Universitat Autònoma
de Barcelona

Desenvolupament d'un gestor d'idees

Memòria del projecte

d'Enginyeria Tècnica en

Informàtica de Gestió

realitzat per

Narcís Davins Riu

i dirigit per

Xavier Verge Mestre

Escola d'Enginyeria

Sabadell, *Juny* de *2010*

El sotasignat, ***Ignacio Cofré Gómez***,
de ***Softonic***,

CERTIFICA:

Que el treball al que correspon la present memòria ha estat realitzat sota la seva supervisió per en ***Narcís Davins Riu***

I per a que consti firma la present.

Sabadell, ***Juny*** de ***2010***

Signat: ***Ignacio Cofré Gómez***

El sotasignat, ***Xavier Verge Mestre***,
professor de l'Escola Universitària d'Informàtica de la UAB,

CERTIFICA:

Que el treball al que correspon la present memòria ha estat realitzat sota la seva supervisió per en ***Narcís Davins Riu***

I per a que consti firma la present.

Sabadell, ***Juny*** de ***2010***

Signat: ***Xavier Verge Mestre***

Índex

Resum introductori

Aquesta memòria explica el desenvolupament del projecte de final de carrera d'Enginyeria Tècnica en Informàtica de Gestió proposat per a l'empresa Softonic.

El projecte sorgeix per intentar donar una solució per gestionar efectivament el talent dels treballadors de l'empresa per tal de que l'empresa en pugui treure profit. Tot i que hi ha altres aplicacions que realitzen aquesta funció, es vol una aplicació feta a mida que pugui complir tots els requisits que es desitgen, tot assolint una total integració en l'aplicació interna de l'empresa.

El projecte ha seguit totes les fases d'un projecte de desenvolupament de software: anàlisi, disseny, desenvolupament i proves. Essent la fase d'anàlisi i disseny més llarga que les d'un projecte que comenci de zero, ja que en aquest cas s'havia d'analitzar l'aplicació interna existent per poder-hi integrar la que s'havia de desenvolupar.

1 Introducció

1.1 Descripció

Actualment, a les empreses tecnològiques la innovació forma part imprescindible en elles. Poder-se mantenir en un mercat no només depèn de la perfecció dels productes existents, sinó també de la definició dels productes i estratègies futures.

En empreses petites pot semblar prescindible disposar d'una tecnologia específica per realitzar aquesta gestió, però avui en dia, gràcies a Internet és possible donar accés a una àmplia comunitat que pot aportar moltes idees al teu projecte d'una manera extremadament fàcil. És sobretot en casos com aquest, on el nombre d'idees a tractar pot ser elevat, que pren molta importància disposar de mètodes i tecnologies eficaces que permetin gestionar correctament aquesta informació, i donar pesos específics d'importància a cada proposta.

Aquest projecte formarà part d'una aplicació interna de l'empresa que actualment aglutina vàries eines diferents: Keep In Tonic, Tweettonic, Videos i Team.

Keep In Tonic és l'eina principal de l'aplicació i és la que li dona nom al conjunt de l'aplicació interna. Aquesta és, juntament amb la missatgeria instantània i el correu electrònic, una de les eines de comunicació internes principals. En ella s'hi poden deixar missatges, tant de feina com de lleure, i apareixen en un format que recorda a una barreja entre un fòrum i una xarxa social, com ara Twitter. Les entrades de l'aplicació poden ser comentades i votades per altres usuaris, llistar les entrades d'un usuari, dels usuaris d'un departament...

Des de Tweettonic qualsevol usuari de l'aplicació pot crear entrades per al compte oficial de Softonic a la xarxa social Twitter. A més a més, aquesta eina serveix també per veure les entrades de Twitter creades des d'aquesta eina, personificant-les amb l'usuari que les ha creat.

L'eina Videos és utilitzada per mostrar vídeos relacionats amb l'empresa, des de reportatges on aquesta hi apareix fins a vídeos d'esdeveniments interns.

Team és probablement l'ultima eina que s'identifica com a tal degut a que no té una pestanya que l'enllaci. L'objectiu d'aquesta eina és col·laborar a que els treballadors es coneguin entre ells, permetre que els usuaris puguin personalitzar el seu perfil, veure el departament al qual pertanyen, els aniversaris...

1.2 Objectius

Amb aquest projecte es pretén aprofitar el talent dels treballadors alhora que se'n facilita l'expressió i recepció de les seves idees, tot fent-los més partícips del projecte de l'empresa.

A més, al tenir una eina eficaç per tractar les idees, es pretén que a través d'elles millori la productivitat i la qualitat del producte que s'ofereix als clients.

Per donar aquests objectius com a assolits caldrà valorar varis aspectes passat un any des de la seva implantació:

- Tenir un volum total d'idees mensuals equivalents a mitja per usuari, que amb el volum actual de treballadors equivaldria a més de 1000 idees passat un any del llançament de l'aplicació.
- Mitjana de 5 vots per idea.
- Un 5% de les idees proposades han estat posades en marxa.
- El temps entre la proposició de la idea, i la decisió de portar-la a terme no sobrepassi els dos mesos.

Per tal d'assolir aquests objectius es pretén desenvolupar una aplicació web que reuneixi tot un seguit de requisits enfocats a augmentar la col·laboració i implicació dels treballadors en aquests aspectes.

1.3 Estat de l'art

L'aplicació més utilitzada últimament a Internet per recollir *feedback* dels seus usuaris és Get Satisfaction: una eina web, amb diferents tipus de llicències. Aquesta aplicació ajuda a les companyies recollint preguntes, opinions, problemes i elogis en una plataforma unificada. Això permet que les accions dels usuaris puguin ser aprofitades per les companyies en diferents aspectes: desenvolupament de producte, màrqueting... És una eina flexible que s'integra fàcilment.

En la figura 1 tenim un exemple d'integració de Get Satisfaction en una pàgina web, en aquest cas en la pàgina foursquare, en la qual només cal prémer el botó feedback del lateral perquè s'obri un requadre per introduir l'aportació de l'usuari.

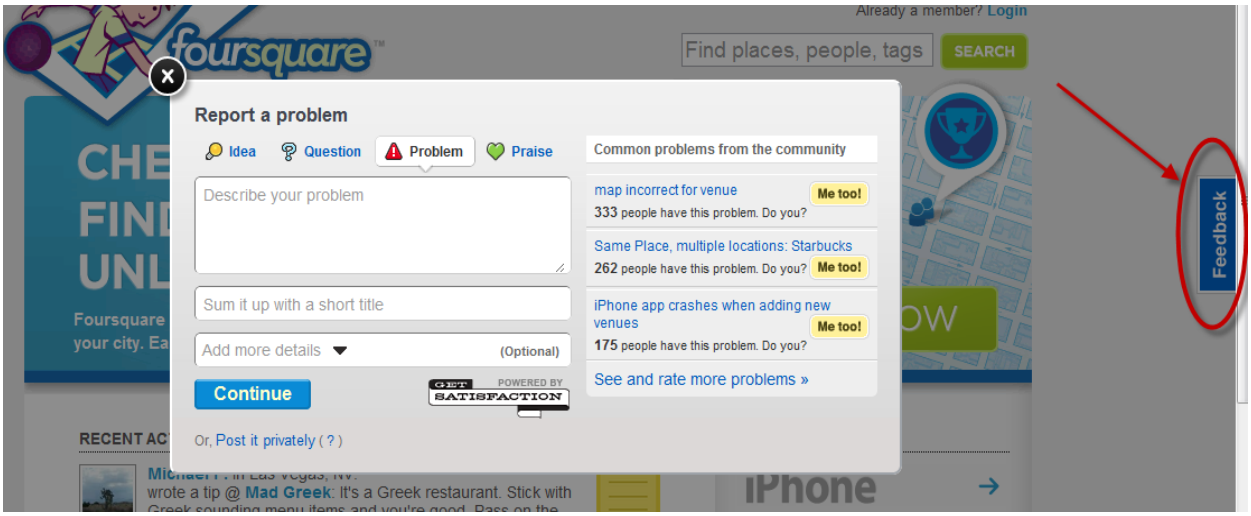


Figura 1: pàgina de Foursquare

Aquesta aplicació, però, guarda la majoria de contingut en obert a la web de Get Satisfaction, cosa que pot ser un gran problema per segons quin sigui l'ús que se'n vol donar al gestor d'idees.

1.4 Motivacions

Les principals motivacions a l'hora de fer aquest treball són aprofundir en el coneixement dels llenguatges PHP, MySQL. Llenguatges apresos profundament en l'empresa abans de l'inici de la codificació d'aquest treball, realitzant-hi un curs de desenvolupament d'aplicacions web d'alt rendiment amb tecnologia LAMP.

Una altre motivació és poder-lo incorporar a la intranet d'aquesta i veure'l en funcionament, tot escoltant-ne l'opinió d'altre gent. A més també era una gran oportunitat per fer un bon treball en una gran empresa d'Internet.

1.5 Estructura de la memòria

Aquesta memòria està estructurada en sis capítols, a part d'aquest primer d'introducció.

En el capítol dos es tracta l'explicació de la fase d'anàlisi de l'aplicació a desenvolupar i de la planificació del projecte.

En el capítol tres hi ha el marc teòric de l'aplicació, on s'expliquen algun dels trets més importants de les diferents aplicacions i tecnologies utilitzades durant el desenvolupament de l'aplicació.

En el quart capítol es troba explicada tota la fase de disseny: casos d'ús, base de dades, arquitectura de l'aplicació i interfície.

En el cinquè s'expliquen diferents convencions que s'han portat a terme a l'hora de desenvolupar l'aplicació.

En el sisè capítol hi ha les conclusions del projecte: assoliment d'objectius, desviacions que hi ha hagut en la planificació del projecte i diferents línies d'ampliació aplicables al projecte per tal d'aprofundir en els objectius en que ha estat creat.

Finalment, en el capítol set, hi ha la bibliografia principal que s'ha utilitzat pel desenvolupament d'aquest projecte.

2 Anàlisi i planificació

2.1 Requeriments funcionals

Llistes d'idees

Aquesta serà la pàgina principal de l'aplicació d'idees. En ella s'hauran de mostrar les llistes d'idees, excepte la de moderació.

Les llistes d'idees seran:

- New: idees amb estat 'new' ordenades per data de creació, de més recent a més antiga.
- Top: idees amb estat 'new' ordenades per increment de vots.
- In-process: idees amb estat 'in process' ordenades per data de creació, de més recent a més antiga.
- Implemented: idees amb estat 'implemented' ordenades per data de creació, de més recent a més antiga.
- Area: idees pertanyents a una àrea.
- Author: idees creades per un usuari.
- Department: idees creades pels usuaris d'un departament.

Introducció, modificació, eliminació d'idees

En la part central superior de la web, hi haurà d'haver disponible a totes les pàgines el formulari d'introducció d'idees, que constarà de títol, cos de la idea, etiquetes i àrea a la que pertany.

Les idees podran ser eliminades i editades en qualsevol moment pel seu responsable, mentre que el seu creador tindrà un temps limitat per a fer-ho.

A més a més, si l'idea està en estat "in process" o "implemented" s'hi podrà afegir l'identificador corresponent a la tasca associada de JIRA, el programa de gestió de tasques utilitzat a l'empresa.

Moderació d'idees.

Cada àrea tindrà designats un o més responsables.

Aquests usuaris seran els encarregats de la moderació i canvis d'estat de les idees.

Disposaran d'una pàgina on hi apareixerà un llistat de les idees de la seva àrea o àrees que els facilitarà la gestió de les seves àrees de responsabilitat.

Llistat de moderació d'idees

Les idees s'han de poder filtrar tant per les seves àrees, com per qualsevol dels estats existents. Podent aplicar els dos filtres simultàniament.

A més aquest llistat s'ha de poder ordenar per qualsevol dels camps mostrats en mode ascendent i també descendent.

Votacions

Cada usuari només ha de poder votar un cop cada idea.

Hi haurà un nombre màxim de vots setmanals per usuari, els vots restants tenen que ser visibles per a l'usuari.

Només es podran votar les idees en estat 'new'.

Subscripcions

Cada usuari ha de poder-se subscriure a qualsevol idea i a qualsevol àrea, independentment de si n'és responsable.

Enviament de correus electrònics

Hi haurà vàries accions que tenen que generar un correu electrònic:

- Creació d'una nova idea: aquesta acció té que generar un correu electrònic a les persones subscrites a l'àrea a la qual s'afegeix la idea, i als responsables de l'àrea.
- Nou comentari: aquesta acció generarà una alerta a les persones subscrites a la idea on s'afegeix el comentari.

- Canvi d'estat d'una idea: aquesta acció generarà una alerta a les persones subscrites tant a la idea com a l'àrea d'aquesta, al seu creador, i als responsables de l'àrea.
- Enviament setmanal: Periòdicament s'enviarà als responsables de cada àrea un resum amb les idees amb un major increment de vots en aquella setmana.

Interacció amb Keep In Tonic

Quan s'introdueixi una idea nova, aquesta acció haurà de ser visible per tots els usuaris en la pàgina principal de Keep In Tonic.

En el perfil d'usuari i de departament, hi ha d'haver una llista amb les darreres idees d'aquest element, de la mateixa manera que existeix la llista de últims worklogs i últims tweets.

