Universitat Autònoma
de Barcelona

Disseny d'una eina de suport a la reutilització de codi

Memòria del projecte
d'Enginyeria Tècnica en
Informàtica de Sistemes
realitzat per

Carlos Saura Martínez

i dirigit per

Jaume Pujol Capdevila

Escola d'Enginyeria

Sabadell, *gener* de *2011*

El sotasignat, Jaume Pujol Capdevila,

professor de l'Escola d'Enginyeria de la UAB,

CERTIFICA:

Que el treball al que correspon la present memòria ha estat realitzat sota la seva direcció per Carlos Saura Martínez.

I per a que consti firma la present.

Sabadell, ***gener*** de ***2011***

Signat: Jaume Pujol Capdevila

FULL DE RESUM – PROJECTE FI DE CARRERA DE L'ESCOLA D'ENGINYERIA

Títol del projecte: Eina de suport a la reutilització de codi	
Autor: Carlos Saura Martinez	**Data:** *02/2011*
Tutor: Jaume Pujol Capdevila	
Titulació: Enginyeria técnica en Informàtica de Sistemes	

Paraules clau

- Català: Reutilització de codi, Ruby on Rails, Gestor de continguts
- Castellà: Reutitlización de código, Ruby on Rails, Gestor de contenidos
- Anglès: Code Reuse, Ruby on Rails , Content Manager System

Resum del projecte

Català: Aquest projecte tracta sobre el desenvolupament d'una eina de suport a la reutilització de codi de programació. La eina té com a objectiu augmentar la productivitat dels desenvolupadors per mitjà de la reutilització de codi. Per facilitar la reutilització la eina proposa un sistema jeràrquic on el codi de programació i la diferent informació sobre aquest es guardat en anotacions, les anotacions formen part d'una pàgina i els conjunts de pàgines son organitzats en projectes.

Castellà: Este proyecto trata sobre el desarrollo de una herramienta de soporte a la reutilización de código de programación. La herramienta tiene como objetivo aumentar la productividad de los programadores mediante la reutilización de código. Para facilitar la reutilización la herramienta propone sistema jerárquico donde el código de programación i la diferente información sobre este queda guardado en anotaciones, las anotaciones forman parte de una página y los conjuntos de páginas son organizados en proyectos.

Anglès: This project is about the development of a tool to support the reuse of programming code. The tool aims to increase the programmer productivity through reuse of code. To facilitate reuse the tool proposed hierarchical system where the programming code and the different information about the code is stored in notes, the notes are part of a page and the page sets are organized into projects.

Índex

1 Introducció

1.1 Descripció general

La creació de software es difícil, sobretot quan un programador no disposa d'una amplia experiència. Els desenvolupadors de software reutilitzen bons dissenys ja creats per evitar el redisseny o minimitzar-lo. Es per això, que els dissenyadors i programadors experts saben que no han de crear aplicacions des de cero. Quan troben una solució bona, la fan servir una vegada rere l'altre.

Si es pogués recordar els detalls d'un problema anterior i com es va resoldre, podríem recolzar-nos en aquella experiència sense tornar a inventar la solució. Aquest recolzament seria molt útil pels programadors de qualsevol nivell, però especialment pels no experimentats.

Existeixen diferents eines per documentar projectes de programació. Aquestes eines ens permeten consultar una documentació que ha estat generada dels comentaris al codi font. El tipus de documentació obtinguda per aquestes aplicacions esta enfocada per grans desenvolupaments o per ser entregada al client final. Es per això que no podem considerar que estiguin enfocades a la reutilització de codi.

Una eina de reutilització de codi correcte pot ser beneficiosa per qualsevol desenvolupador, independentment del projecte o llenguatge que faci servir. Però seria especialment beneficiosa pels desenvolupadors que facin servir metodologies àgils, donat que els permetria fer iteracions mes curtes.

Aquest projecte intentarà cobrir la necessitat de ser mes eficient a l'hora de programar software a través de la reutilització. S'estudiarà quina serà la solució mes adient i viable per dur a terme el propòsit indicat. Per últim, es desenvoluparà una eina amb els requisits necessaris per garantir una bona experiència l'usuari i una fàcil reutilització de codi.

