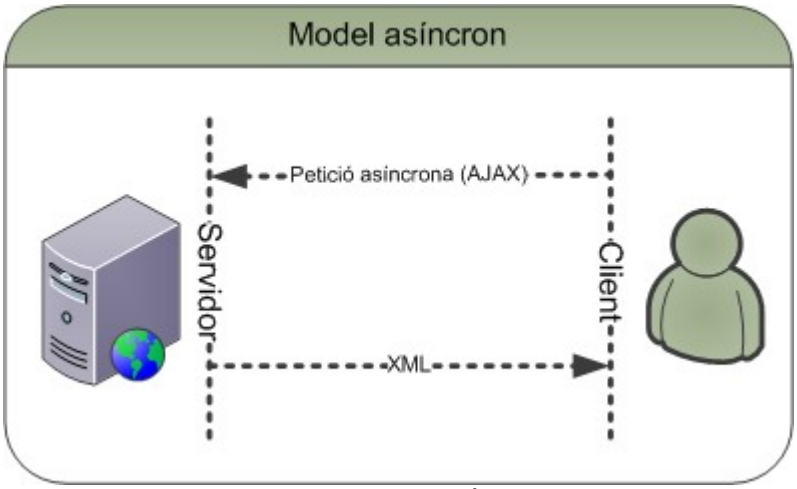


Figura 22: El model asíncron. AJAX.

AJAX permet realitzar connexions asíncrones sense tenir que enviar i rebre tota la informació cada cop. Cada petició asíncrona feta amb AJAX obté una resposta del servidor en format XML. D'aquesta manera només s'actualitzen aquelles parts de la plana que ho requereixen.

Tot i que el ús de components com el UpdatePanel milloren notablement l'experiència del usuari al utilitzar l'aplicació, la veritable potència del framework AJAX radica en la possibilitat de cridar serveis web des de Javascript. És per això que la plana de cerques s'ha implementat utilitzant aquest mètode. La quantitat de dades que cal transmetre a cadascuna d'aquestes crides es limita a les dades que realment sol·licita el client, no és transmet informació sobre el estat de la plana (el conegut *viewstate* de les planes ASP.NET) ni cap altre mena d'informació. Això fa que aquest tipus de crides siguin realment ràpides, aconseguint que el usuari tingui la sensació d'estar treballant amb dades que són a la seva pròpia màquina.

Per a la composició o disseny de les planes s'a utilitzat un *framework* CSS que s'encarrega de tenir en compte les peculiaritats de cadascun dels navegadors. D'aquesta manera el desenvolupament del projecte s'ha pogut centrar en la construcció de l'aplicació, i no en resoldre les deficiències i peculiaritats dels navegadors existents.

Per a la realització dels gràfics estadístics s'ha utilitzat un component gratuït desenvolupat amb Silverlight anomenat Visifire. És un component que permet obtenir resultats professionals i és altament configurable.

El informe en format PDF es genera utilitzant una llibreria d'ús gratuït anomenada iTextSharp que permet la creació de documents PDF dinàmicament des del codi.

4.3 Diagrama de classes

L'arquitectura en capes de l'aplicació permet concentrar tota la lògica del projecte en unes classes anomenades Gestors. Cada gestor s'encarrega d'un subsistema de l'aplicació. La classe IdeesDataContext és una classe que conté tots els mètodes d'accés a dades i és el únic punt d'accés al model d'entitats creat per el component LINQ to SQL. D'aquesta manera el accés a les dades queda completament centralitzat, i el que és més, les classes de la capa de negoci desconeixen per complert quin és el model de dades. Pel que fa a la capa de negoci tot funcionarà igual si es canvia el motor de bases de dades.

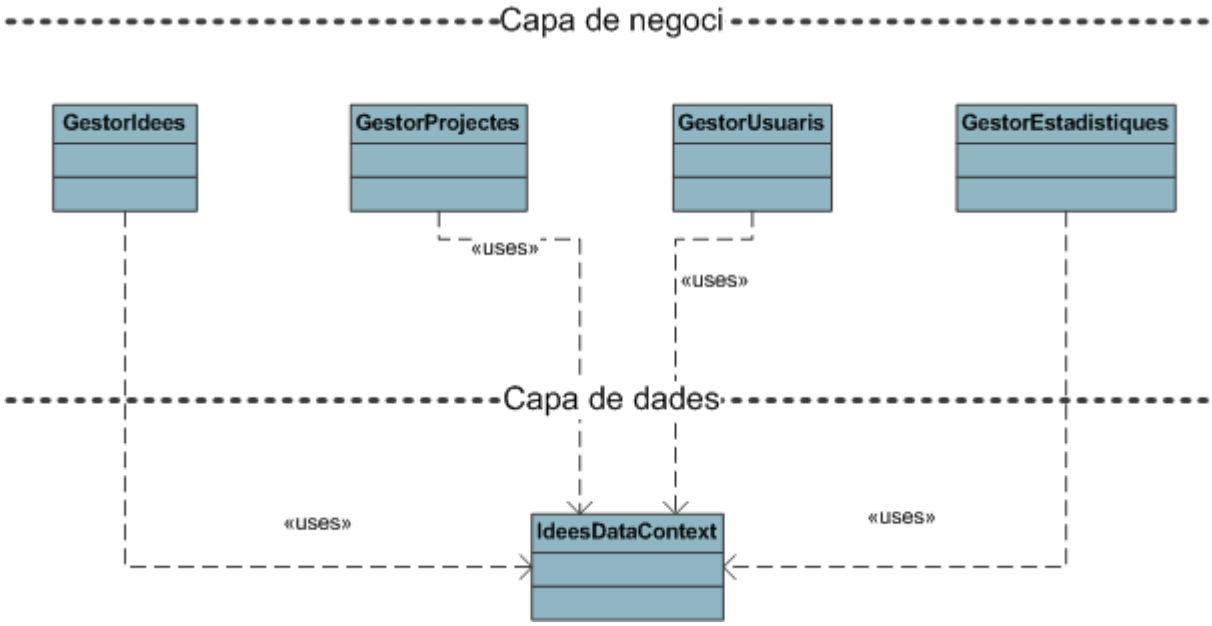


Figura 23: Classes gestor.

El diagrama anterior només mostra els gestors que utilitzen el accés a dades, hi ha doncs, altres gestors que no requereixen d'accés a les dades. L'apartat destinat a la persistència detallarà les classes entitat i les relacions entre elles.

A continuació un diagrama de classes ampliat on es poden veure la majoria dels gestors i les dependències entre ells. La classe Agent depèn dels gestors per a realitzar algunes de les seves tasques, però no s'han representat aquestes relacions en el diagrama per no saturar-lo.