2.2 Requeriments no funcionals

Llistes d'idees

Les llistes: New, Top, In Process i Implemented tenen que aparèixer en format pestanya, seguint el mateix estil de Lastest worklogs i Hotness de la pàgina principal de Keep in Tonic.

El llistat per defecte de la pàgina principal d'idees té que ser la llista New.

En el llistat d'àrea té que aparèixer la opció per subscriure's-hi.

Introducció, modificació, eliminació d'idees

En la fitxa d'idea hi haurà d'haver un botó per tornar a la pàgina des de la qual s'havia arribat a la idea, en el cas de no provenir de cap llistat d'idees, no hauria d'aparèixer aquest botó.

En la part central superior de la web, hi haurà d'haver disponible a totes les pàgines el formulari d'introducció d'idees, que constarà de títol, cos de la idea, etiquetes i àrea a la que pertany.

S'haurà de poder accedir a la fitxa d'idea i a la de moderació des de qualsevol de les llistes, i a la d'edició des de la fitxa de la pròpia idea.

En la pàgina d'edició d'idees els responsables hi tindran l'eina per canviar-les d'estat.

Moderació d'idees.

Els responsables son els usuaris que representen la figura d'administrador, dins de les seves àrees. Podran esborrar i editar qualsevol idea en qualsevol moment.

A més seran els responsables dels canvis d'estat d'aquestes.

Llistat de moderació d'idees

Aquesta llista haurà de permetre als responsables tenir una visió global de les idees de les seves àrees per facilitar-ne la moderació.

Hi haurà de ser present el títol de la idea, el nombre de comentaris, el nombre de vots, l'increment de vots en els últims dies, data de creació... com qualsevol altre dada que pugui ser significativa.

Votacions

Les votacions s'hauran de fer en segon pla, per tal de millorar l'experiència de l'usuari evitant recarregar la pàgina. Per aconseguir-ho s'utilitzarà AJAX (Asinchronous Javascript and XML).

Serà possible votar tant a qualsevol idea que aparegui en les llistes d'idees de la pàgina principal com en la fitxa d'una idea en concret.

El nombre de vots restants serà visible en totes les pàgines on es pugui votar una idea.

Subscripcions

Els usuaris podran subscriure's i eliminar la subscripció a una idea mitjançant un botó en la fitxa de l'idea. Mentre que les subscripcions a una àrea es realitzaran en la corresponent llista d'idees de l'àrea en qüestió.

Totes les subscripcions s'hauran de fer en segon pla, per tal de millorar l'experiència de l'usuari evitant recarregar la pàgina. Per aconseguir-ho s'utilitzarà AJAX.

Requisits no funcionals generals

L'aplicació haurà d'estar integrada en l'actual Keep In Tonic, mantenint-ne l'aparença i compartint-ne el domini. De la mateixa manera que també ho te Keep In Tonic, l'aplicació d'idees haurà de seguir utilitzant *Friendly URLs.* S'haurà d'intentar minimitzar les paraules reservades que provoquin les noves adreces de l'aplicació d'idees.

L'aplicació haurà d'estar implementada en PHP 5 orientat a objecte i utilitzar una base de dades MySQL, a més a més haurà de ser una implementació eficaç en el tractament de recursos i escalable, tot aplicant-hi els coneixements adquirits durant el curs realitzat en l'empresa i seguint una arquitectura MVC(Model Vista Controlador).

Totes les dades numèriques, com ara el període de edició de les idees, el nombre d'idees que apareixen en les llistes, el temps en que es compte l'increment de vots... hauran de ser completament configurables i estar definides en un arxiu de configuració ".ini".

La sensació de l'usuari ha de ser de que està utilitzant una eina més de Keep In Tonic, i per tant l'aplicació d'idees haurà de mantenir el màxim possible l'aspecte visual de l'aplicació actual.

2.3 Mètodes de desenvolupament

Per tal de realitzar el projecte es van tenir en compte dos models diferents de desenvolupament de software, el model seqüencial i el model iteratiu.

El model seqüencial, o en cascada, es basa en ordenar rigorosament les etapes del cicle de vida del software de tal manera que una etapa no pot iniciar-se fins que acabi l'anterior. En la figura 2.1 tenim l'esquema d'aquest mètode.

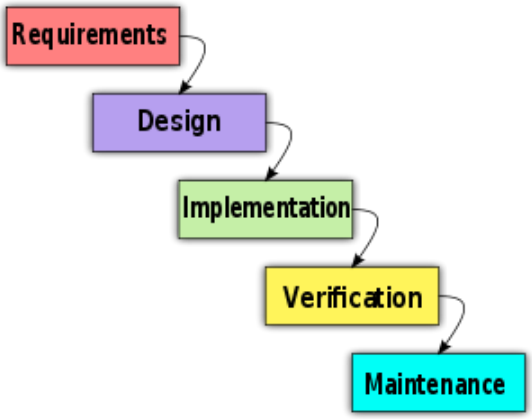


Figura 2.1: model seqüencial

La principal avantatge d'aquest model és la seva senzillesa i la distinció entre etapes que en realitza. Al ser un model que no permet la modificació dels requisits, redueix el possible error en el temps estimat de desenvolupament del projecte. D'altra banda és difícil assegurar en l'inici d'un projecte que es disposa de tota la informació necessària i que no hi haurà cap canvi en els requisits del projecte. En projectes no massa grans és el model més eficaç i econòmic.

El model iteratiu es basa en separar el desenvolupament per etapes, anomenades iteracions. Cadascuna d'aquestes iteracions inclourà etapes d'anàlisi, disseny implementació i test. En la figura 2.2 hi ha l'esquema orientatiu de com és aquest model, en ell s'hi pot observar com en les primeres iteracions les fases més importants son requeriments, anàlisi i disseny, i a mesura que van passant iteracions, perden pes a favor de les d'implementació i test.

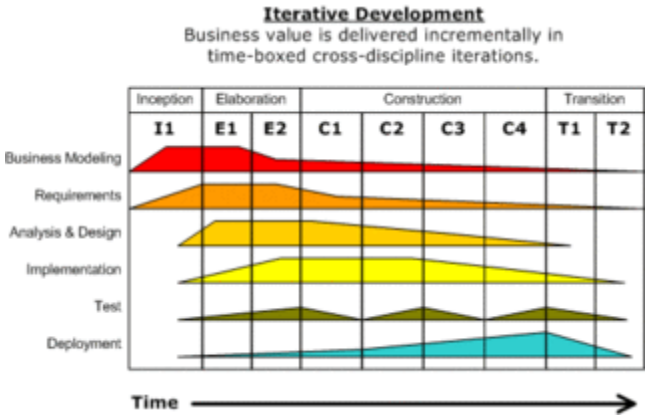


Figura 2.2: model iteratiu

Aquest model, al contrari que l'anterior és flexible a canvis en els requisits, i per tant en fa disminuir els riscos de que el producte final no sigui el que s'espera. Per altra banda el fet d'analitzar l'aplicació per etapes i poder-ne fer canvis, incrementa significativament el risc d'errada en l'estimació del temps de desenvolupament del projecte respecte el previst inicialment.

Tenint en compte els avantatges i desavantatges que comporten cadascun, finalment s'ha decidit seguir el model seqüencial, degut al menor risc d'allargar el temps de desenvolupament del projecte. Un altre factor determinant ha estat la separació clara entre fases, que al ser una sola persona la que treballarà en el projecte, en facilita la feina. Això implicarà haver de fer una fase d'anàlisi i de disseny més profunda per intentar evitar problemes durant el desenvolupament.

2.4 Eines de desenvolupament

Els llenguatges de programació que s'utilitzaran en el projecte són PHP, MySQL i Javascript mitjançant la llibreria jQuery. S'utilitzaran aquests degut a que són els que s'utilitzen a l'empresa i en els que ja està programada l'aplicació actual.

El servidor en que s'allotjarà l'aplicació final en l'empresa no en tenim cap control, i per tant, el servidor que s'escolleixi serà utilitzat només en el desenvolupament de l'aplicació, s'optarà per Apache degut a la fàcil posada en marxa, gratuïtat i a la seguretat que aporta que sigui un projecte de llarg recorregut amb una gran comunitat al darrere.

Concretament i per motius de rapidesa en la configuració i posada en marxa de l'entorn de desenvolupament de l'aplicació s'instal·larà XAMPP (Cross platform, Apache, MySQL, PHP, Perl). La versió de XAMPP que s'utilitzarà és la versió 1.7.1, aquesta versió inclou Apache HTTP Server 2.2, PHP versió 5.2.9, MySQL 5.1.33, phpMyAdmin 3.1.3.1.

A l'hora d'escollir un editor per fer el projecte, es van plantejar dues opcions, NetBeans i Eclipse PDT. Un editor de text pla estava descartat, ja que no ressalten sintaxi, no completen codi... i aquestes son funcionalitats que si que tenen els IDE i que faciliten en gran mesura la programació.

Ambdós programes permeten moltes opcions, les diferencies més grans entre ells es redueixen bàsicament, encara que no exclusivament, a:

Eclipse PDT té un nivell més baix d'auto completat, no suportant auto completat per jQuery ni per a SQL mentre que NetBeans si que ho permet.

L'auto completat de NetBeans no suporta PHP4, mentre que Eclipse si que ho fa.

A priori NetBeans sembla ser un editor més complet, però al ser Keep In Tonic un projecte amb codi antic, s'utilitzarà Eclipse PDT per facilitar l'interacció amb pàgines que utilitzin codi antic.

2.5 Riscs

Existeixen varis riscs a tenir en compte en aquest projecte:

- Planificació temporal optimista: és un risc amb una probabilitat alta, però que en aquest cas, a no ser que s'allargui en excés no comportarà conseqüències greus.
- Canvi de requisits: aquest risc te una probabilitat força probable també i provocaria que s'allargués el projecte, igual que en el cas anterior a priori no seria massa greu.
- Dificultat per accedir als *stakeholders*: aquest risc si ocorregués podria fer augmentar el temps del projecte en les fases d'anàlisi de requisits. És un risc difícil de que succeeixi ja que s'estarà diàriament en contacte amb un representant de l'empresa.
- Tests insuficients: aquest és un risc crític, i per tant s'haurà de fer minuciosament la fase de proves per tal d'evitar-lo.

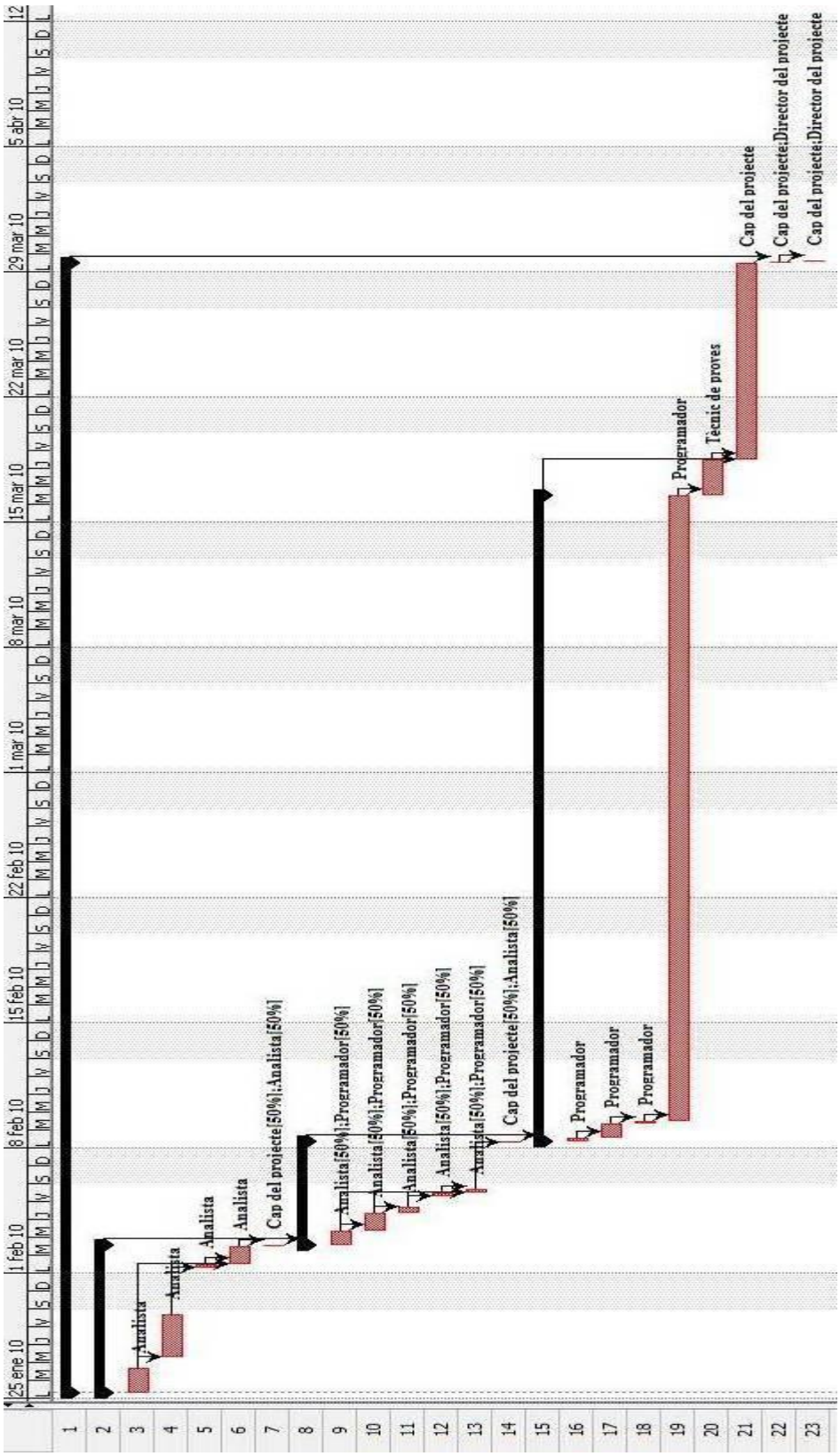
2.6 Planificació del projecte

El projecte es desenvoluparà del 25 de gener de 2010 al 29 de març de 2010 amb una dedicació de 20 dies al més i de 40 hores setmanals durant el més de febrer, i 20 hores setmanals la resta.