1.2 Motivacions personals

Estudiar una carrera universitària no és només obtenir un grapat de coneixements. Des del principi a l'estudiant se li demana aprendre uns coneixements que no es donen a la classe presencial i així començar a "aprendre a aprendre". L'estudiant a mes d'aprendre a aprendre, desenvolupa tècniques per saber com i on ha de buscar informació, en comptes de retenir-la sense sentit.

A les primeres pràctiques de programació vaig descobrir que tenia facilitat de creació i de fer ús de les eines que disposava. Per contra, vaig trobar-me que no m'era tan fàcil recordar el que havia fet a practiques o projectes anteriors. Ara, al final de carrera aquestes característiques encara m'acompanyen en major o menor mesura.

A la hora de programar sempre he intentat buscar alguna eina o mètode per poder enregistrar les parts mes importants del que desenvolupava. En el moment que una funció m'era molt difícil o m'havia costat molt de temps ho enregistrava a un bloc de notes. Quan començava un projecte nou intentava trobar l'aplicació que hem fos mes útil sense resultat, donat que tornava al fitxer de text.

Per altre banda m'agradaria realitzar una eina que no només proporcioni un conjunt de solucions a problemes de programació, per mi el marc ideal es que sempre siguin les millors solucions. Un programador al disposar d'una solució ja creada pot dedicar el temps guanyat a millorar, ampliar o simplement refactoritzar la solució obtinguda. D'aquesta manera cada programador pot contribuir a millorar el repositori de solucions.

Per concloure en aquest projecte m'agradaria cobrir les diferents necessitats que es pot trobar un programador a l'hora de reutilitzar codi. Per portar a terme el projecte faré servir les meves experiències de desenvolupador. Per fer mes extensible el projecte també vull valorar les necessitats dels programadors d'una empresa de desenvolupament de software.

1.3 Objectiu del projecte

L'objectiu d'aquest es augmentar de la productivitat dels desenvolupadors per mitjà de la reutilització de codi de programació. Per poder portar a terme aquest augment i fer us de la reutilització es crearà un sistema que ho permeti de forma fàcil.

El sistema a desenvolupar ha d'assolir una sèrie d'objectius descrits als següents punts:

- Permetre fer ús de l'aplicació des de qualsevol ordinador que disposi d'accés a la xarxa on esta allotjada per mitjà d'un navegador d'internet.
- Permetre l'accés a múltiples usuaris a la informació simultàniament.
- Mostrar informació de forma clara i organitzada per pàgines vinculades projectes.
- El contingut de les pàgines ha de facilitar l'addició de codi de programació i permetre afegir contingut variat que permeti una bona explicació del codi.
- Els projectes s'han de poder vincular a un repositori d'un sistema gestor de versions.
- Mostrar informació sobre el contingut mes rellevant de l'aplicació.
- Permetre cercar al contingut del web.

1.4 Organització de la memòria

La memòria esta estructurada en 5 capítols. Al següent llistat mostrem els títols dels capítols amb una breu descripció del que tracta cada capítol:

- **ESTUDI DE VIABILITAT**

  L'objectiu de l'estudi de viabilitat és, a partir d'un conjunt de necessitats plantejades o objectius, escollir la solució que millor les cobreixi d'entre totes les possibles.

  En l'estudi de viabilitat es tindran en consideració l'abast del projecte i la situació actual. Per poder definir i escollir la solució es realitzarà un llistat de requisits indispensables que ha assolir. Un cop escollida la solució que cobreix els requisits, seran avaluades les possibles alternatives. Per últim es

realitzarà una planificació i un anàlisi de la relació cost-benefici per conèixer si el projecte es viable.

- **ANÀLISI DEL SISTEMA**

  En aquest capítol es detalla la solució escollida entre les descrites a l'estudi de viabilitat.

  La solució serà dividia en seccions i subseccions per ser descrites mes acuradament. Per cada secció de la solució es farà un breu descripció i detallarà quins requisits ha d'acomplir.