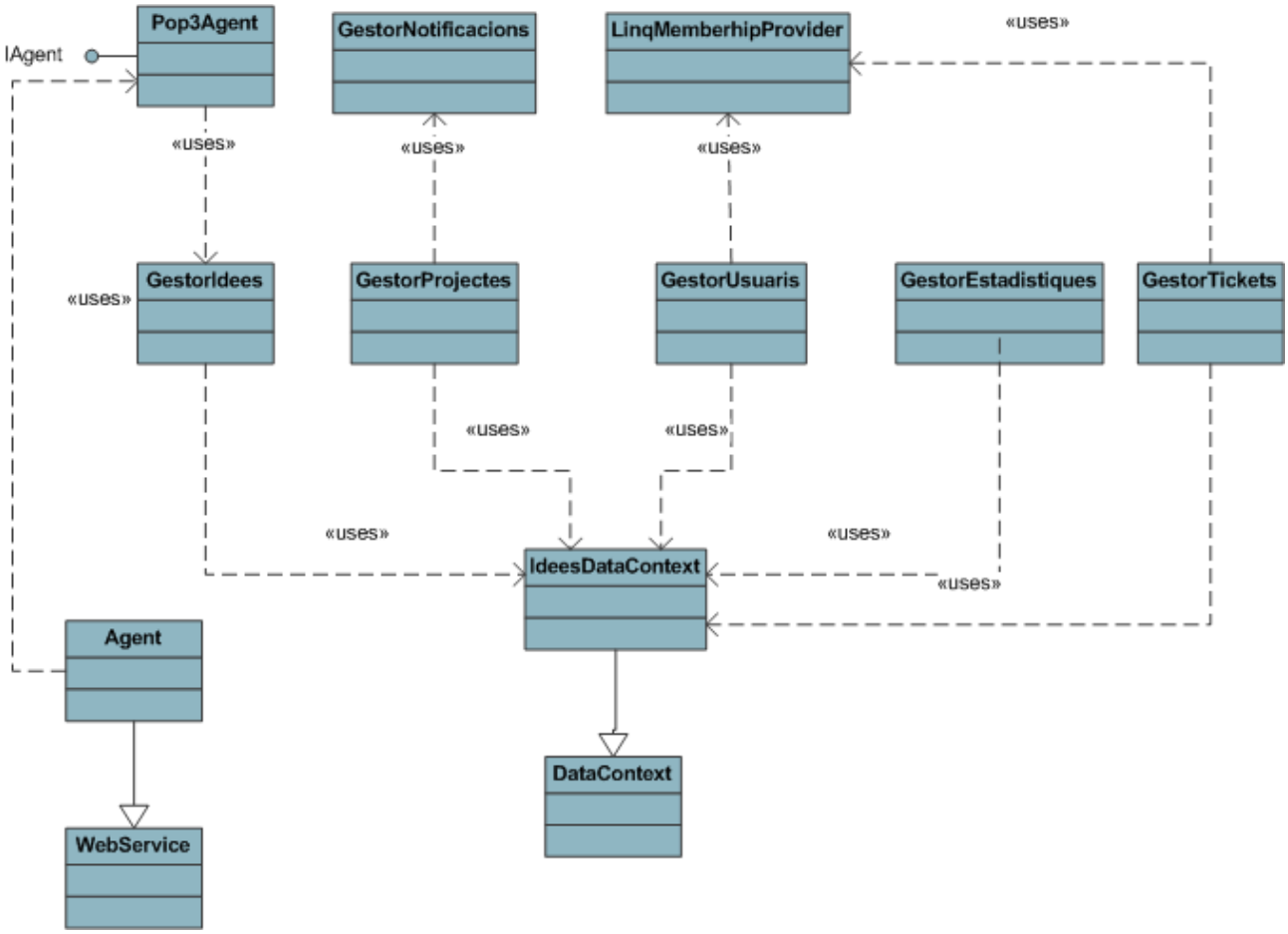


Figura 24: Diagrama de classes ampliat.

4.4 Disseny de la persistència

El ús del component LINQ to SQL ens ha alliberat de tenir que realitzar un disseny específic per a la persistència. El propi component realitza el modelat de la base de dades i gestiona la persistència. El únic inconvenient que presenta el modelat és que no es reconeixen les relacions de molt a molts, com per exemple la relació usuari projecte, on un usuari pot participar en més d'un projecte i un projecte està assignat a més d'un usuari. Per resoldre aquest petit inconvenient, cal utilitzar classes que representin aquesta relació, de la mateixa manera que al dissenyar la base de dades cal crear una taula intermitja per representar aquest tipus de relacions. A continuació es mostra el modelat de la base de dades a entitats, es pot veure clarament com s'han resolt les relacions de molts a molts.

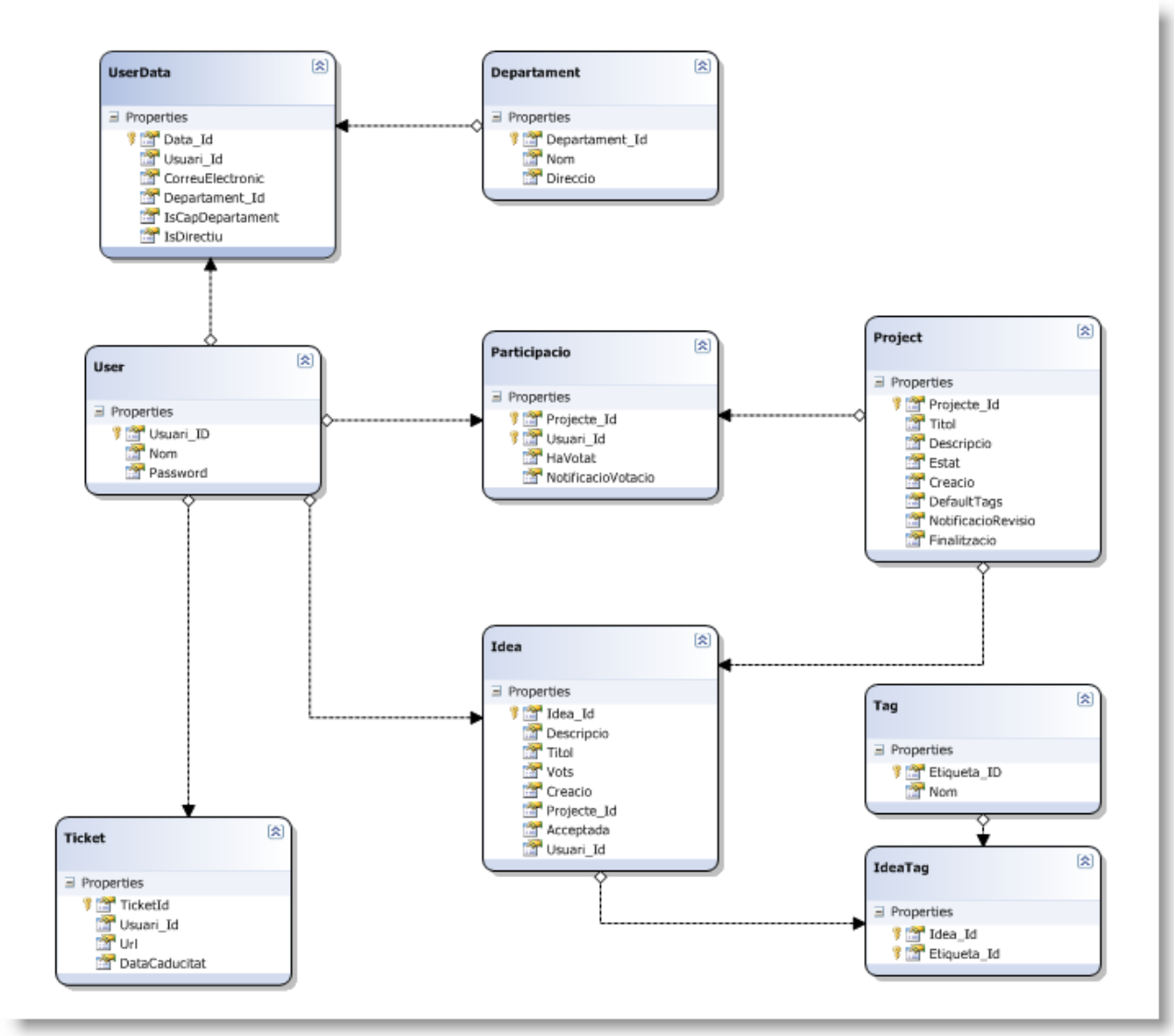


Figura 25: Modelat de la base de dades.