Tasques del projecte

	Nombre	Duración	Nombres del Recurso
1	**Gestor d'idees**	91,5 days	
2	**Anàlisi**	13,5 days	
3	Anàlisi de requisits	4 days	Analista
4	Anàlisi Keep In Tonic	6 days	Analista
5	Anàlisi de les proves	1 day	Analista
6	Documentació de l'anàlisi	2 days	Analista
7	Aprovació de l'anàlisi	0,5 days	Cap del projecte[50%];Analista[50%]
8	**Disseny**	7 days	
9	Disseny de la base de da	1 day	Analista[50%];Programador[50%]
10	Disseny del framework	2 days	Analista[50%];Programador[50%]
11	Disseny modular	1,5 days	Analista[50%];Programador[50%]
12	Disseny de l'interfície	1 day	Analista[50%];Programador[50%]
13	Documentació disseny	1 day	Analista[50%];Programador[50%]
14	Aprovació del disseny	0,5 days	Cap del projecte[50%];Analista[50%]
15	**Desenvolupament de l'**	53 days	
16	Preparació de l'entorn de	1,5 days	Programador
17	Configuració de Keep In	1 day	Programador
18	Creació i configuració de	0,5 days	Programador
19	Desenvolupament de l'ap	50 days	Programador
20	Proves	4 days	Tècnic de proves
21	Redacció de la memòria	14 days	Cap del projecte
22	Tancament del projecte	0,25 days	Cap del projecte;Director del projecte
23	Defensa del projecte	0,5 days	Cap del projecte;Director del projecte

Planificació temporal



3 Marc Teòric

3.1 Apache HTTP Server

Apache HTTP Server és un software col·laboratiu robust, ple de funcionalitats i disponible gratuïtament que permet amb poc temps tenir un servidor web funcionant en qualsevol ordinador.

Aquest software permet tenir varis *hosts* funcionant simultàniament, mitjançant *virtual hosts* configurables en l'arxiu de configuració d'apache "httpd.conf".

3.2 HTML (HyperText Markup Language)

HTML és el llenguatge de marcat utilitzat per l'elaboració de pàgines web. Aquest llenguatge no només en defineix l'estructura, sinó també la semàntica. Encara que a l'usuari se li pugui mostrar unes dades de forma idèntica utilitzant etiquetes diferents, cada etiqueta porta un concepte associat que ajuda a que usuaris no humans, puguin entendre'n el contingut i interpretar-lo degudament.

Aquest llenguatge es basa en la definició d'etiquetes, escrites entre els símbols "<" i ">". Les etiquetes poden contenir diferents atributs per distingir-los, definir-ne el comportament, donar-li cert format... A més també poden contenir text i altres etiquetes, aquestes etiquetes per tal de saber on comencen i on acaben s'han d'escriure utilitzant una etiqueta d'obertura i una de tancament, quedant de la següent manera "<etiqueta>text</etiqueta>".

Editors

Aquest llenguatge pot ser creat i editat en qualsevol editor de textos bàsic, com per exemple "bloc de notes" en Windows, "gedit" en Linux o qualsevol altre editor que admeti text sense format.

A part, existeixen altres editors més complexes per la creació de pàgines web amb característiques WYSIWYG (What You See Is What You Get). Aquest tipus d'editors permeten veure el resultat del que s'està editant en temps real a mesura que es va editant el document. Alguns exemples d'aquests editors serien: Macromedia Dreamweaver o Microsoft FrontPage. El principal problema d'aquests és que no acostumen a tenir en compte la semàntica web ni tampoc respecten la separació entre el contingut i la presentació.

Un altre tipus d'editors són els WYSIWYM (What You See Is What You Mean). Aquests editors estan pensats per separar l'estructura de la presentació, i tenir en

compte la semàntica web per d'aquesta manera solucionar el problema dels anteriors. El problema que tenen aquests editors és que s'ha de tenir clara des d'un inici l'estructura que tindrà el document, i sovint tenen un llenguatge propi. Aquests problemes fan que requereixin un període d'adaptació i que a vegades resulti difícil d'utilitzar.

La importància del marcat HTML en el posicionament en els buscadors

En els punts anteriors s'ha remarcat la importància d'una bona estructura HTML. Encara que els buscadors no facilitin dades sobre el procés de registre dels llocs web, el que si que és clar es que en més del 90 per cent dels casos aquest s'efectua de forma automàtica. Els encarregats d'aquesta tasca son programes anomenats *spiders.* Aquests programes son cecs a tot el contingut visual de la nostra pàgina (imatges, vídeos, Flash...), per tant en el que es fixen és en el contingut HTML i aquesta és una de les coses que tindrà en compte a l'hora d'indexar una pàgina web.

3.3 CSS (Cascading Style Sheets)

CSS és un llenguatge utilitzat per definir la presentació d'una pàgina web. L'objectiu d'aquests llenguatge és la separació entre l'estructura i significat d'una pàgina web, de la seva aparença, fent així possible visualitzar una mateixa pàgina de diferents maneres, sense tenir-ne que modificar el codi HTML. Aquesta separació entre el codi HTML i la presentació de la pàgina millora l'eficiència a l'hora de modificar qualsevol de les dues coses, ja que el codi està més ordenat.

3.4 jQuery

jQuery és una llibreria de JavaScript ràpida i concisa que simplifica el tractament d'elements HTML, tractament d'events, animació i interaccions AJAX. jQuery no ofereix cap funcionalitat extra que no es pogués fer únicament amb JavaScript, però en simplifica molt l'ús. Tant és així que per exemple, una acció tant recurrent com pot ser una petició AJAX i posar el contingut en un element HTML, que en JavaScript ocupa unes 10-15 línies com a mínim, en jQuery en pot resoldre en una sola línia utilitzant només dues funcions.

Una de les principals característiques que permet aquesta simplificació del codi és que qualsevol funció sempre en retorna l'objecte per d'aquesta manera poder encadenar varies funcions que es vulguin aplicar a un mateix element una rera l'altre.

3.5 MySQL

MySQL és una base de dades relacional amb llicencia GNU que s'executa com a un servidor *multithread* i multi-usuari.

Una de les principals característiques de MySQL és que et permet escollir entre varis sistemes d'emmagatzemament, podent coexistir en una mateixa base de dades taules de més d'un sistema.

Els sistemes d'emmagatzemament són: MyISAM, InnoDB, Memory, Merge i Federated entre altres.

Per poder escollir adequadament el sistema a utilitzar cal conèixer les principals característiques que puguin afectar al rendiment.

MyISAM

- És el sistema que té el maneig d'auto increments més flexible i eficaç de tots els sistemes d'emmagatzemament.
- Suporta índexs FULL TEXT.
- Múltiples sentencies SQL poden llegir simultàniament d'una taula, però al escriure cadascuna bloqueja la taula sencera mentre dura l'acció.
- No suporta transaccions.

Merge

- Una taula MERGE es una col·lecció de taules MyISAM idènticament estructurades.
- Lògicament, una consulta en una taula MERGE actua com una consulta en cadascuna de les taules MyISAM que la formen.
- Són més lentes llegint índexs, ja que es tenen que buscar índexs de múltiples taules.

InnoDB

- Suporta transaccions.
- Proporciona un sistema d'auto recuperació després d'una fallada del servidor MySQL o del host on s'està executant.
- Múltiples sentencies SQL poden llegir simultàniament d'una taula, al escriure cadascuna bloqueja només la fila afectada mentre dura l'acció.

- Suporta claus externes i integritat referencial, incloent events en cascada, eliminació i actualització.

Memory

- Té un rendiment molt ràpid.
- Els continguts de les taules Memory no perduren després d'una fallada o del reinici del sistema. Si que ho fa l'estructura.
- Al utilitzar memòria, no són adequades per grans continguts.
- No poden contenir camps de tipus TEXT ni BLOB.

Federated

Aquest sistema d'emmagatzemament és un sistema nou, inclòs en la versió 5 de MySQL. Aquest sistema permet a MySQL server utilitzar taules d'altres servidors MySQL per ser utilitzades com si fossin del propi servidor. Una de les grans avantatges que aporta és que en una sola consulta pots accedir a la informació de varis servidors, però al ser un sistema nou, encara no està del tot ben optimitzat.

- No suporta transaccions.
- No s'efectua cap mena de blocatge a les taules.

3.6 PHP (PHP: Hypertext Processor)

Història

PHP és un llenguatge d'*scripting*. La majoria de la seva sintaxi està treta d'altres llenguatges com C, Java i Perl. L'objectiu d'aquest llenguatge de programació és permetre als desenvolupadors web codificar ràpidament pàgines generades dinàmicament.

La primera versió de PHP va néixer l'any 1995, sota el nom de PHP/FI. Inicialment era un recull d'*scripts* de Perl, amb la finalitat de controlar els accessos a una web personal. L'autor d'aquest codi, que va anar augmentant la col·lecció de codi amb implementacions de C per connectar a base de dades per exemple, va fer públic el codi font perquè tothom el pogués utilitzar. Probablement aquest naixement tant personal i enfocat a una finalitat concreta sigui la que va marcar-ne el seu objectiu de facilitar la codificació de pàgines web, així com la seva poca consistència, en els inicis, com a llenguatge de programació.

L'any 1998 va aparèixer la versió 3, anomenada ja com a PHP, el seu punt més fort era la gran quantitat de compatibilitats amb bases de dades, protocols i API's,

el que va provocar que molta gent s'apuntés a crear mòduls per a PHP. L'aparició de la versió 4 va suposar importants canvis. L'objectiu d'aquesta versió va ser millorar el rendiment de les aplicacions complexes i millorar-ne la modularitat del codi de PHP, ja que fins llavors no estava preparat per suportar extensions externes de manera eficaç. PHP5 va significar un canvi considerable al introduir un nou model de programació orientada a objectes, més estricte i eficaç tot mantenint el màxim possible la compatibilitat amb la versió anterior.

Perquè PHP

No és estrany llegir opinions negatives que diuen que PHP és un llenguatge poc rigorós, per inexperts... PHP des del seu naixement va ser creat per facilitar la programació de pàgines web dinàmiques, mentre que altres llenguatges com ara Java poden esdevenir molt feixucs d'aprendre per algú que mai no hagi programat, PHP és just el contrari, resulta molt senzill començar-lo a utilitzar a un nivell bàsic, només cal anar inserint etiquetes PHP que continguin petits *scripts* enmig d'una pàgina HTML. A més, per aquest perfil de gent, PHP té un altre gran avantatge, i és que no cal declarar variables prèviament a utilitzar-les, ni definir-los cap tipus de dades, tot i que internament PHP adjudica un tipus de dades, que pot variar, a totes elles.

Aquest tipus de pàgines, a més a més, acostumen a tenir totes el gran inconvenient de no tenir nitidesa del codi, i de no ser escalables. PHP, però, permet estructures molt més complexes que no pas aquest tipus de pàgines, com per exemple des de la versió 5 una efectiva programació orientada a objectes que entre altres coses permet aplicar correctament el patró de disseny MVC (Model Vista Controlador). Aquest patró de disseny simplifica el desenvolupament i el manteniment del software al separar l'aplicació en tres components lògics:

- **Model:** aquesta capa és l'encarregada de la lògica de negoci i l'accés a les dades emmagatzemades. En definitiva és un conjunt de mètodes encapsulats en classes segons la seva naturalesa que en permeten la reutilització en qualsevol punt de l'aplicació.
- **Vista**: aquesta capa, en aplicacions web, seria el que es considera com a disseny web, és a dir, l'encarregada de la sortida de l'aplicació, tot el que visualitzarà l'usuari.
- **Controlador**: aquesta és la capa principal, aglutina el flux d'execució del programa. És l'encarregada de recollir les dades d'usuari, comunicar-se amb els models i finalment donar la informació a la vista perquè la mostri a l'usuari.

Framework

Els *frameworks* són un conjunt d'eines amb la finalitat de disminuir la càrrega dels desenvolupadors al aportar tot un seguit d'eines utilitzables en qualsevol tipus de projecte i independentment de la seva mida. Les primeres versions públiques i les més utilitzades de frameworks en PHP van ser CakePHP, Symfony i Zend Framework les quals van aparèixer l'any 2005.