  L'anàlisi del sistema també contindrà una definició bàsica de les possibles interfícies d'usuari de les que disposarà la solució.

  En aquest apartat també s'indicarà quin model de desenvolupament seguirà el projecte i per quines raons. Per últim s'indicarà quin serà el pla de proves que ha de seguir la solució.

- **DISSENY DEL SISTEMA**

  L'objectiu de la fase de disseny del projecte es obtenir els models i especificacions que s'han definit a l'anàlisi del sistema.

  El primer apartat d'aquest capítol es indicar quina serà l'arquitectura de software del projecte i les pràctiques de desenvolupament que es portaran a terme. Tant l'arquitectura com les pràctiques seran definides i s'indicarà per quines raons han estat escollides.

  El següent a indicar en aquest capítol es la tecnologia escollida per desenvolupar la solució amb els requisits indicats a l'anàlisi.

  La part principal de disseny del sistema serà descriure com es desenvoluparan les diferents seccions de la solució incloent diagrames de casos d'ús i Entitat-Relació quan sigui necessari. Aquest disseny serà una base prèvia per fer un desenvolupament estructurat.

- **DESENVOLUPAMENT I PROVES**

  En el capítol de desenvolupament contindrà una descripció dels elements que s'han creat per portar a terme una secció de la web. Per cada secció

s'indica quin model fa servir, la seves accions per interactuar entre l'usuari i el sistema, i les visualitzacions de la secció.

En aquest capítol s'inclouran les proves que s'han realitzat al llarg del desenvolupament. S'indicaran tant les proves per mitjà de testos unitaris i funcionals, com les proves realitzades manualment.

- **CONCLUSIÓ**

  En aquest últim capítol es farà una reflexió dels resultats obtinguts i es formularan les possibles ampliacions que es podrien realitzar.

2 Estudi de viabilitat

En els següents apartats s'estudiarà en línies generals quins problemes es volen resoldre, quines solucions possibles hi ha i quina és la més adequada.

2.1 Abast del sistema

L'objectiu de qualsevol organització és treure el màxim partit del menor temps possible. Per els desenvolupadors una manera d'aconseguir-ho és disposant d'una bona eina per reutilitzar el codi dels anteriors projectes. La implantació d'una eina per reutilitzar codi tindria repercussió als següents departaments d'una empresa:

- **DESENVOLUPAMENT**: Els grups de desenvolupament podran augmentar la velocitat de creació d'aplicacions de forma exponencial. Quan més solucions introdueixin a l'eina, mes podran reutilitzar. Tindran que fer menys dissenys des de cero, donat que agafaran solucions anteriors com a base. Al introduir noves incorporacions al departament, aquests tindran una millor corba d'aprenentatge.

- **FINANCES**: El dissenyadors de software podran aprofitar solucions anteriors i no caldrà reinventar-les. Aquest fet escorçarà hores de programació i per tant els beneficis dels projectes seran majors.

- **VENDES**: Els venedors podran ajustar el preu per aconseguir mes projectes per l'empresa.

S'han mostrat diferents avantatges de l'eina en una empresa, però també pot ser útil per a altres col·lectius. Un programador amateur, un estudiant o un grup de recerca disposarien dels mateixos beneficis que el departament de desenvolupament d'una empresa.

2.2 Objectius a assolir

Els següents punts es mostren el diferents objectius que els requisits han de cobrir.