Cal tenir present que aquest diagrama representa classes i no taules en una base de dades. Aquí estan representades les classes entitat del projecte i les relacions entre elles.

4.5 Disseny de la base de dades

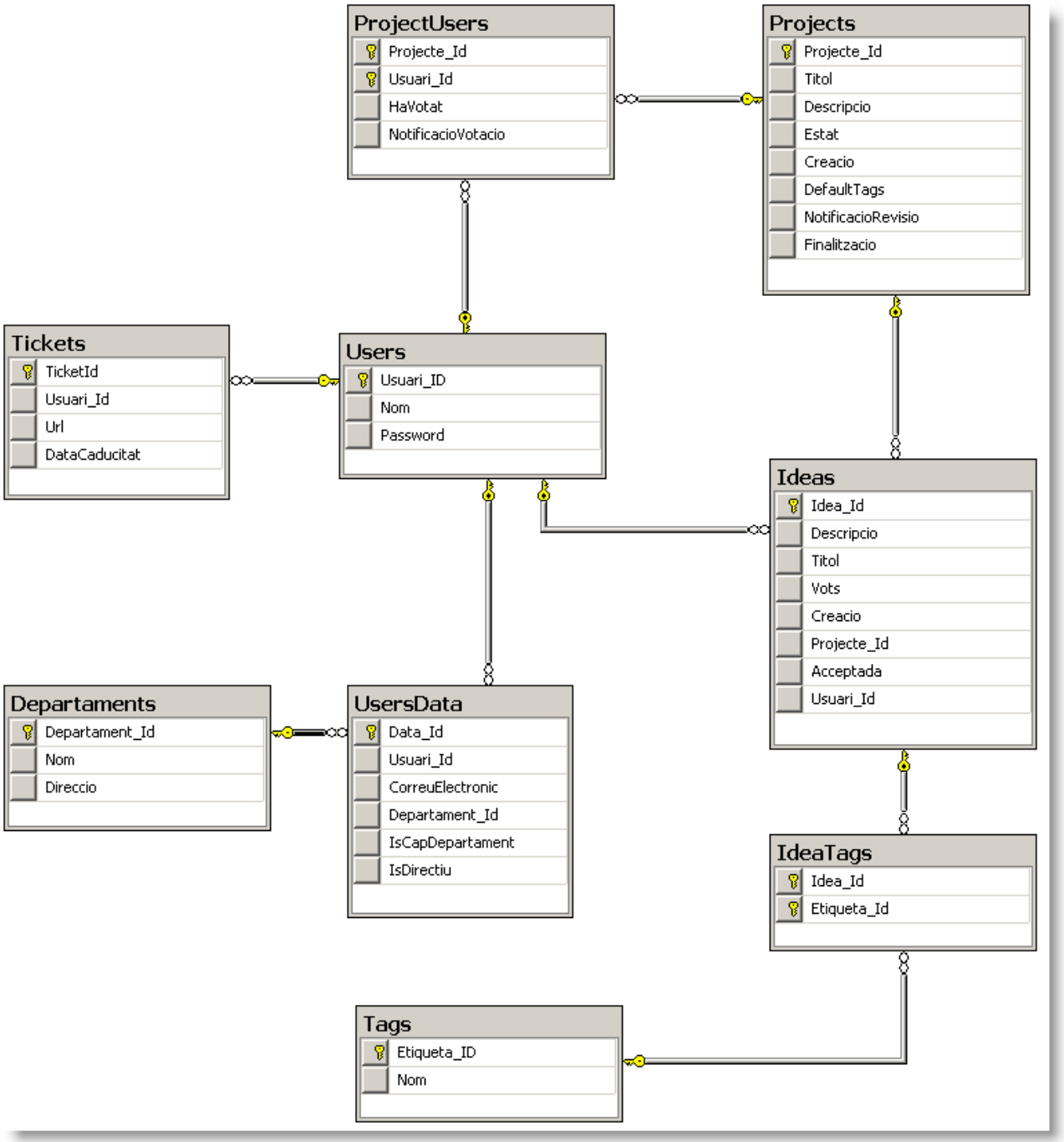


Figura 26: Base de dades.

5 Implementació

En aquest capítol es destaquen aquells elements de l'aplicació que ens semblen més interessants. Es tracta d'explicar de forma senzilla com funcionen els principals elements de l'aplicació i de detallar aquells problemes amb els que un s'ha trobat en el moment de la implementació.

5.1 Gestió d'usuaris

La informació sobre els comptes de correu i despertaments es pot obtenir bé d'unes taules a la base de dades o bé des de el propi directori actiu. Això implica que el gestor obtindrà la informació d'un lloc o altre segons la configuració. Realitzar consultes sobre la base de dades del directori actiu és relativament senzill utilitzant la classe *DirectorySearcher*, però el que ja no ha resultat tant senzill era determinar els rols de cada usuari (ideòleg, cap de departament i directiu). Aquest problema s'ha resolt indicant en el fitxer de configuració el nom del departament de direcció i el títol (o càrrec) que correspon amb els caps de departament.

El model d'autenticació d'usuaris que ens facilita ASP.NET treballa amb la seva pròpia base de dades. És a dir, cal configurar una base de dades específica que conté la informació dels usuaris i, si cal, un altre amb detalls sobre els rols disponibles. Com que no es volia afegir una nova base de dades, ha estat necessari implementar un sistema propi de validació d'usuaris que treballi amb la base de dades existent. Per fer-ho ha calgut derivar de les classes *MembershipProvider* i *RoleProvider.* El mètode més destacable és el encarregat de validar els usuaris:

```
public override bool ValidateUser(string username, string password)
{
    //En qualsevol cas podem entrar com a usuari root.
    if ((username == "root") && (password == ROOT_PASSWORD))
        return true;

    User u = this.db.UserByNom(username);
    if (u == null)
        return false;
    else
        return (u.Password.CompareTo(ToMd5(password)) == 0);
}
```

Figura 27: LinqMembershipProvider - Validació dels usuaris.

5.2 Gestió del correu electrònic

La gestió automatitzada del correu electrònic per part de l'aplicació presentava, principalment, dos problemes:

I. Com processar automàticament els missatges de correu electrònic?

II. Com saber a quin projecte fa referència un missatge en concret?

La solució al primer problema era relativament senzilla, només calia implementar un client POP3 que fos capaç de processar tot el correu rebut. Seguint les especificacions del protocol es va implementar un client que fos capaç d'obtenir i processar el correu entrant. El client es bastant rudimentari però es compatible amb qualsevol servidor de correu. Cal tenir en compte que el client no pot treballar amb servidors segurs, com és el cas de *Gmail*.