A part de proporcionar als desenvolupadors eines per facilitar la programació d'una manera modular i configurable amb eines de filtratge, registre d'errors, arxius de configuració... també ajudar al programador a separar les diferents parts del codi en les tres capes corresponents a una arquitectura MVC i a que aquestes estiguin ben organitzades o a gestionar les direccions de les diferents pàgines de l'aplicació.

4 Disseny

4.1 UML: Casos d'ús

Introducció

El propòsit d'aquest document és definir les funcionalitats del sistema (casos d'ús) i el context d'aquest (interacció amb entitats externes o actors). Es presenten els diferents diagrames de casos d'ús per als actors en què es divideix el sistema i les descripcions resumides de cada actor i cas d'ús.

El model de casos d'ús que es presenta dividit en 6 parts i els respectius diagrames casos d'us.

Actors:

- **Usuari identificat:** Aquest actor representa un usuari identificat en l'aplicació, els actors: usuari autor i responsable són també actors d'aquest tipus.
- **Usuari autor:** Representa un usuari actuant amb idees i comentaris creats per ell.
- **Responsable:** Aquest actor representa un usuari identificat, responsable d'una o més àrees.
- **Servidor:** Representa aquelles accions iniciades pel propi sistema sense intervenció humana.

Casos d'ús:

- **Afegir una idea**

  Consisteix en afegir una nova idea, amb el seu títol, contingut, una àrea a la qual estarà associada i una llista de etiquetes opcionalment.

- **Afegir un comentari**

  Quan l'usuari estigui observant una idea, podrà crear un comentari associat a aquella idea. També al canviar una idea d'estat es generarà un comentari que no es podrà borrar.

- **Afegir un worklog**

  Algunes accions crearan un worklog en l'aplicació de Keep In Tonic.

- **Visualitzar llistes d'idees**

Aquest serà un dels casos d'ús més recurrents, les llistes d'idees mostraran el títol, contingut amb un nombre limitat de caràcters, data de creació, numero de comentaris, àrea, estat, foto de l'autor, etiquetes i nombre de vots. Aquest cas d'ús aglutina les llistes d'idees en estat 'new', 'implemented','in process ', la llista d'un usuari, d'un departament, d'una àrea i d'una etiqueta.

- **Tornar a la llista anterior**

Quan l'usuari estigui en una pàgina de l'aplicació que no sigui d'una llista, podrà tornar a la llista des de la qual havia anat a aquesta pàgina, mantenint els possibles filtres i la pàgina en la que es trobava.

- **Filtrar una llista**

Quan l'usuari estigui a una llista, la podrà filtrar per àrea i/o estat.

- **Visualitzar una idea**

Aquest cas d'ús correspon a la visualització completa d'una idea. Seran visibles totes les propietats relacionades amb l'idea excepte els etiquetes.

- **Visualitzar comentaris**

L'usuari podrà directament anar a veure els comentaris d'una idea des de la llista d'idees.

- **Votar una idea**

L'usuari podrà votar una idea sempre que l'estat de la idea ho permeti, que l'usuari tingui vots restants i que encara no l'hagi votat. Aquest cas d'ús pot succeir tant en la pàgina de la idea com en qualsevol de les llistes, excepte la de moderació.

- **Esborrar una idea**

L'usuari podrà esborrar les seves idees durant un període de temps, o si és responsable podrà esborrar qualsevol idea que pertanyi a alguna de les seves àrees i en qualsevol moment.

- **Esborrar un comentari**

Un usuari podrà esborrar qualsevol comentari escrit per ell mateix, o qualsevol comentari esborrable de qualsevol idea pertanyent a les àrees de les quals sigui responsable.

- **Editar una idea**

Aquest cas d'ús correspon a l'edició d'idees, es podrà editar títol, contingut, etiquetes, l'àrea de la idea i afegir o editar l'identificador de la tasca de JIRA associada a la idea.

- **Canviar estat d'una idea**

Un usuari responsable podrà modificar l'estat d'una idea en qualsevol moment.

- **Visualitzar llista de moderació**

Els usuaris que siguin responsables tindran accés a la llista de moderació des de la qual se'ls facilitarà la gestió de les idees permetent una visió ràpida de les idees de les seves àrees de responsabilitat.

- **Ordenar una llista**

Els responsables podran ordenar la llista de moderació per qualsevol de les columnes existents en ordre tant ascendent com descendent.

- **Subscripció a una idea**

Els usuaris podran subscriure's a una idea per tal de rebre notificacions dels canvis que succeeixin, com afegir comentaris o canviar l'estat de la idea.

- **Subscripció a una àrea**

Els usuaris podran subscriure's a una àrea per tal de rebre notificacions dels canvis que succeeixin, com afegir noves idea a l'àrea o canviar l'estat d'alguna de les idees d'aquesta.

- **Enviar correu**

Aquest cas d'ús correspon a l'enviament de correu electrònic, aquest succeirà tant per accions realitzades pels usuaris com periòdicament per part del servidor.

Vistes

A continuació es mostren les diferents vistes funcionals del sistema.

1. Keep In Tonic

En la figura 4.1 veiem el diagrama de casos d'ús referents a les accions de la nova aplicació que es podran realitzar des de les eines antigues de Keep In Tonic pels usuaris de l'aplicació d'idees.

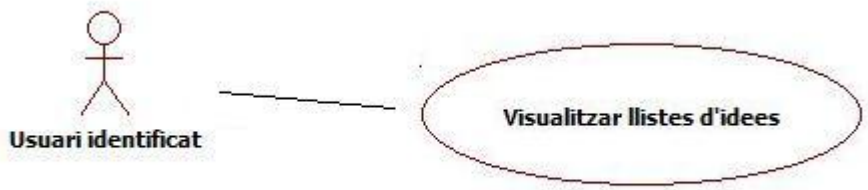


Figura 4.1: Diagrama UML de les pàgines de KiT

2. Pàgina principal de l'aplicació d'idees

En la figura 4.2 tenim el diagrama de casos d'ús de la pàgina principal de l'aplicació d'idees.

Figura 4.2: Diagrama UML de la pàgina principal d'idees

3. Fitxa d'idea

En les figures 4.3, 4.4 i 4.5 tenim el diagrama de casos d'ús relatius a les pàgines de fitxes d'una idea.

Figura 4.3: Diagrama UML de la fitxa d'idea

Figura 4.4: Diagrama UML de la fitxa d'idea

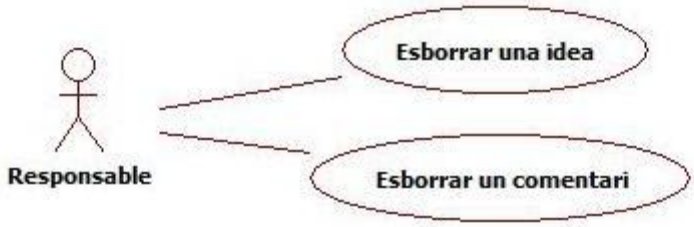


Figura 4.5: Diagrama UML de la fitxa d'idea

4. Fitxa d'edició d'idea

En les figures 4.6 i 4.7 hi ha el diagrama de casos d'ús referents a la pàgina d'edició d'idees.

Figura 4.6: Diagrama UML de la fitxa d'edició d'idea

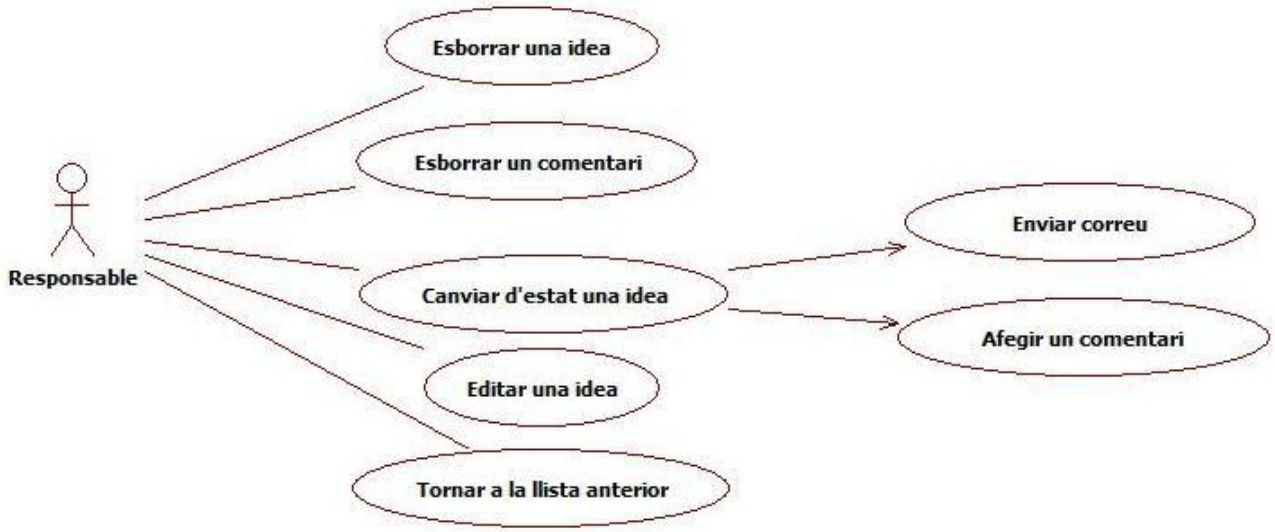


Figura 4.7: Diagrama UML de la fitxa d'edició d'idea

5. Apartat de moderació d'idees

En la figura 4.8 hi ha el diagrama de casos d'ús referents a la pàgina de moderació d'idees.

Figura 4.8: Diagrama UML de la pàgina de moderació

6. Altres casos d'us

En la figura 4.9 hi ha representat el diagrama de casos d'ús que no estan relacionats directament amb cap pàgina.

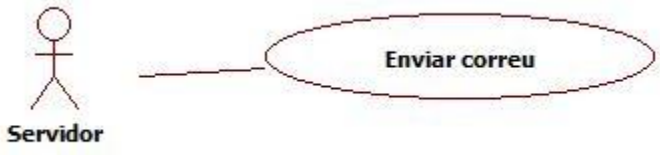


Figura 4.9: Diagrama UML no encapsulable en cap pàgina

4.2 Arquitectura del sistema

Al ser aquest un projecte que s'ha d'integrar en una altre eina, s'ha de fer un primer anàlisi de l'estructura de l'aplicació actual i un estudi dels elements d'aquesta.

4.2.1 Anàlisi de Keep In Tonic

Keep In Tonic, com ja s'ha esmentat anteriorment, utilitza un sistema d'adreces anomenat Friendly URLs (Uniform Resource Locator) que consisteix en que la

direcció de les diferents pàgines no descriu les direccions reals de l'arxiu a executar de l'aplicació, sinó que existeix un objecte que tracta aquesta direcció i n'extreu la informació que serà utilitzada per carregar el controlador adequat i per utilitzar-ne els paràmetres en aquest. Cal tenir en compte a més que un gestor d'URLs no només serveix per aconseguir el nom del controlador a executar, sinó que sovint provoca que hi hagi paraules reservades en alguna posició de la URL.

Per tal d'aconseguir això s'han de redirigir totes les peticions HTTP a un mateix fitxer PHP i aquest serà l'encarregat d'inicialitzar els elements de configuració necessaris, executar el gestor d'URLs i seguidament el controlador adequat mitjançant un objecte nomenat *dispatcher.*

Cal també identificar qualsevol element en el flux principal de l'aplicació que pugui afectar a l'execució de l'aplicació.

inc/index.php

Aquest és el primer fitxer que s'executa per a qualsevol petició. La seqüència d'accions en aquest fitxer són:

1. Aplica la funció strip_tags a les variables globals $_GET, $_POST, $_REQUEST i $_SESSION.
2. Inicialitza la sessió.
3. Carrega el fitxer de configuració inc/config.lib.php.
4. Inicialitza l'objecte sql implementant el patró de disseny *multiton*.
5. Inicialitza i executa el gestor d'URLs.
6. Inicialitza l'objecte *singleton* user.
7. Si l'usuari no està identificat el redirecciona a la pàgina corresponent.
8. Executa el *dispatcher*.
9. Executa un seguit d'operacions per depurar l'aplicació.

inc/config.lib.php

En aquest arxiu es defineixen constants referents a URLs, com per exemple el domini de l'aplicació i les rutes de les diferents carpetes de l'aplicació. També fa una inclusió de diferents classes pròpies i externes. A més executa en cas de que així sigui requerit, tot un seguit de funcions relacionades amb la depuració de l'aplicació.

inc/misc.lib.php

Aquest arxiu conté una llibreria de funcions vàries com creació d'URLs, obtenir la IP de l'usuari, donar format a certs tipus de dades...

Gestor d'URLs

Keep In Tonic té una classe nomenada URLHandler que s'encarrega d'aquesta funció. Aquesta classe consta de dos mètodes i un constructor.

En el constructor agafa tot de paràmetres opcionals, globals en l'aplicació, i els guarda en una propietat pública.