1. **Objectius sobre l'accés al sistema.**
    a. Permetre l'accés des de qualsevol ordinador amb un navegador d'internet.
    b. Ha de ser accessible des de els navegadors actuals.
    c. Varis usuaris han de poder fer ús simultàniament.
    d. Permetre la gestió d'usuaris i el control d'accés.
    e. Mantenir una sessió oberta.
2. **Organització de la informació.**
    a. La informació s'ha de poder guardar de forma jeràrquica.
3. **Visualització de la informació.**
    a. S'ha de facilitar l'addició de codi de programació.
    b. Permetre guardar el diferents projectes on es guarda la informació.
    c. Disposar d'un gestor de pàgines o notes, on guardar informació.
    d. La informació de les pàgines de ser diferent procedència.
4. **Accés a la informació.**
    a. Permetre ressaltar informació.
    b. Permetre guardar informació personalitzada.
    c. Disposar d'un sistema per cercar informació
5. **Disseny.**
    a. El disseny i aspecte visual de ser consistent.
6. **Velocitat.**
    a. Fer servir recursos per millorar la velocitat de carrega.
7. **Solidesa**.
    a. Disposar d'un sistema de notificacions i control d'errors.
    b. Permetre realitzar ampliacions del sistema.
    c. La informació ha d'estar emmagatzemada en un sistema que sòlid.
8. **Compliment d'estàndards.**
    a. Complir el estàndards marcats per la W3C.

9. **Compromís amb el software lliure.**
    a. Fer ús de software lliure pel desenvolupament.
    b. Ser alliberada amb una llicència que permeti la seva continuïtat.
10. **Aspectes econòmics.**
    a. Per realitzar el projecte, els sistemes han de tenir el menor cost possible.

2.3 Anàlisi de Requisits

Els requisits són la descripció i especificació d'un software. Inclouen tant els serveis que el client requereix del sistema, com les restriccions sota les quals funcionarà o serà desenvolupat.

A l'anàlisi de requisits s'estableixen els requisits que ha de complir el resultat de desenvolupament per poder-se anomenar producte final. L'anàlisi de requisits també serveix de contracte entre els desenvolupadors i el client. Una vegada validats per part del client , els desenvolupadors es responsabilitzen de incloure tot allò que s'hagi definit en aquests.

Mitjançant l'estudi de la situació actual, el punt de millora i problemes detectats, s'han identificat els següents requisits.

2.3.1 Requisits funcionals

Els requisits funcionals son una descripció del diferents serveis que ha d'aportar el sistema, com han de reaccionar a unes entrades definides i el comportament en les possibles situacions. Els requisits funcionals que ha de complir el projecte son:

1) Disposar de gestió d'usuaris amb diferents rols.
2) Permetre l'accés a l'aplicació per mitjà d'una identificació d'usuari i una contrasenya.
3) Restricció d'accés a qualsevol contingut de l'aplicació a usuaris no registrats.
4) Permetre l'accés a diferents usuaris de forma simultània.
5) Disposar d'un gestor de projectes on crear, editar i eliminar projectes.
6) Disposar d'un gestor de pàgines o notes, on guardar informació sobre del codi font a reutilitzar.

7) Facilitar la organització de les pàgines per projectes
8) La informació de les pàgines o notes ha de poder ser com a mínim, codi font, text amb format i imatges.
9) Els projectes s'han de poder vincular a un repositori d'un sistema gestor de versions i ser visibles via web.
10) Permetre cercar contingut i ser mostrat de forma clara.
11) Els usuaris registrats a la web han de poder guardar-se pàgines o notes com a preferides.
12) Disposar d'eines per mostrar la informació mes utilitzada o rellevant.

2.3.2 Requisits no funcionals

Els requisits no funcionals son no descriuen les funcionalitats de l'aplicació, sinó les propietats de les que ha de disposar l'aplicació en quant a qualitat.

1) L' ús de l'aplicació s'ha de poder realitzar des de qualsevol ordinador que disposi d'accés a la xarxa on esta allotjada per mitjà d'un navegador d'internet.
2) El navegadors d'internet suportats son: Internet Explorer 7 i 8, Safari 4 i 5, Opera 9, i Firefox de la versió 3.0 a la 3.6 .
3) Mostrar informació de forma clara i organitzada per pàgines vinculades projectes.
4) El contingut de les pàgines ha de facilitar l'addició de codi de programació i permetre afegir contingut variat que permeti una bona explicació del codi.
5) Un cop identificat un usuari a l'aplicació amb les dades correctes, no haurà de tornar a entrar aquestes dades fins que no doni l'acció de desconnectar-se.
6) El disseny de l'aplicació ha de ser consistent per totes les pàgines.
7) L'aplicació de permetre tractar i modificar la informació guardada minimitzant els temps de carrega i les recarregues de pàgina.
8) Si un usuari vol accedir a un secció sense estar registrat se li ha de denegar l'accés i facilitar el registre.
9) Per a què l'aplicació Web sigui consistent, tots els mecanismes s'hauran d'utilitzar sempre de la mateixa manera, per tal de no confondre a l'usuari, on les seccions estaran dissenyades amb una estètica semblant.