La part més difícil era com saber a quin projecte anava dirigida cadascuna de les idees enviades per correu electrònic. Inicialment es va pensar en fer ús de les capçaleres del missatge, però al respondre els missatges no es reenviaven les capçaleres originals. Descartada aquesta opció es va optar per fer ús del camp destinat al remitent per a afegir-hi la referència al identificador del projecte. Al enviar un missatge s'utilitza el següent format per a l'adreça del camp *from*:

Projecte #32 <agent@domini.cat>

Utilitzant expressions regulars és fàcil obtenir el identificador del projecte i així poder associar cada idea amb el seu corresponent missatge. La part més difícil en aquest cas, era l'expressió regular per a obtenir l'adreça des de la qual s'envia el missatge. Davant la impossibilitat d'aconseguir-ho amb una única expressió, es va optar per fer-ho utilitzant, consecutivament, dues expressions.

```
public static string Subject(string msg)
{
    return System.Text.RegularExpressions.Regex.Match(msg, @"(?<=Subject:\s+).*\r\n").Value.Trim();
}

public static string From(string Header)
{
    string fromLine;

    //Primer obtenim la línea sencera: From: "Roger Saladrigas" <roger@landsraad.net>
    fromLine = System.Text.RegularExpressions.Regex.Match(Header, @"From:.*\r\n").Value;
    //Obtenim l'adreça de correu electrònic
    return System.Text.RegularExpressions.Regex.Match(fromLine,
        @"\b(?<from>[a-zA-Z0-9._%-]+@(?:[a-zA-Z0-9-]+\.)+[a-zA-Z]{2,4})\b").Value;
}

public static Int32 ProjecteId(string Header)
{
    int id = 0;
    Int32.TryParse(System.Text.RegularExpressions.Regex.Match(Header,
        @"(?<=Projecte.*?#)[0-9]*").Value, out id);
    return id;
}
```

Figura 28: Expressions regulars per a extreure els camps del correu electrònic.

L'aplicació requeria d'algun sistema que, automàticament, realitzés el processament del correu entrant i que, al mateix temps, envies notificacions als usuaris. Treballant amb WebForms no era possible utilitzar un temporitzador que executés els processos automàtics cada cert temps i l'única opció que existia era programar una tasca que executés una plana web cada cert temps. Com que les opcions disponibles no eren gaire elegants, es va optar per crear una petita aplicació amb Win Forms que, utilitzant un temporitzador, realitzés crides a un servei web. L'aplicació, anomenada Agent, només s'encarrega de realitzar les crides periòdicament, és el servei web qui realment s'encarrega de processar el correu entrant i enviar les notificacions pertinents als usuaris.

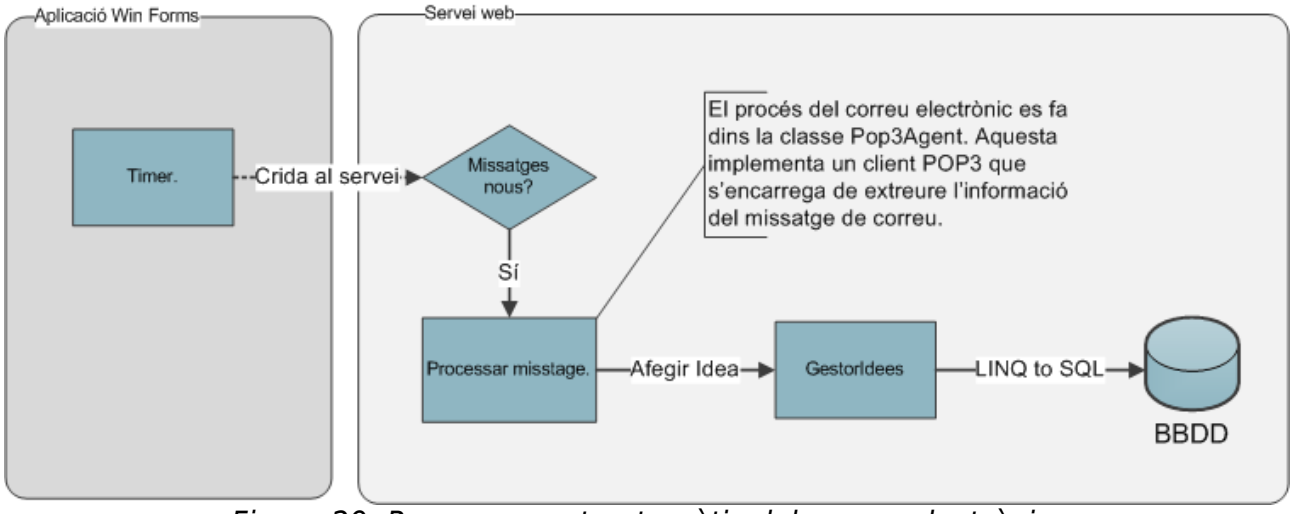


Figura 29: Processament automàtic del correu electrònic.

A hores d'ara el sistema encara presenta algunes deficiències en quant a la identificació de la codificació utilitzada al processar el correu. Aquest és un dels punts que caldria reforçar en futures línies de desenvolupament.

5.3 AJAX i els serveis web

El *framework* AJAX de ASP.NET permet implementar serveis web que es pugin cridar desde JavaScript. Per fer-ho només cal incloure la referència al servei web dins del *ScriptManager* i, automàticament, ja es podran fer crides al servei web des de JavaScript. Així de fàcil.

Per a fer crides al servei web des de JavaScript no cal incloure cap *UpdatePanel*, ni tampoc cap control de ASP.NET. Cal utilitzar un formulari HTML (sense incloure el *tag runat="server"*) i afegir events JavaScript als controls HTML quan sigui necessari.

```
<script type="text/javascript">
    function Cercar()
    {
        var cerca = $get("editCercar");
        //Crida al servei web
        Uoc.Tfc.Idees.Agent.BuscarByTag(cerca.value,llistaIdees,mostrarError);
    }

    function llistaIdees(result)
    {
        var llista = $get("resultats");
        llista.innerHTML = "";
        $get('labelResultat').innerHTML = " " + result.length.toString() + " idees trobades.";
        if (result.length == 0)
        {
            llista.innerHTML = "No hi ha cap idea amb aquesta etiqueta.";
            return;
        }

        var ulist = document.createElement("ul");
        for (var i=0;i<=result.length-1;i++)
        {
            var el = document.createElement("li");
            el.innerHTML =  result[i].Descripcio;
            ulist.appendChild(el);
        }
        llista.appendChild(ulist);
    }
```

Figura 30: Crides a un servei web des de JavaScript.

Aquest codi JavaScript permet obtenir un llistat d'idees de la base de dades i mostrar-lo en la plana HTML i tot sense necessitat de recarregar la plana en cap moment. El resultat de la cerca apareix, quasi immediatament, davant dels ulls del usuari. L'eficiència d'aquest sistema de treball és superior a la que s'obté utilitzant el component *UpdatePanel*, però també és més difícil d'implementar i requereix certs coneixements avançats del llenguatge JavaScript.

6 Captures de pantalla

A continuació es mostraran i comentaran breument les principals pantalles de l'aplicació.