El mètode detectMainSections, cridat en el constructor, defineix en funció del primer paràmetre de la URL quina és la secció que li correspon, definint en cas de que sigui necessari variables extra. La llista de noms de secció utilitzats, i que per tant són paraules reservades en primera posició de les direccions, és:

- worklog
- upload
- worklog-edit
- tweettonic
- login
- logout
- search
- avatar
- team
- team-photos
- account
- get-avatar
- ajax
- restricted
- crons
- external
- index.xml

El mètode "prepare" inicialitza la propietat de classe "main_controller" amb la ruta de l'arxiu que conté el *dispatcher.* I afegeix a la variable global $_GET els paràmetres obtinguts de l'adreça.

Amb aquesta informació sabem que les adreces utilitzades segueixen el següent format, estant entre claus les parts opcionals:

http://domini/secció[/paràmetres][/valor_tag/tag][/offtopic|/work][/hotness][/page/num_pàgina]

Dispatcher

El *dispatcher* és l'element encarregat d'executar el controlador adequat. En el cas de Keep InTonic ens trobem amb un arxiu que inicialitza varis objectes Template, un *switch* amb cadascuna de les diferents seccions i l'arxiu que té que carregar. Finalment imprimeix la plantilla per pantalla.

sql

Aquesta es una classe que permet tenir connexions simultànies a diferents bases de dades, però només una a cadascuna d'elles. Les dades de connexió les obté de l'arxiu "inc/profiles.lib.php"

Té com a mètode principal "query" que permet fer consultes a la base de dades pertinent, en cas de ser una consulta d'obtenció de dades, en retorna els resultats en format array.

user

Aquesta classe és de tipus *singleton* i conté tots els mètodes relacionats amb usuari: identificar-se, obtenir dades, subscriure's, obtenir llistes d'usuari, editar dades d'usuari, buscar usuaris...

Al inicialitzar-se guarda en propietats de l'objecte les dades de l'usuari connectat.

template

Aquesta classe es la que fa les funcions de sistema de plantilles de l'aplicació. A diferència d'altres sistemes de plantilles, com ara smarty, aquesta no té un llenguatge propi, sinó que utilitza el propi llenguatge PHP per representar les operacions que s'hagin de fer en les plantilles.

Consta de 3 mètodes:

- Constructor, en el qual se li defineix la plantilla a utilitzar.
- set, que és el que permet definir variables en la plantilla.
- fetch, que és el mètode que ens retorna la sortida resultant d'executar la plantilla amb les variables definides mitjançant el mètode set.

PHPMailer

PHPMailer és una classe de PHP per a la transferència de correus electrònics amb una gran varietat de funcions i característiques. Entre altres coses permet:

- Enviar correus a múltiples recipients.
- Utilitzar servidors i autentificació SMTP.
- Correus amb part alternativa pels clients que no permetin HTML.
- Diferents tipus de codificació
- Capçaleres personalitzables.
- Múltiples arxius adjunts.
- Suport per imatges incrustades.

item

Aquesta classe conté un conjunt de mètodes relacionats amb worklogs i "tweets" (elements de twitter). Afegir, editar, eliminar...

4.2.2 Framework de l'aplicació d'idees

Per tal d'aprofitar els beneficis de la programació utilitzant un *framework,* se n'ha creat un a mida per a l'aplicació. Els elements d'aquest *framework* són:

__autoload()

Per tal de no haver-se que preocupar constantment d'incloure els fitxers necessaris, o de incloure'ls tots, cosa que sobrecarregaria el programa amb molts elements innecessaris, s'utilitzarà la funció de PHP __autoload()

Aquesta és una funció global de PHP, que es cridada sempre que s'intenta crear un objecte que no ha estat definit encara a l'aplicació. Com a paràmetres rep el nom de la classe que es vol instanciar. Per tant hem de definir un patró en els noms de les classes que ens permetin, en aquesta funció, conèixer on està situat l'arxiu que la conté i quin és el nom d'aquest.

En l'aplicació tindrem 3 tipus d'objectes ben diferenciats: controladors, models i les classes referents al *framework*.

- Controladors
  - Estaran situats a la carpeta scripts, agrupats en carpetes.
  - El nom dels arxius serà del tipus nom.controller.php.
  - Els nom dels objectes tindrà el següent format: NomControllerCarpeta.

- Models:
  - Estaran situats a la carpeta inc, agrupats en carpetes.
  - El nom dels arxius serà del tipus nom.model.php
  - Els nom dels objectes tindrà el següent format: NomModelCarpeta.
- Classes del *framework:*
  - Estaran situats a la carpeta inc.
  - El nom dels arxius serà nom.class.php
  - El nom dels objectes no podrà contenir ni les paraules Controller ni Model.

Config

Aquesta classe serveix per obtenir dades de diferents arxius de configuració ".ini"

Factory

Aquesta implementa els patró de disseny *factory* i no podrà instanciar-se. D'aquesta manera aconseguirem un punt únic des d'on poder instanciar qualsevol objecte, a més no es podrà instanciar dues vegades un mateix objecte a través d'aquest, de manera que si es demana un objecte que ja ha estat instanciat amb anterioritat, s'aconseguirà l'instanciat anteriorment, com si d'un objecte que implementi el patró *singleton* es tractés.

Filter

Aquesta classe l'ha proporcionat l'empresa, conté diferents tipus de mètodes de validació i filtratge, així com classes de filtre específiques per a les variables $_GET, $_POST, $_COOKIE i $_FILES. Al instanciar una d'aquestes classes específiques de filtratge, n'elimina la variable original, fent així que només sigui possible obtenir les variables que puguin contenir a través d'aquests filtres.

Mailer

Aquesta classe implementa el patró de disseny *adapter* que consisteix en crear una classe intermèdia per utilitzar una classe externa, PHPMailer en aquest cas. L'objectiu d'aplicar aquest patró de disseny és facilitar una futura actualització de la classe externa, podent adaptar fàcilment qualsevol canvi que s'hagi produït en aquesta.

Controller

Aquesta és una classe abstracta de la qual n'extendran tots els controladors de l'aplicació. La classe definirà el mètode abstracta getData que serà el mètode que s'executarà al carregar un controlador, i que per tant contindrà l'execució principal de cada controlador, i tindrà tot un seguit de mètodes comuns en la majoria de controladors com per exemple:

- Obtenir un filtre.
- Obtenir un model.
- Obtenir variables de configuració.

Model

Aquesta és una classe abstracta de la qual n'extendran tots els models de l'aplicació. La seva funció és crear una connexió a la base de dades, que serà accessible a tots els models que n'extenguin.

Excepcions

L'aplicació tindrà un sistema d'excepcions per tractar els possibles errors i/o restriccions d'accés que puguin succeir. Hi ha una classe GeneralException, la qual no tindrà cap funció específica, però que es crea amb la intenció de que tota la resta d'excepcions de l'aplicació n'extenguin i d'aquesta manera si en un futur es necessita afegir alguna funcionalitat a totes les excepcions, com per exemple guardar un registre d'errors, sigui més simple.

Les altres excepcions que existiran seran:

- ForbiddenException: defineix un error de restricció d'accés. quan sorgeix-hi una excepció d'aquestes s'haurà de retornar una capçalera HTTP 403.
- NotFoundException: Aquesta pàgina excepció mostrarà una pàgina d'error de pàgina no trobada, i utilitzarà capçaleres 404 o 500 depenent del tipus d'error que hagi succeït, 404 si s'intenta accedir a un element que no existeix, i 500 en casos estranys en que el servidor no respongui com s'espera.
- AjaxException: Aquesta pàgina servirà per retornar un element d'error en format Json. S'utilitzarà en aquells controladors que siguin cridats via AJAX.

4.2.3 Integració en l'aplicació Keep In Tonic

Un dels primers punts que cal decidir, es quin tipus d'URLs utilitzarà l'aplicació. Per tal de complir el requisit de minimitzar l'impacte de paraules reservades que pugui provocar l'aplicació d'idees, s'englobaran totes dins de la secció "idea", i utilitzant el següent paràmetre per decidir quin es el controlador a executar. D'aquesta manera, a més dels paràmetres addicionals que agafava el gestor d'URLs, les direccions de l'aplicació d'idees quedaran de la següent manera:

http://domini/idea[/seccio_idea][/paràmetres]

D'aquesta manera aconseguim que respecte a l'aplicació Keep In Tonic, només s'hagi afegit una paraula reservada. Això provoca que en en *switch* del dispatcher, s'hagi d'afegir un nou cas "idea", que serà el punt inicial de la nova aplicació.

En aquest punt d'entrada s'hi crearà un *dispatcher* propi de l'aplicació d'idees. En aquest punt es recolliran totes les diferents excepcions que es puguin llençar des de la nostre aplicació i es tractaran degudament.

En quant a la manera de carregar els controladors, s'utilitzarà un arxiu de configuració que relacionarà el paràmetre de la URL amb el controlador que té que carregar, aconseguint d'aquesta manera una gran flexibilitat a l'hora de modificar les URLs de l'aplicació.

Com el dispatcher de Keep In Tonic treu per pantalla un seguit de plantilles inicialitzades abans de l'execució dels controladors, per tal de tenir ple control de la sortida del programa en als controladors, s'ha de fer que aquests ens retornin els diferents objectes template en format de matriu, per així sobreescriure les variables del dispatcher de Keep In Tonic per les que ens retorni el controlador.

El diagrama de classes resultant un cop muntat el framework esta representat en les figures 4.10, 4.11 i 4.12:

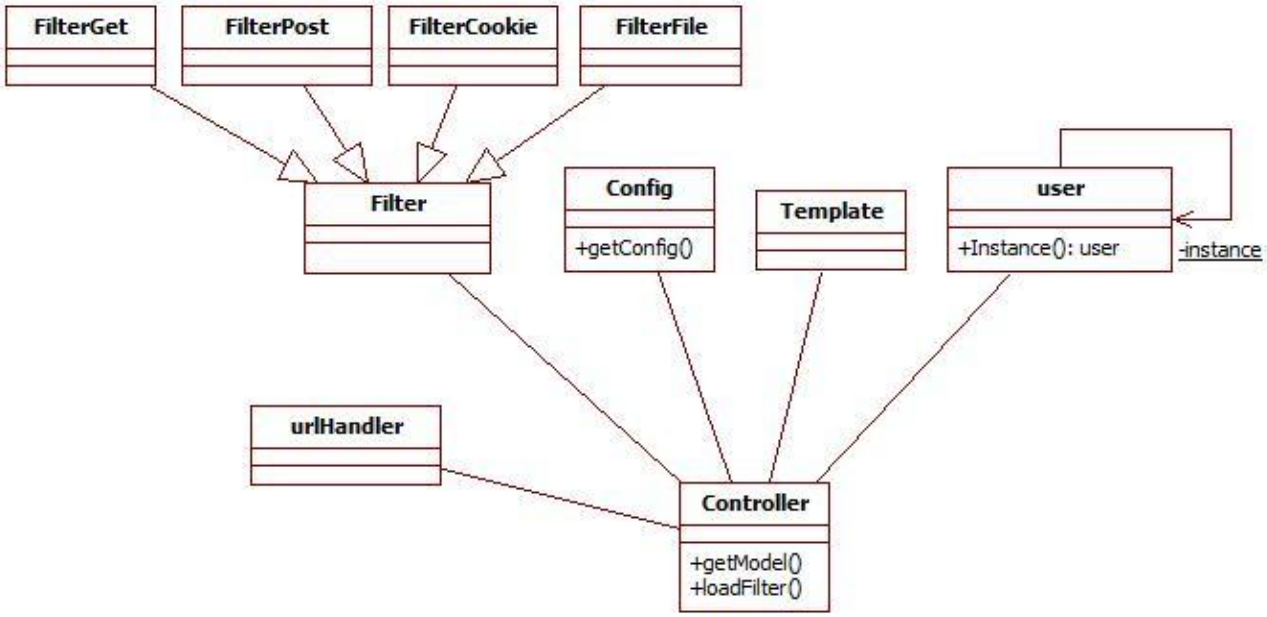


Figura 3.10: Diagrama de classes de l'aplicació

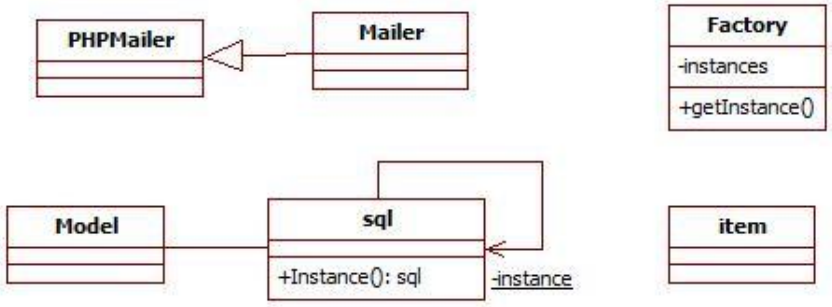


Figura 4.11: Diagrama de classes de l'aplicació

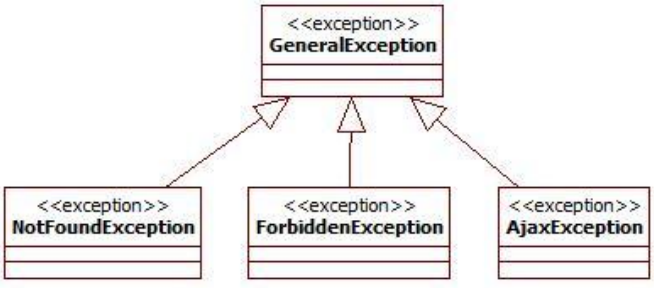


Figura 4.12: Diagrama de classes de l'aplicació

En l'aplicació s'ha de definir dos models, un per cadascun dels dos grans objectes del projecte: àrees i idees. Cadascun d'aquests models han de tenir totes les funcions relacionades amb aquests: afegir idees, comentaris, editar, obtenir llistes...