10) En quant a solidesa el sistema guiarà, en certes ocasions, al usuari. Ho farà a partir de notificacions d'avís, o simplement, de caràcter informatiu. De la mateixa manera, l'aplicació controlarà els possibles errors que es produeixin en temps d'execució i aquests seran notificats a l'usuari.
11) El lloc haurà de complir les normatives i/o estàndards de la marcats per la World Wide Web Consourtium (HTML, CSS ..).
12) La informació haurà de ser emmagatzemada en un sistema gestor de base de dades.
13) L'aplicació ha de permetre realitzar ampliacions i millores, i disposar d'un sistema de detecció d'inconsistències.

2.3.3 Requisits legals

Els requisits legals determinen els aspectes legals que ha de complir l'aplicació.

1) El software que permetrà que l'aplicació funcioni ha de ser de software lliure i amb una comunitat que asseguri la seva continuïtat.
2) L'aplicació desenvolupada ha de disposar de la llicència en el menys restrictiva possible

2.3.4 Requisits econòmics

Els requisits econòmic descriuen els aspectes econòmics del projecte.

1) L'aplicació ha fer ús de software amb llicència lliure de manera que no hi hagin despeses per llicenciament de software.
2) La despesa de corresponent al sistema operatiu ha de ser la mes petita possible.

2.4 Relació dels objectius amb els requisits

En la següent taula es reflexa la relació entre els objectius del projecte i els requisits:

Objectiu 1:

Permetre l'accés des de qualsevol ordinador amb un navegador d'internet.

Requisit no funcional 1:

L' ús de l'aplicació s'ha de poder realitzar des de qualsevol ordinador que disposi d'accés a la xarxa on esta allotjada per mitjà d'un navegador d'internet.

Objectiu 2:

Ha de ser accessible des de els navegadors actuals

Requisit no funcional 2:

El navegadors d'internet suportats son: Internet Explorer 7 i 8, Safari 4 i 5, Opera 9, i Firefox de la versió 3.0 a la 3.6 .

Objectiu 3:

Varis usuaris han de poder fer ús simultàniament.

Requisit funcional 4:

Permetre l'accés a diferents usuaris de forma simultània.

Objectiu 4:

Permetre la gestió d'usuaris i el control d'accés.

Requisit funcional 1:

Disposar de gestió d'usuaris amb diferents rols.

Requisit funcional 2:

Permetre l'accés a l'aplicació per mitjà d'una identificació d'usuari i una contrasenya.

Requisit Funcional 3:

Restricció d'accés a qualsevol contingut de l'aplicació a usuaris no registrats.

Requisit no Funcional 8:

Si un usuari vol accedir a un secció sense estar registrat se li ha de denegar l'accés i facilitar el registre.

Objectiu 5:

Mantenir una sessió oberta.

Requisit no funcional 5:

Un cop identificat un usuari a l'aplicació amb les dades correctes, no haurà de tornar a entrar aquestes dades fins que no doni l'acció de desconnectar-se.

Objectiu 6:

La informació s'ha de poder guardar de forma jeràrquica.

Requisit funcional 7:

Facilitar la organització de les pàgines per projectes

Objectiu 7:

S'ha de facilitar l'addició de codi de programació.

Requisit funcional 9:

Els projectes s'han de poder vincular a un repositori d'un sistema gestor de versions i ser visibles via web.

Requisit no funcional 4:

El contingut de les pàgines ha de facilitar l'addició de codi de programació i permetre afegir contingut variat que permeti una bona explicació del codi.

Objectiu 8:

Permetre guardar el diferents projectes on es guarda la informació.

Requisit funcional 5:

Disposar d'un gestor de projectes on crear, editar i eliminar projectes.