6.1 Gestió d'usuaris

Al afegir un usuari, l'aplicació té en compte si la configuració especifica que cal obtenir la informació addicional del directori actiu o no. Quan totes les dades del usuari s'obtenen de la base de dades, cal especificar el correu electrònic i la informació relativa al departament.

Figura 31: Afegir usuari a la base de dades.

Per contra, si l'aplicació treballa en el mode de directori actiu, s'obtindrà la informació addicional directament del servei de directori. Cal tenir en compte que el nom d'usuari de l'aplicació ha de coincidir amb el nom d'usuari del directori actiu.

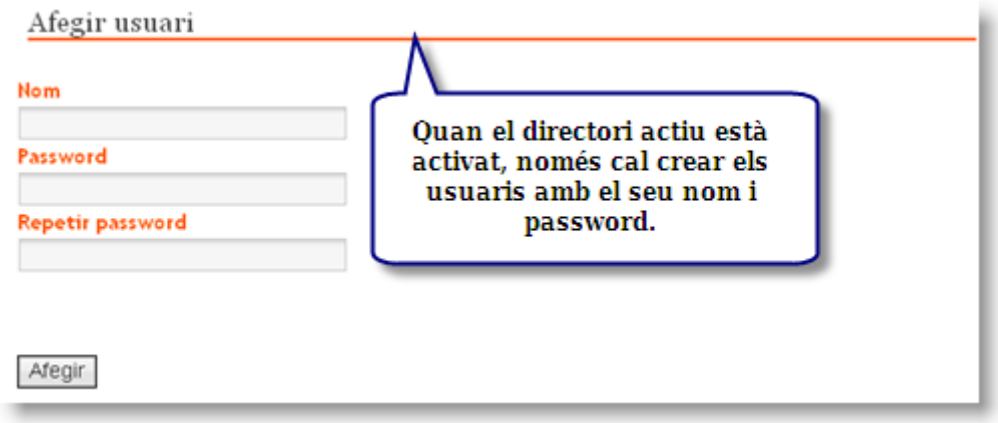


Figura 32: Suport per a Active Directory.

6.2 Gestió de projectes

La gestió de projectes inclou la possibilitat d'afegir nous projectes i la gestió dels canvis d'estat dels projectes existents.

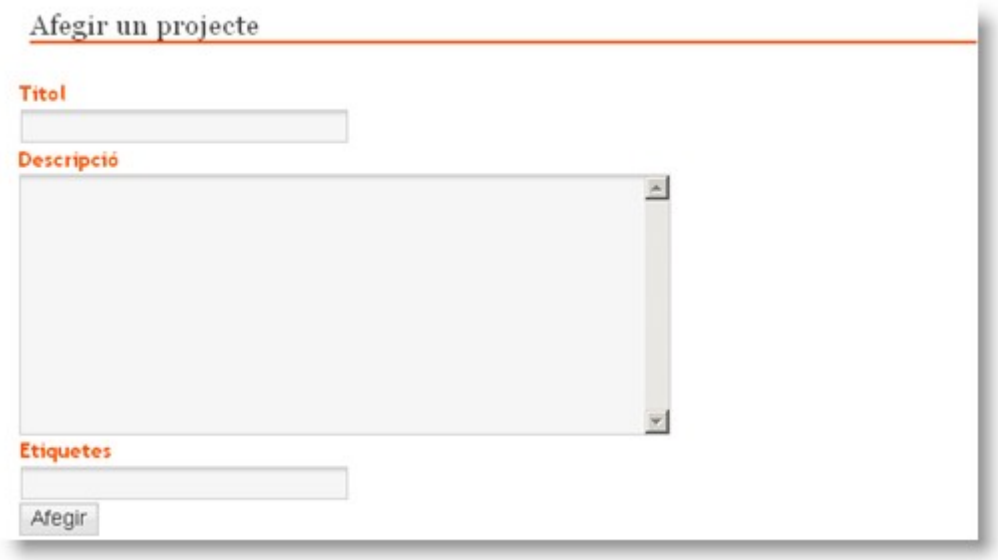


Figura 33: Afegir un nou projecte.

La llista de projectes ja iniciats permet als directius decidir quan començarà la fase de votació.

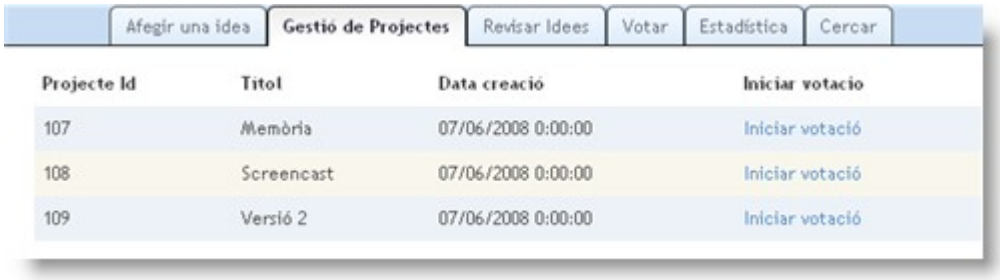


Figura 34: Llistat dels projectes iniciats.

El directiu també haurà de decidir quan es finalitza un projecte, per fer-ho disposa d'un llistat de projectes en votació.

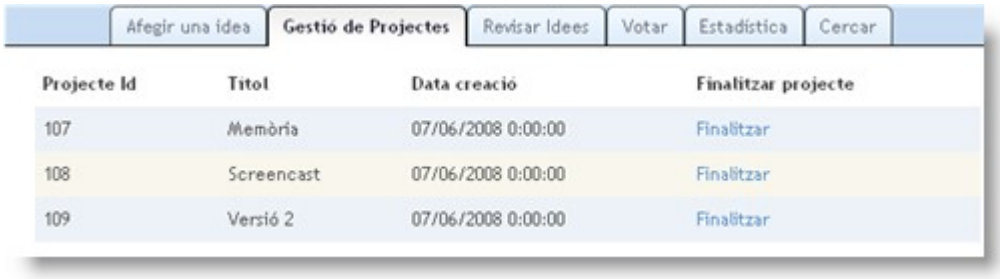


Figura 35: Llistat dels projectes en etapa de votació.

6.3 Gestió d'idees

La pantalla d'afegir idees permet seleccionar un projecte actiu i afegir-hi una idea.