L'aplicació disposarà de 4 pàgines dinàmiques principals, l'esquema de pàgines de l'aplicació és el següent:

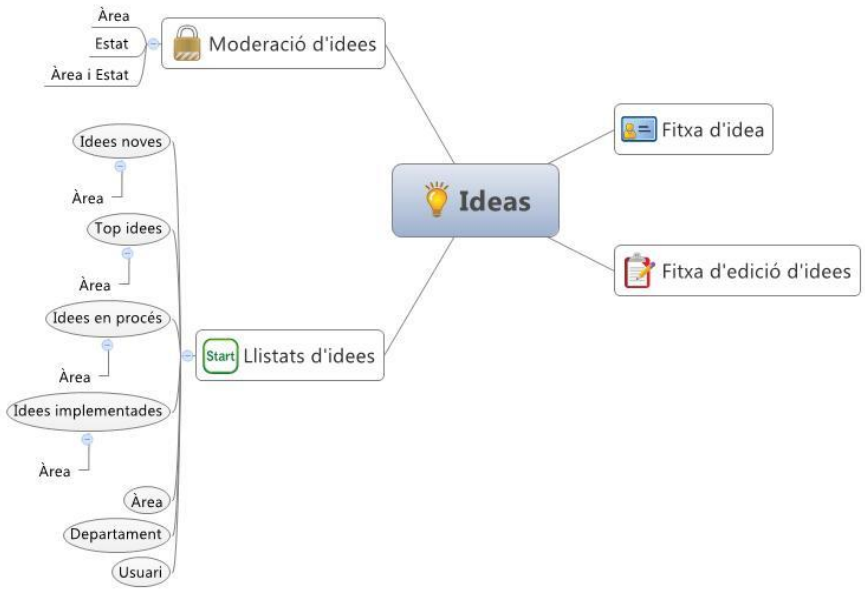


Figura 4.13: Esquema de pàgines de l'aplicació

En quant a controladors se'n defineixen 7 diferents, un per cadascuna de les 4 pàgines de l'aplicació, dos per les diferents peticions AJAX i un per gestionar l'enviament periòdic de correus electrònics. Els controladors per tant són:

- main
- edit
- idea
- manage
- subscription
- vote
- subscriptionBulletin

4.3 Base de dades

En aquest apartat, de la mateixa manera que en l'anterior, al no començar el projecte des de zero, sinó que existeix una plataforma sobre la qual s'hi ha de integrar el projecte provoca que abans de començar a dissenyar la base de dades del projecte s'hagi de fer un estudi de la base de dades de Keep In Tonic.

Base de dades de Keep In Tonic

Aquesta base de dades consta de 12 taules, totes elles utilitzant el motor d'emmagatzemament MyISAM, amb un índex full text a la taula 'users'. L'esquema de la base de dades és el representat en la figura 4.14:

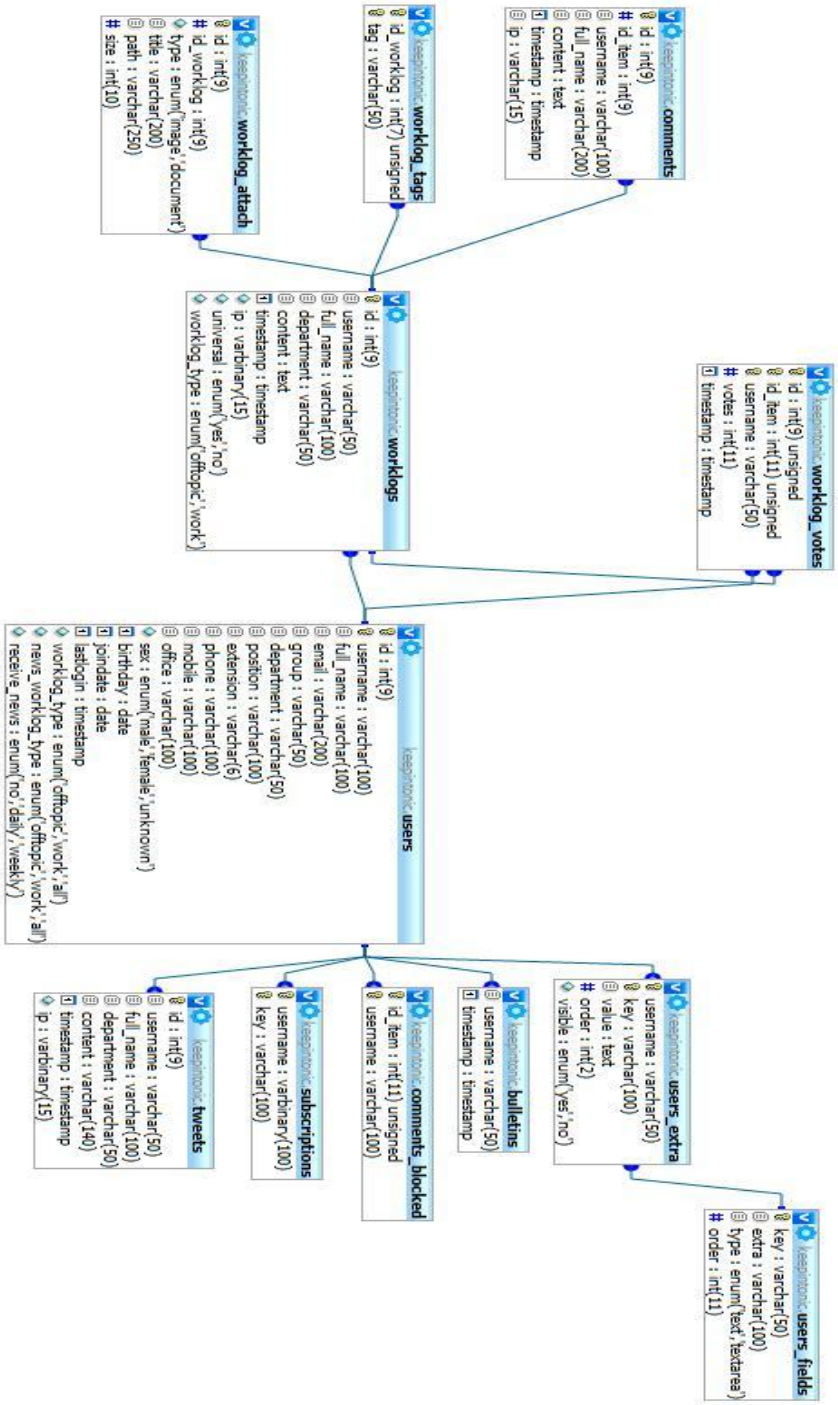


Figura 4.14: Esquema de la base de dades original

Una de les coses importants que s'hi observa és que tot i que la taula ‘users’ conté un camp 'id', que és la clau primària de la taula, s'utilitza el camp 'username' per fer d'enllaç entre les diferents taules de la base de dades, cosa que provoca que canviar el nom d'usuari comporti molts maldecaps comparant-ho amb la flexibilitat que comportaria utilitzar el camp auto incremental 'id'.

Un altre inconvenient és la utilització de diferents noms per referir-se a un mateix camp, aquest fet pot provocar errades o confusió a l'hora de programar o llegir el codi. En les taules 'worklog_attach' i 'worklog_type' es refereix al camp 'id' de la taula 'worklog' com a 'id_worklog', mentre que en les taules 'worklog_votes' i comments el nom que utilitza per referir-se a aquest mateix camp és el de 'id_item'.

Disseny de la nova base de dades

Tot i que la base de dades anterior presenta alguns inconvenients, no forma part del projecte la modificació del programa proporcionat, per tant no es modificarà l'estructura de la base de dades.

Com no hi ha previst cap buscador en la nova aplicació, el motor utilitzat per a la nova part de la base de dades serà InnoDB en totes les taules per poder aprofitar el fet de tenir un sistema de base de dades transaccional i relacional, excepte en els punts en que s'enllaci a taules antigues.

El nom de totes les taules estarà precedit per la cadena de text "idea". D'aquesta manera seran més fàcils d'identificar i intuir-ne l'ús al veure el nom.

Les taules necessàries per a la nova aplicació són les següents:

- idea
- idea_vote
- idea_comment
- idea_state
- idea_tag
- idea_tag_relation
- idea_area
- idea_area_responsible
- idea_subscription
- idea_area_subscription

El diagrama resultant de la base de dades, minimitzant les taules antigues que no presenten cap canvi o que no interactuen amb les noves, és representat en la figura 4.15.

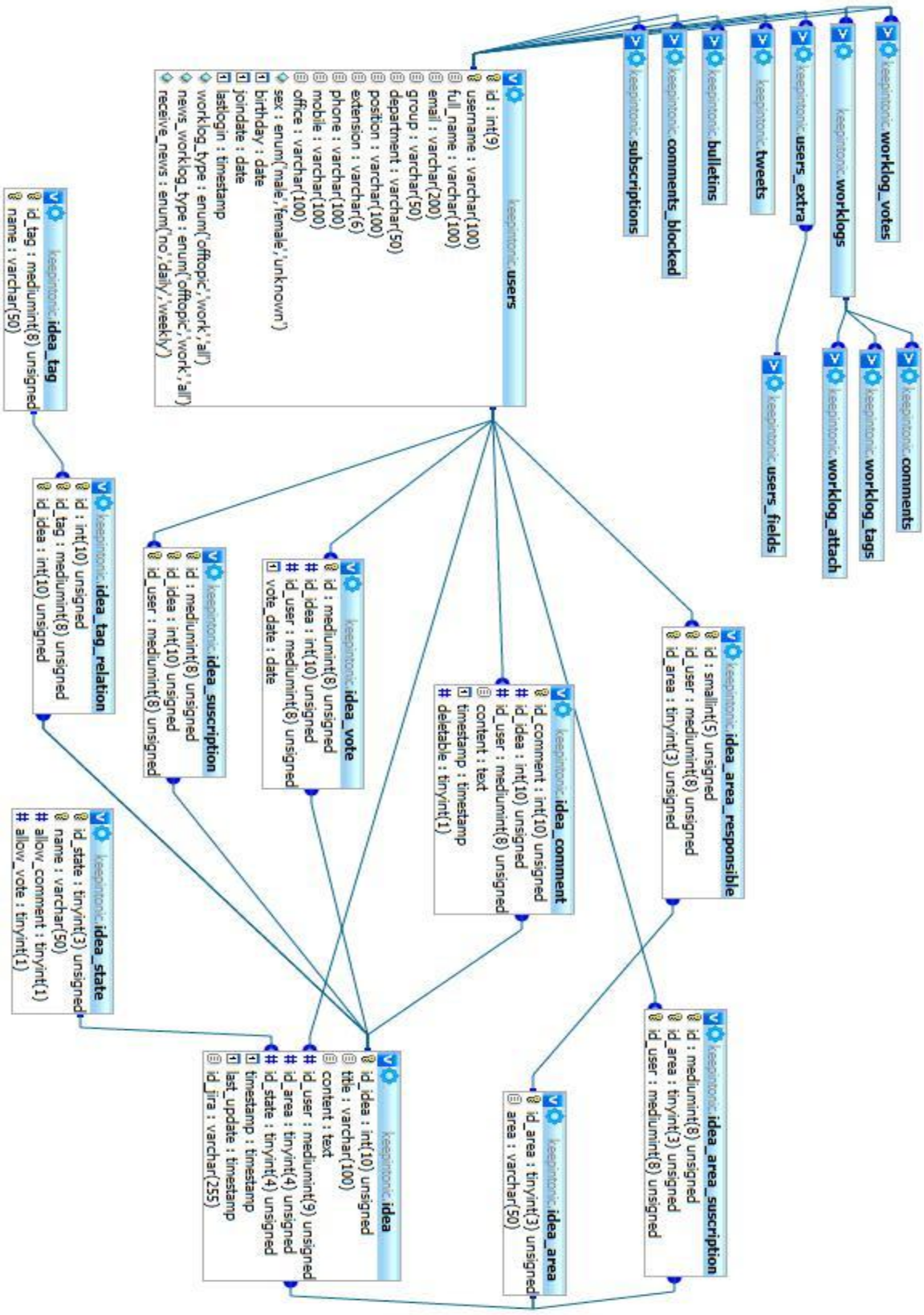


Figura 4.15: Esquema de la base de dades de l'aplicació

4.4 Interfície d'usuari

L'interfície de l'aplicació ha de seguir l'estil de Keep InTonic. Les parts comunes en les pàgines de l'aplicació que mantindrà la nova eina són:

- Un iframe a la part superior (figura 4.16).