Objectiu 9:

Disposar d'un gestor de pàgines o notes, on guardar informació.

Requisit funcional 6:

Disposar d'un gestor de pàgines o notes, on guardar informació sobre del codi font a reutilitzar.

Objectiu 10:

La informació de les pàgines de ser diferent procedència

Requisit funcional 8:

La informació de les pàgines o notes ha de poder ser com a mínim, codi font, text amb format i imatges.

Objectiu 11:

Permetre ressaltar informació.

Requisit funcional 12:

Disposar d'eines per mostrar la informació mes utilitzada o rellevant.

Objectiu 12:

Permetre guardar informació personalitzada

Requisit funcional 11:

Els usuaris registrats a la web han de poder guardar-se pàgines o notes com a preferides.

Objectiu 13:

Permetre guardar informació personalitzada

Requisit funcional 11:

Disposar d'un sistema per cercar informació.

Objectiu 14:

El disseny i aspecte visual de ser consistent

Requisit no funcional 6:

El disseny de l'aplicació ha de ser consistent per totes les pàgines.

Objectiu 15:

Fer servir recursos per millorar la velocitat de carrega

Requisit no funcional 7:

L'aplicació de permetre tractar i modificar la informació guardada minimitzant els temps de carrega i les recarregues de pàgina.

Objectiu 16:

Disposar d'un sistema de notificacions i control d'errors.

Requisit no funcional 10:

En quant a solidesa el sistema guiarà, en certes ocasions, al usuari. Ho farà a partir de notificacions d'avís, o simplement, de caràcter informatiu. De la mateixa manera, l'aplicació controlarà els possibles errors que es produeixin en temps d'execució i aquests seran notificats a l'usuari.

Objectiu 17:

Permetre realitzar ampliacions del sistema.

Requisit no funcional 13:

L'aplicació ha de permetre realitzar ampliacions i millores, i disposar d'un sistema de detecció d'inconsistències.

Objectiu 18:

La informació ha d'estar emmagatzemada en un sistema que sòlid

Requisit no funcional 12:

La informació haurà de ser emmagatzemada en un sistema gestor de base de dades.

Objectiu 19:

Complir el estàndards marcats per la W3C

Requisit no funcional 11:

El lloc haurà de complir les normatives i/o estàndards de la marcats per la World Wide Web Consourtium (HTML, CSS ..).

Objectiu 20:

Fer ús de software lliure pel desenvolupament

Requisit no funcional 11:

El lloc haurà de complir les normatives i/o estàndards de la marcats per la World Wide Web Consourtium (HTML, CSS ..).

Objectiu 21:
Ser alliberada amb una llicència que permeti la seva continuïtat
Requisit legal 1:
El software que permetrà que l'aplicació funcioni ha de ser de software lliure i amb una comunitat que asseguri la seva continuïtat.
Requisit legal 2:
L'aplicació desenvolupada ha de disposar de la llicència en el menys restrictiva possible

Objectiu 22:
Per realitzar el projecte, els sistemes han de tenir el menor cost possible
Requisit econòmic 1:
L'aplicació ha fer ús de software amb llicència lliure de manera que no hi hagin despeses per llicenciament de software.
Requisit econòmic 2:
La despesa de corresponent al sistema operatiu ha de ser la mes petita possible.

2.5 Estudi de la situació actual

A l'actualitat no hi ha gaires aplicacions dedicades a la reutilització de codi de programació. Podem trobar varies eines que ens poden proporcionar una ajuda però molt lluny de ser específiques per la reutilització. Tot seguit descrivim algunes d'aquestes eines:

2.5.1 Generadors de documentació

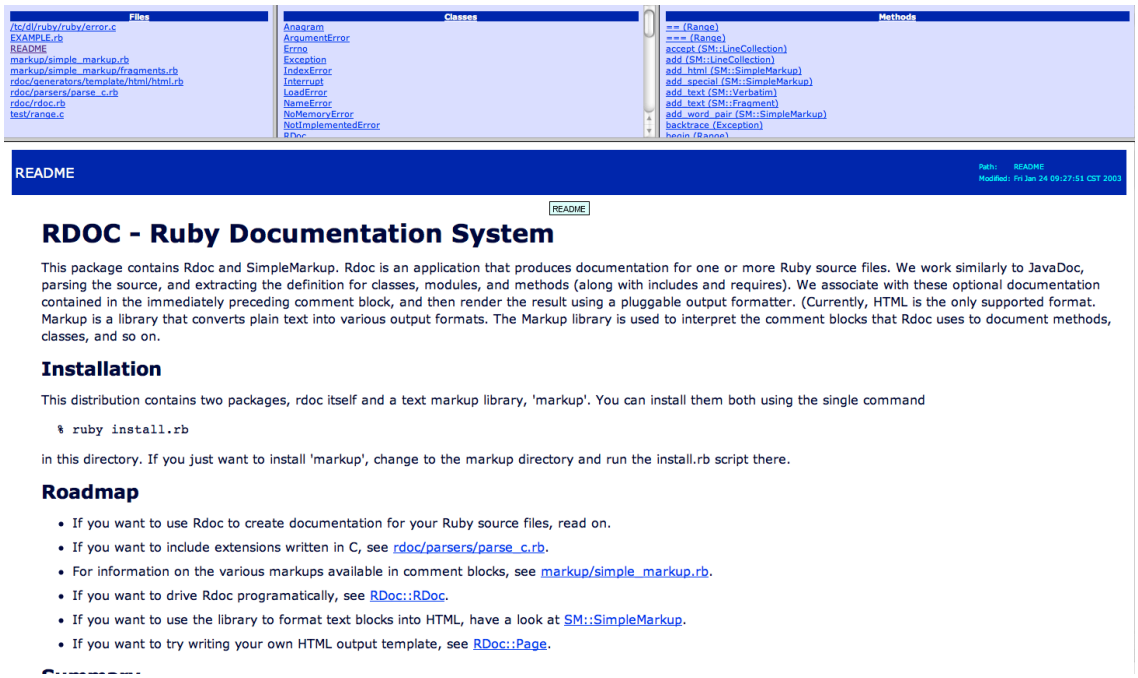
Els generadors de documentació són eines que autogeneren un conjunt de documents que aporten informació sobre el codi font d'un projecte. Per la creació dels documents es fa servir la estructura del mateix codi font i els comentaris a aquest.

Aquest tipus d'aplicacions es solen fer servir a projectes interns de gran envergadura, a projectes de codi obert o com a suport per ser entregat al client final de l'aplicació. Per els àmbits descrits els generadors de documentació son idonis per la seva facilitat i el poc esforç per els desenvolupadors, donat que es el propi generador qui s'encarrega de organitzar i realitzar els documents.

A la xarxa ràpidament podem trobar un ampli ventall de generadors de documentació amb les característiques bàsiques en comú. Però també es fàcil detectar que cada generador aporta alguna característica nova que la fa diferenciadora de la resta. Un exemple el podem trobar en els generadors específics que estan centrats en un sòl llenguatge i integrats a l'entorn de desenvolupament. D'altres son completament externs a l'entorn de programació però poden generar documentació de diferents llenguatges de programació.

Podem dir que els generadors de documentació ajuden a la reutilització de codi per donat que permeten disposar de manera agrupada les funcions i classes de codi conjuntament amb les seves descripcions.