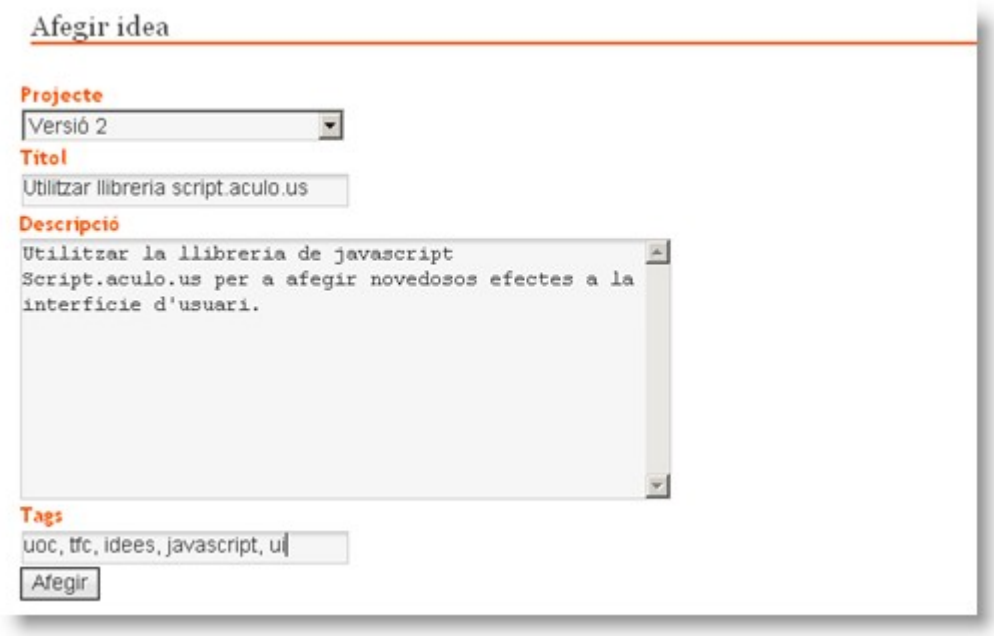


Figura 36: Afegir una idea.

Els caps de departament poden avaluar les idees proposades pels ideòlegs des de l'opció “Revisar idees” del menú principal. Un cop seleccionat el projecte a revisar, es mostren totes les idees pendents de revisió. Cadascuna d'elles conté un enllaç per aprovar-la i un per refusar-la.

Figura 37: Revisió d'idees.

Al refusar una idea cal indicar un motiu que serà enviat al ideòleg. Aquest sistema permet obrir un canal de comunicació directe entre el cap de departament i els seus empleats.

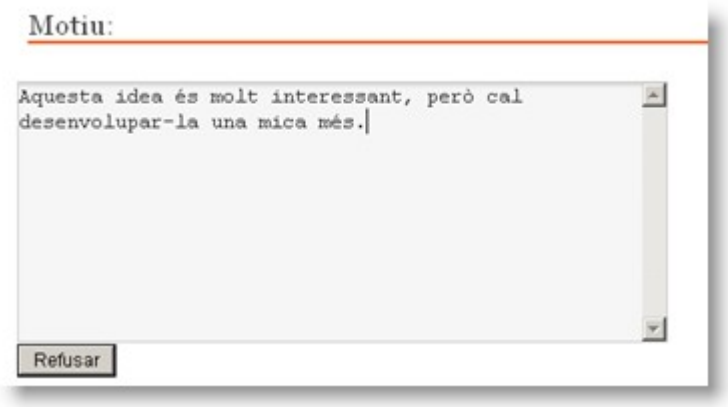


Figura 38: Refusar una idea.

També es possible revisar idees des de un dispositiu mòbil, com ara un Pocket PC.

Figura 39: Revisar idees, versió per a dispositius mòbils.

Finalment els usuaris disposen d'una plana per poder votar les idees que considerin més innovadores. Aquesta tasca es pot realitzar tant des d'una estació de treball com des d'un dispositiu mòbil.

Figura 40: Votar.

La versió per a dispositius mòbils també permet als usuaris de l'aplicació emetre el seu vot des de qualsevol lloc, sempre i quan tinguin connexió a Internet.

Figura 41: Votar des d'un dispositiu mòbil.

6.4 Gestió d'estadístiques

El mòdul d'estadístiques permet als directius obtenir informació estadística sobre tot els projecte existents. Els gràfics permeten obtenir informació ràpida sobre els projectes i els usuaris de forma visual.

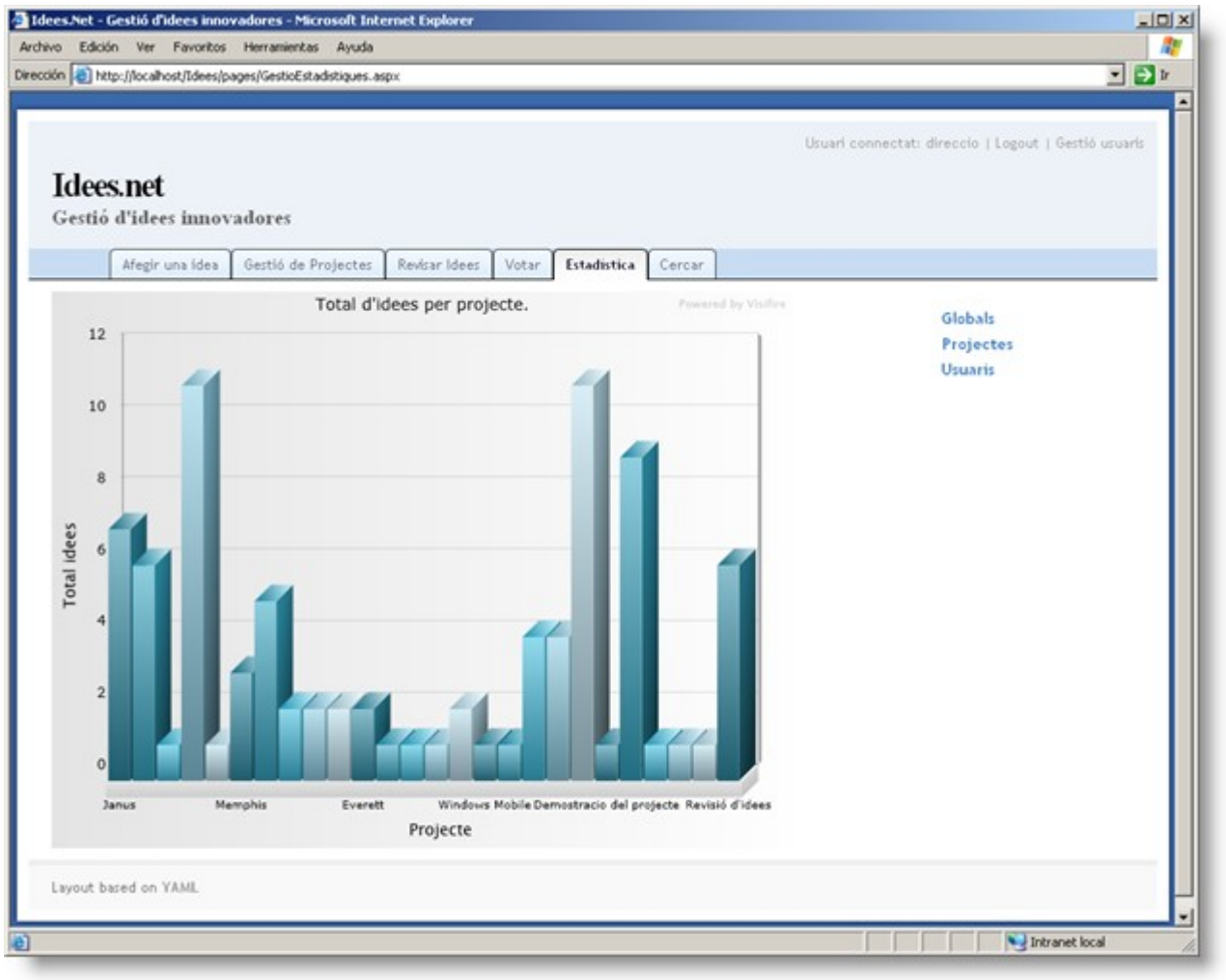


Figura 42: Estadístiques.