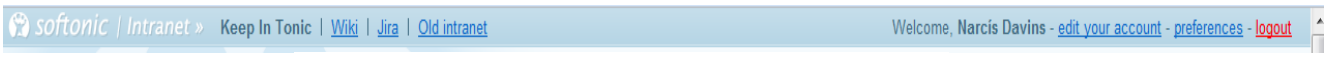


Figura 4.16: captura de pantalla de l'iframe superior

- Navegació per pestanyes entre aplicacions (figura 4.17).

Figura 4.17: captura de pantalla de les pestanyes de les aplicacions

- Navegació per pestanyes entre diferents tipus de llistes (figura 4.18).

Figura 4.18: captura de pantalla de les llistes

- Barra lateral a la part dreta de la pantalla amb enllaços i accions possibles (figura 4.19).

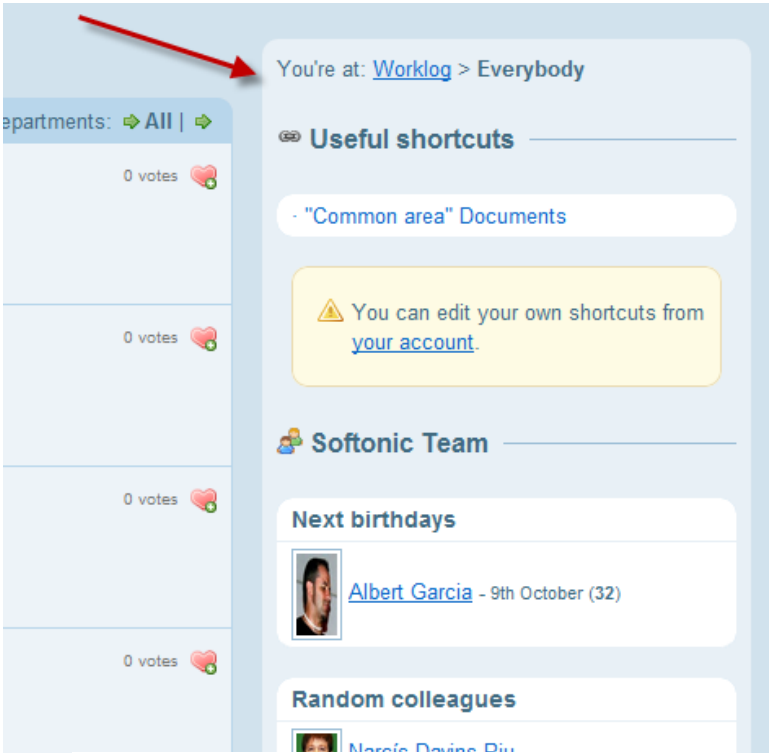


Figura 4.19: captura de pantalla de la barra lateral

- Logotip de Keep In Tonic (figura 4.20).

Figura 4.20: captura de pantalla del logotip

- Tonalitats blaves en tota l'aplicació.

També l'aplicació disposa d'una llibreria d'icones, que s'han aprofitat en diversos punts de la nova aplicació.

5 Codificació i proves

5.1 Estil de codificació

L'homogeneïtzació de l'estil de codificació és un aspecte molt important en qualsevol tipus de projecte, sobretot quan hi participen vàries persones. Entre altres avantatges, seguir un mateix estil en tot un projecte agilitza la lectura de codi, facilita la detecció d'errors i ajuda a terceres persones a entendre el codi del teu projecte.

Algunes de les directrius sobre les quals s'ha basat la codificació d'aquest projecte són les següents:

Idioma

Tant en els comentaris, com en qualsevol variable, classe o mètode, l'idioma utilitzat és l'anglès. Aquesta decisió es deu a que l'empresa per la qual s'ha desenvolupat el projecte hi ha programadors de moltes nacionalitats diferents.

Noms

Els noms de variables, mètodes i classes són el més descriptius possibles del que representen.

Els noms compostos de les classes i dels seus mètodes s'escriuen sense guió baix, les paraules es diferencien escrivint la primera lletra en majúscula, excepte la primera paraula en el cas dels mètodes, per tal de diferenciar-los del nom de les classes més fàcilment.

Les variables globals s'escriuen amb tots els caràcters en majúscula, separant les diferents paraules per guions baixos.

Claus ("{}")

En el cas de les claus hi havia dues opcions per triar, posar-les en la mateixa línia que inicia el bloc que contindrà, o fer-ho en la següent.

Les dues opcions, il·lustrades, són:

Opció 1

```
if( xxx ) {
    ...
}
```

Opció 2

```
if( xxx )
{
    ...
}
```

L'opció escollida és la segona ja que si s'utilitzen editors que detectin les claus, com Eclipse que ha estat l'utilitzat en el desenvolupament del projecte, és més fàcil identificar la clau que obre si està a l'inici de la línia que no si tens que buscar-la a finals de línia. Això es deu a que saps més precisament per on la tens que buscar que no pas en la primera opció ja que la longitud de la condició és diferent en cada cas.

Alineació

Quan s'inicialitzen varies variables consecutivament, s'alineen els "=" mitjançant tabuladors per tal de facilitar-ne la lectura.

Igualtats

Al comparar una variable a un valor, sempre es posa el valor al davant, d'aquesta manera s'evita el possible error produït al deixar-se un dels dos iguals de la igualtat, ja que al executar-ho PHP llançaria un error i per tant es detectaria i es solucionaria.

5.2 Proves

Tot i ser una aplicació per a la intranet d'una empresa, amb el control i la seguretat, a priori, de que no hi haurà usuaris malintencionats, s'ha donat la mateixa importància a que l'aplicació funcioni degudament a que ho faci de manera segura.

Les proves de que l'aplicació funciona com s'espera s'han anat fent a mesura que s'anaven creant els diferents mòduls. Aquestes proves simulaven tant les actituds esperades d'un usuari: votar una idea, editar… com altres menys corrents: votar

una mateixa idea dos cops, votar una idea inexistent, intentar editar una idea que no sigui teva sense ser-ne responsable…

A part d'aquestes proves, també s'han provat, un cop finalitzada la aplicació, diferents tipus d'atacs per veure si aquesta responia com s'esperava. Per tal de fer aquestes proves abans s'han de conèixer diferents atacs que pot rebre una web a nivell de deficiències de codi. D'aquestes proves en quedaran excloses per tant tots aquells atacs que afectin al servidor com pot ser DoS (Denial of Service), o les que puguin provenir de deficiències en l'aplicació ja existent, com robatori de sessions.

Cross Site Scripting (XSS)

Aquest atac consisteix en introduir codi de *script*, javascript per exemple, en la pàgina web. Per tal de detectar si l'aplicació té aquest problema s'han comprovat totes les possibles entrades de l'aplicació utilitzant com a entrada una cadena especial per localitzar vulnerabilitats d'aquest tipus. La cadena utilitzada en aquest cas és:

```
></script>">'><script>alert(string.fromcharcode(88,83,83))</script>
```

SQL injection

Aquest atac consisteix en introduir codi SQL en les consultes realitzades pel servidor a la base de dades. Per comprovar-ho s'han tingut que provar totes les entrades una a una.

Execució maliciosa de codi

Aquesta vulnerabilitat existeix quan l'aplicació utilitza les entrades de l'usuari per carregar arxius. En aquesta aplicació no es dona aquesta situació i per tant no és necessari fer cap prova.

Fuga d'informació

Aquest error consisteix en mostrar els errors de programació que succeeixin. Això suposa un error de seguretat ja que els errors de PHP mostren rutes d'arxius i a vegades també petites parts del codi.

6 Conclusions

6.1 Assoliment d'objectius

Aquest és un aspecte difícil de valorar degut a que els objectius d'aquest projecte són objectius a llarg termini, que només poden ser valorats un cop hagi passat un temps des de que el projecte s'hagi posat en marxa.

Tot i que encara no s'ha posat en marxa i que per tant no es poden valorar els objectius, l'aplicació compleix tots els requisits que es van demanar pensant en assolir aquests objectius. Un altre aspecte encoratjador és que hi ha diversos treballadors de l'empresa, que tot i que no han tingut encara possibilitat de provar l'aplicació, un cop se'ls ha explicat diferents aspectes de com seria aquesta, han opinat favorablement sobre la proposta i n'han mostrat interès.

Tenint en compte aquests aspectes, tot sembla indicar que els objectius que es pretenen assolir amb aquest projecte es compliran.

6.2 Desviacions sobre la planificació

En les primeres fases: anàlisi, disseny, desenvolupament i proves, van efectuar-se segons el calendari establert, amb petites variacions en les seves durades. Totes elles van ser acabades segons estava previst a mitjans del més de març.

L'ultima fase de redacció de la memòria però, ha sofert un considerable retard segons el calendari previst degut a situacions vàries, endarrerint el seu inici fins a principi de maig. La duració d'aquesta tasca ha estat reduïda en quant a nombre total d'hores, però dedicant-hi menys hores diàries de les previstes, que eren quatre.

El diagrama de GANTT de seguiment del desenvolupament del projecte ha estat el de la figura 6.1.

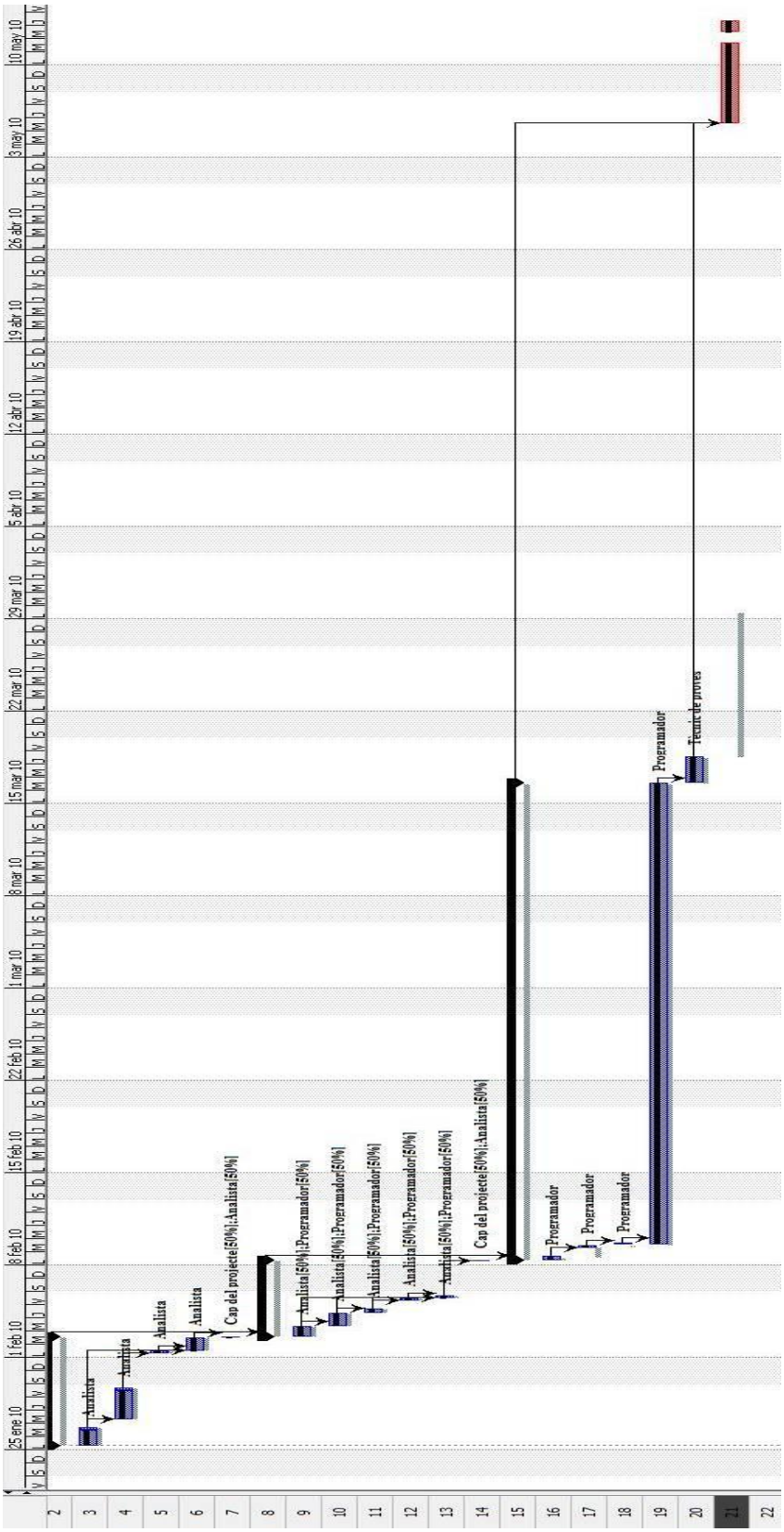

25 ene 10
1 feb 10
8 feb 10
15 feb 10
22 feb 10
1 mar 10
8 mar 10
15 mar 10
22 mar 10
29 mar 10
5 abr 10
12 abr 10
19 abr 10
26 abr 10
3 may 10
10 may 10
Analista
Analista
Analista
Cap del projecte[50%];Analista[50%]
Analista[50%];Programador[50%]
Analista[50%];Programador[50%]
Analista[50%];Programador[50%]
Analista[50%];Programador[50%]
Analista[50%];Programador[50%]
Cap del projecte[50%];Analista[50%]
Programador
Programador
Programador
Programador
Tests de proves

Figura 6.1

6.3 Línies d'ampliació

Hi ha varies línies possibles d'ampliació d'aquest projecte, algunes d'elles són:

Integrar l'aplicació amb el gestor de tasques que utilitza l'empresa: JIRA. Aquesta integració podria permetre conèixer l'estat de la tasca associada a una idea, canviar automàticament l'estat d'una idea a implementada quan la tasca associada finalitza, facilitar la creació d'una tasca des de la fitxa d'una idea...