6.5 Cercar

Aquesta última opció permet realitzar cerques sobre la base de dades d'idees utilitzant les etiquetes o *tags* com a criteri de cerca.

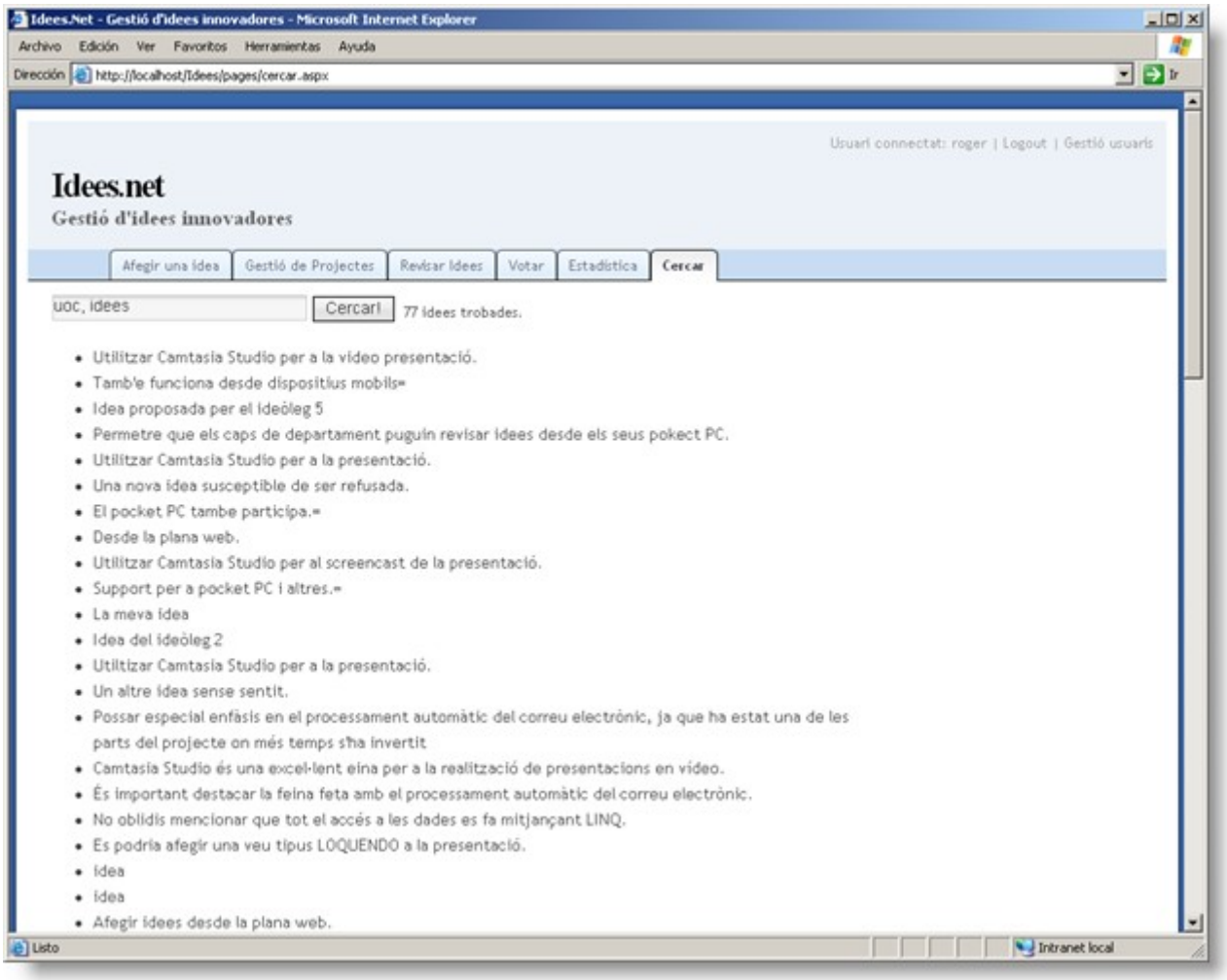


Figura 43: Cercar per etiquetes.

7 Conclusions

Al finalitzar aquest projecte s'han complert els principals objectius exposats al iniciar-lo. S'ha desenvolupat una aplicació web que complia amb tots els requisits, funcionals i no funcionals, establerts i s'han assolit coneixements bàsics de totes les tecnologies utilitzades en el projecte. Així mateix s'han consolidat coneixements adquirits al llarg dels estudis en Enginyeria tècnica en informàtica de gestió.

Tot i que el temps només ha permès tenir una primera presa de contacte amb algunes de les tecnologies utilitzades ha estat suficient per poder entreveure la potència i les possibilitats que ofereixen cadascuna d'elles.

8 Línies de desenvolupament futur

El desenvolupament del projecte s'ha realitzat en un interval de temps molt ajustat, fet que no ha permès invertir més temps en certs aspectes de l'aplicació. A continuació s'enumeren alguns aspectes que, amb una mica més de temps, es podrien afegir de cara a millorar el producte final.

- Implantar el ús de serveis webs que poden ser cridats des de AJAX. Els formularis per afegir usuaris, projectes o idees són candidats perfectes per a això.

- Afegir un sistema de control d'errors que permeti controlar totes les excepcions. D'aquesta manera totes les excepcions queden emmagatzemades i es facilita la tasca de manteniment i millora de l'aplicació. La implantació d'un sistema com el que ofereix ELMAH (Error Logging Modules And Handlers) permetria tenir un control absolut de tots els errors i excepcions.

- Millorar la interfície d'usuari. Actualment existeixen llibreries de JavaScript (com ara script.aculo.us) especialment dissenyades per a crear efectes i, en general, millorar l'experiència del usuari. A part d'incloure aquestes millores es necessari millorar el disseny general de l'aplicació.

- Ampliar el conjunt de gràfics estadístics.

- El projecte ha estat dissenyat per a ser utilitzat en una intranet, però una de les línies de desenvolupament futur més interessants seria incloure les modificacions necessàries per tal de poder convertir-lo en una aplicació utilitzable per múltiples col·lectius.

- Possibilitat d'afegir idees mitjançant una aplicació de missatgeria instantània. La naturalesa d'aquest tipus d'aplicacions permetria agilitzar els processos d'afegir idees i enviar notificacions.

9 Glossari

- **Agent**: és un procés automàtic del sistema que s'executa periòdicament.

- **Ajax**: acrònim de ***A****synchronous* ***J****avaScript* ***A****nd* ***X****ML*. Tecnologia que fa ús de JavaScript i el llenguatge de marques XML per a crear aplicacions web més interactives.

- **Cap de departament**: és l'encarregat d'acceptar o rebutjar les idees proposades pels ideòlegs que pertanyen al seu departament.

- **CSS**: **C**ascading **S**tyle **S**heets. Un llenguatge formal utilitzat per a definir la presentació d'un document estructurat en HTML (o XHTML).

- **Directiu**: és l'encarregat de gestionar els projectes. Reben els informes finals de cada projecte.

- **Directori actiu**: traducció del nom anglès Active Directory. Es terme que utilitza Microsoft per referir-se a la seva implementació del protocol LDAP.

- **Estat o fase del projecte**: els projecte té un cicle de vida dividit en tres estats: *iniciat*, *en votació* i *finalitzat*.

- **Ideòleg**: son els encarregats d'enviar noves idees.

- **Informe final**: és un informe en format PDF on consten les deu idees que més vots han obtingut en un determinat projecte.

- **JavaScript**: és un llenguatge de programació interpretat utilitzat en planes web.

- **LDAP**: és un protocol que permet obtenir informació relacionada amb la xarxa com ara els noms dels empleats i les seves adreces de correu.

- **LINQ**: prové de **L**anguage **IN**tegrated **Q**uery. Integra dins dels propis llenguatges de la plataforma .NET la possibilitat de realitzar consultes.

- **PDF**: del angès **P**ortable **D**ocument **F**ormat. És un format pensat per al intercanvi de documents gràcies a l'alta qualitat d'impressió que s'obté dels mateixos i les seves característiques multi plataforma.

- **POP3**: prové del terme **P**ost **O**ffice **P**rotocol. Permet a un client local obtenir els missatges de correu electrònic emmagatzemats en un servidor remot.

- **Servei web**: es un conjunt de protocols i estàndards que serveixen per intercanviar dades entre aplicacions a través d'Internet

- **TFC**: treball final de carrera.

- **XHMTL**: acrònim anglès de e**X**tensible **H**yper**t**ext **M**arkup **L**anguage. És el llenguatge de marques utilitzat per a la creació de planes web.

- **XML**: sigles per *eXtensible Markup Language*. És un llenguatge de marques que permet el intercanvi d'informació estructurada entre diferents plataformes.

- **YAML**: és un framework CSS que permet la creació de composicions amb múltiples columnes. El codi generat compleix amb tots els estàndards i amb suport per a tots els navegadors actuals. YAML son les sigles per a ***Y****et* ***A****nother* ***M****ulticolumn* ***L****ayout*.

10 Bibliografia

- **ELMAH – Error logging modules and handlers for ASP.NET**
  Sistema per al control d'errors d'una aplicació ASP.NET.
  http://code.google.com/p/elmah/

- **LINQ to SQL – Visual Studio 2008 Developer center**
  Component ORM del Net Framework 3.5
  http://msdn.microsoft.com/en-us/library/bb386976.aspx

- **Using LINQ**
  Weblog de Scott Guthrie, desenvolupador de Microsoft.
  http://weblogs.asp.net/scottgu/archive/2007/05/19/using-linq-to-sql-part-1.aspx

- **RFC – 1939 Protocol POP3**
  Especificació del protocol POP3.
  ftp://ftp.isi.edu/in-notes/rfc1939.txt

- **LINQ 101 Samples**
  Exemples de instruccions.
  http://msdn.microsoft.com/en-us/vcsharp/aa336746.aspx

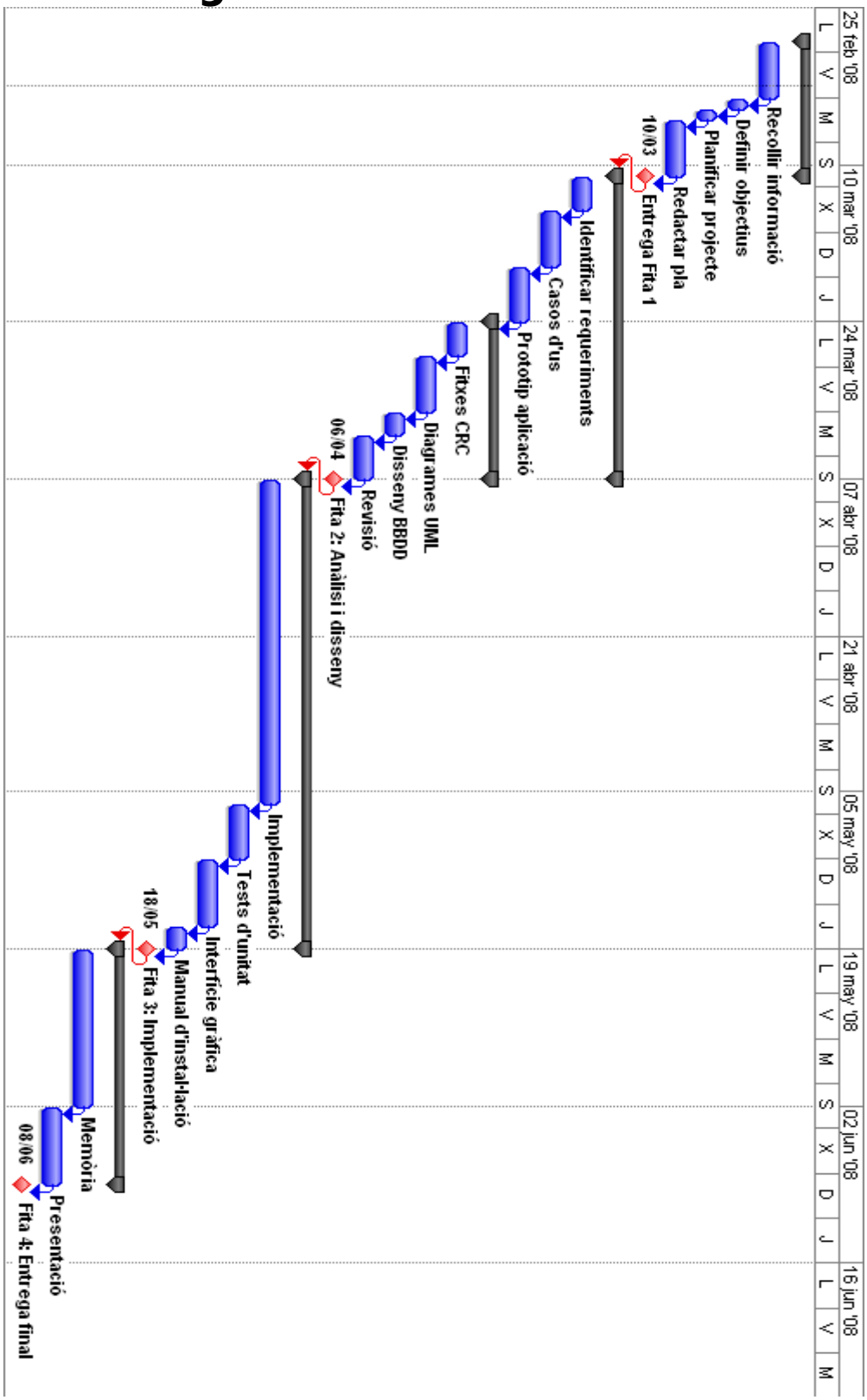
- **Official Microsoft ASP.NET Site**
  Informació sobre ASP.NET i AJAX.
  http://www.asp.net/ajax

- **Yet Another Multicolumn Layout**
  Framework CSS.
  http://www.yaml.de/en/home.html

- **Silverlight Chart – Visifire**
  Component per a crear gràfics estadístics.
  http://www.visifire.com/

- **iTextSharp**
  Creació de documents PDF dinàmicament.
  http://itextsharp.sourceforge.net/

Annex A – Diagrama de Gantt



Annex B – Classes gestores