Una altre possible línia d'ampliació important seria la creació de temes sobre els quals donar idees. Aquesta millora permetria rebre idees focalitzades en un tema en concret d'interès per a l'empresa, a més de tenir les idees pertanyents a un mateix tema agrupades i així facilitar-ne la valoració.

A part d'aquestes dos grans línies també hi hauria altres ampliacions més petites possibles, com per exemple canviar el sistema de *BBCodes* utilitzat en la creació de idees i comentaris, heretat de Keep In Tonic, per un sistema WYSIWYG, d'aquesta manera per a usuaris menys avançats seria més fàcil crear textos enriquits. Aquesta ampliació, però, requereix també modificar-ho a Keep In Tonic per tal de mantenir l'uniformitat en el conjunt d'aplicacions.

7 Bibliografia

Bibliografia fonamental

- Paul Dubois, Stefan Hinz, Carsten Pedersen, "MySQL 5.0 Certification Study Guide". MySQL Press.
- Kevin McArthur, "Pro PHP: Patterns, Frameworks, Testing and More". Apress 2008

Fonts Electròniques

Pàgines web

- Wikipedia [online]. Disponible a: http://en.wikipedia.org (2010 febrer).
- PHP [online]. Disponible a: http://www.php.net (2010 febrer).
- Oracle Corporation. MySQL reference Manual [online]. Disponible a: http://dev.mysql.com/doc/refman/5.0/en/ (2010 febrer).
- The Apache Software Foundation. Disponible a: http://httpd.apache.org/docs/2.0/vhosts/ (2010 gener).
- The jQuery Project. jQuery API. Disponible a: http://api.jquery.com/ (2010 febrer)
- Sergi Quiñonero. Diferentes tipos de ataques en páginas y aplicaciones web [online]. Disponible a: http://www.sergiquinonero.net/diferentes-tipos-de-ataques-en-paginas-y-aplicaciones-web.html (2010 febrer)
- Epsilon Eridani C.B. phpDocumentor [online]. Disponible a: http://www.epsilon-eridani.com/cubic/ap/cubic.php/doc/phpDocumentor---documentacion-para-codigo-PHP-246.html (2010 gener).
- Get Satisfaction [online]. Disponible a: http://getsatisfaction.com/ (2010 gener)

Documents Web

- PHP_IDE_EN [online]. Disponible a: https://spreadsheets.google.com/ccc?key=pV8XyUSUOM7ET07rn4n7NYA#gid=0 (2010 gener)

Annex 1: Exemple de codificació

A continuació es mostren tres exemples de codi corresponents a:

- Exemple de codificació: un controlador
- Exemple de codificació: classe factory
- Exemple de codificació: funció __autoload

Exemple codificació: un controlador

Aquesta classe és la que s'executa periòdicament amb un *cron tab*. Per a cada usuari responsable, li envia la llista de les idees més votades de l'últim període per a les seves àrees de responsabilitat.

```
<?php
/**
 * Idea Responsible Bulletin controller.
 * This class is the responsible for sending the periodical emails when the cron tab is
executed.
 * @author Narcís
 *
 */
class ResponsibleBulletinControllerIdea extends Controller
{
	/**
	 * Class var that will contain the users resposability areas list.
	 *
	 * @var array
	 */
	private $area_idea_list_result = array();

	/**
	 * Mailer object.
	 * @var Mailer
	 */
	private $mailer;

	/**
	 * Sends an email containing a summary of the ideas most voted during the last
period on every user areas of responsability.
	 */
	public function getData()
	{
		$this->loadConfig();

		$area_model			= $this->getModelInstance( 'AreaModelIdea' );
		$idea_model			= $this->getModelInstance( 'IdeaModelIdea' );
		$this->mailer			= Factory::getInstance( 'Mailer' );
		$area_list			= $area_model->getAreaList();
		$users_responsible_list	= $area_model->getUsersResponsabilityAreas();

		foreach ( $area_list as $value )
		{
			$this->area_idea_list_result[ $value['id_area'] ] = $idea_model-
>getListTop( 0, $value['id_area'], $this->configVars['email_list_size'], 1, $this-
>configVars['email_vote_period'] );
		}

		foreach ( $users_responsible_list as $user_id => $value )
		{
			$this->sendMailToResponsible($user_id, $value);
		}

	}

	/**
	 * Method to send an email to a responsible.
	 * @param integer $user_id
	 * @param array $value
	 */
	private function sendMailToResponsible( $user_id, $value )
	{
		$tpl_mail_body = new Template( 'email/idea_responsible_bulletin.tpl.php' );
		$tpl_mail_body->set ( 'idea_lists' , $this->area_idea_list_result );
```

```
                $tpl_mail_body->set ( 'user_areas' , $value['areas'] );

                $this->mailer->AddAddress( $value['email'] );
                $this->mailer->Body = $tpl_mail_body->fetch();
                $this->mailer->Send();
                $this->mailer->ClearAddresses();
        }
}
```

Exemple de codificació: classe Factory

Aquesta classe és la responsable de la creació dels objectes de l'aplicació, assegurant al programador que si ja ha instanciat una classe anteriorment, se li retornarà el mateix objecte, evitant així gastar més memòria de la necessària.

```
<?php
/**
 * Factory Class, responsible for creating singletoned objects.
 *
 * @author Narcís Davins
 */
class Factory
{
        // Propierty that holds classes instances.
        protected static $instances = array();

        // Private constructor to prevent the creation of instances of Factory externally.
        private function __construct(){}

        /**
         * Returns the requested instance.
         *
         * @param string class to be instantiated
         * @return object
         */
        public static function getInstance( $class_name )
        {
                //If it hasn't already been instantiated
                if ( !array_key_exists ( $class_name, self::$instances ) )
                {
                        if ( !class_exists( $class_name ) )
                        {
                                throw new FactoryException( 'Error trying to create "' .
$class_name .'", class doesn\'t exist' , 500 );
                        }
                        self::$instances[$class_name] = new $class_name;
                }

                return self::$instances[$class_name];

        }
}
```

Exemple de codificació: funció __autoload

Aquesta és una funció que permet que no s'hagin d'incloure els fitxers manualment, permetent així també tenir centralitzat el punt des d'on s'inclouen tots els arxius i per tant facilitant la depuració del codi i el canvi en la nomenclatura dels arxius i de les direccions físiques d'aquests.

```
<?php
/**
 * Framework __autoload function.
 * @author Narcís
 */


/**
 * Includes the file for the requested class name
 * @param string $class name of the class to include.
 */
function __autoload( $class )
{

		$pattern = '/^([a-z]+)(controller|model)([a-z]+)$/i';
		preg_match( $pattern , $class , $matches );

		$type = isset($matches[2]) ?  $matches[2] : '';

		switch ( mb_strtolower( $type ) )
		{
			case 'controller':
				$path = PATH_SCRIPTS;
				if ( $path == NULL ) throw new Exception( 'Failed loading
controller path' , 500 );

				list( , $class , , $folder ) = $matches;
				$fullPath = $path . $folder . DIRECTORY_SEPARATOR . $class .
'.Controller.php';

				break;

			case 'model':
				$path = PATH_INCLUDES;
				if ( $path == NULL ) throw new Exception( 'Failed loading
model path' , 500 );

				list( , $class , , $folder ) = $matches;
				$fullPath = $path . $folder . DIRECTORY_SEPARATOR . $class .
'.Model.php';

				break;

			default:
				$path = PATH_INCLUDES;
				if ( $path == NULL ) throw new Exception( 'Failed loading
class path' , 500 );

				$fullPath = $path . $class . '.Class.php';

				break;
		}

		if ( !file_exists( mb_strtolower ( $fullPath ) ) ) throw new Exception (
'File "' . $fullPath . '" not found.' , 404 );

		require_once ( mb_strtolower ( $fullPath ) );
}
```

Annex 2: Captures de pantalla

A continuació es mostren a tall d'exemple diverses captures de pantalla de l'aplicació:

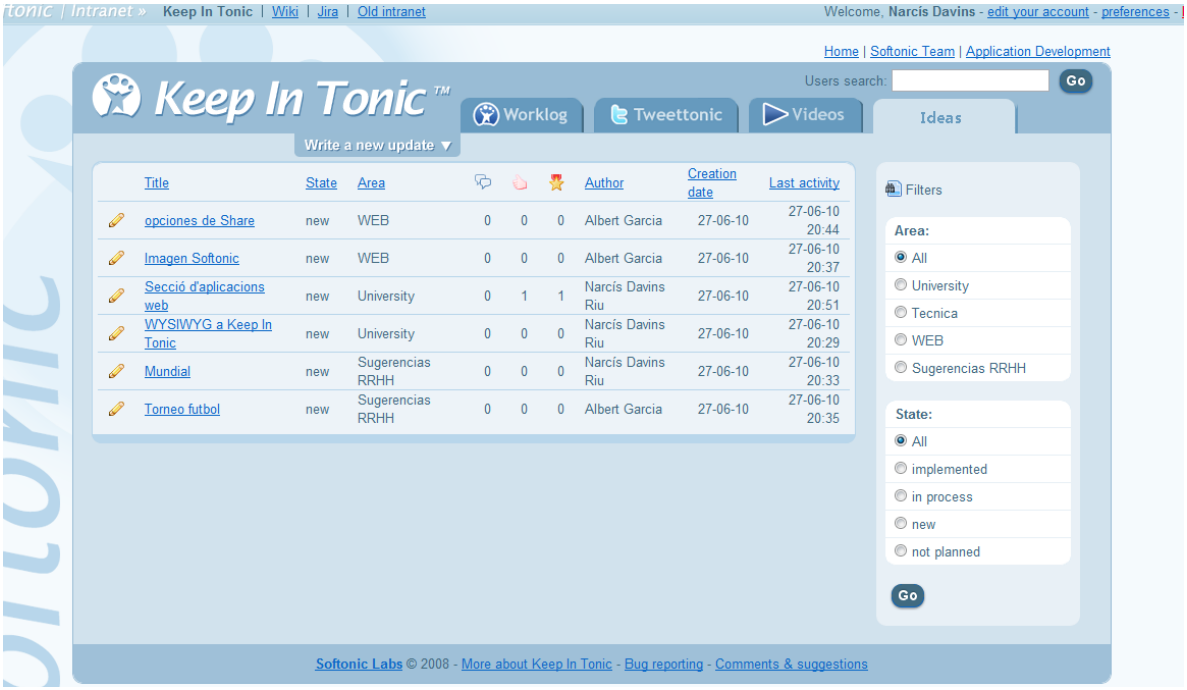
1. Pàgina principal
2. Fitxa d'idea
3. Edició d'idees
4. Moderació d'idees
5. Últimes idees
6. Worklog d'idea

1. Pàgina principal

2. Fitxa d'idea

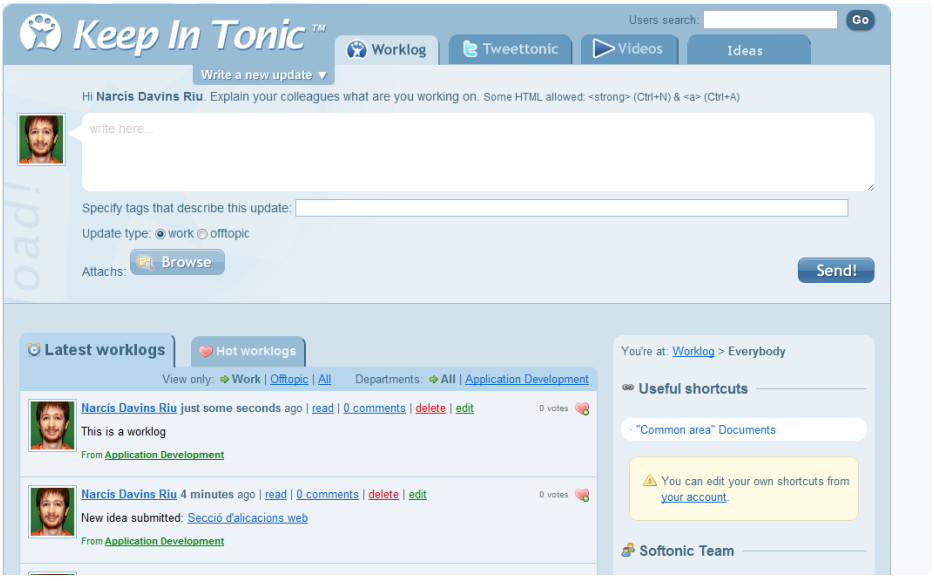
3. **Edició d'idea**

4. **Moderació d'idees**



5. **Últimes idees**

6. **Worklog d'idea**



Narcís Davins Riu
Sabadell, Juny de 2010