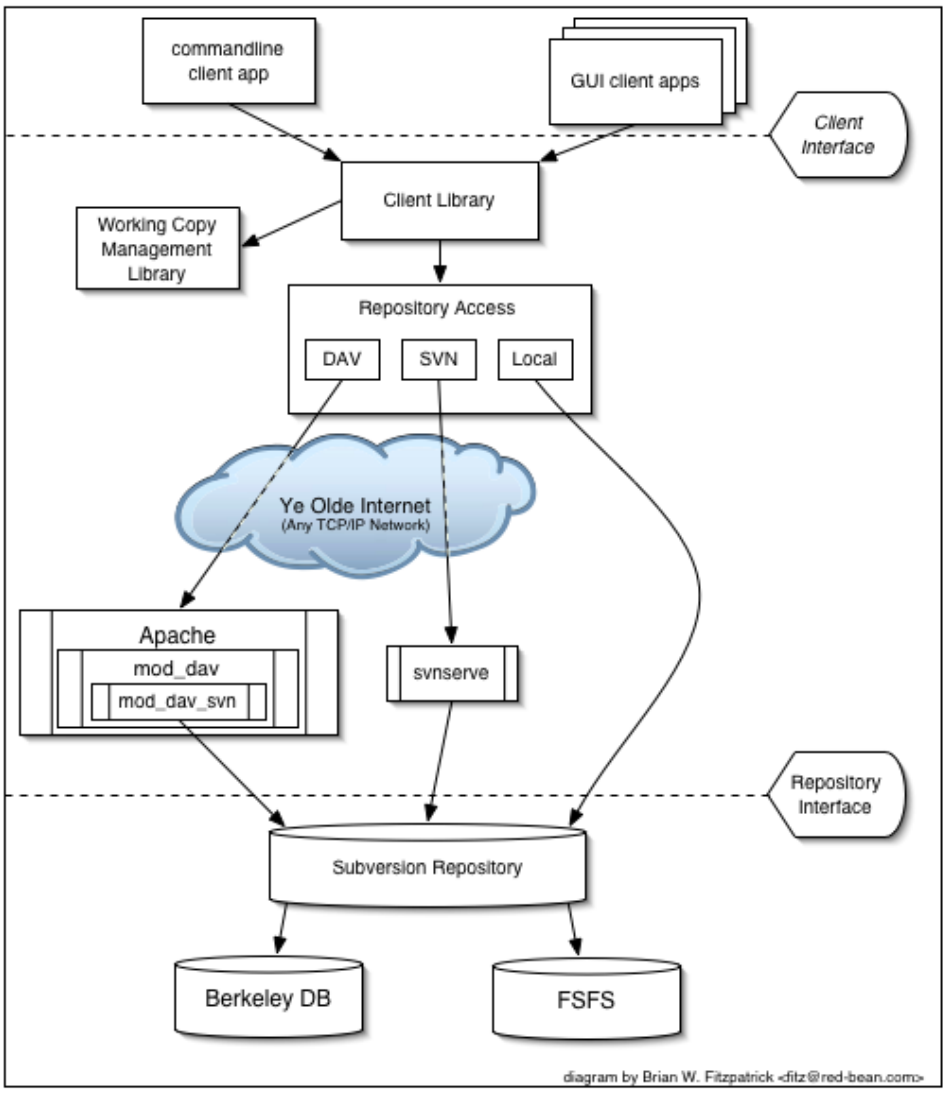
Exemples de generadors de documentació: RDoc, JavaDoc, Doxygen



2.5.2 Sistemes gestors de versions

Els sistemes gestor de versions son aplicacions que permeten mantindré un control de canvis al projecte. Permet saber en quin estat esta un fitxer i els canvis que ha des de que es va afegir al repositori. Un sistema gestor de versions ajuda a la reutilització mantenint tot el codi d'un projecte agrupat i permetent veure l'històric de cada fitxer. Aquest tipus d'aplicacions facilita el mantenir tots els projectes disponibles per tots els desenvolupadors, el codi actualitzat i disponible per visualitzar-lo de forma natural.

Exemples: Git, Subversion, Mercurial



2.5.3 Gestors de continguts

Els gestors de contingut son aplicacions per guardar informació de forma clara i ordenada. Les capacitats que proporcionen son les de creació, edició, gestió i publicació de contingut digital en diversos formats. El contingut es estructurat en pàgines web per ser visualitzat. Per consultar una pàgina un usuari ha de fer una petició a una l'adreça única que serà captada pel servidor que extraurà de la base de dades el contingut. En molts casos a mes de permetre gestionar el contingut per pàgines, els gestors de continguts faciliten gestions de pàgines per categories, projectes, seccions o altres. La majoria de gestors de contingut faciliten una gestió d'usuaris per controlar l'accés i un sistema de cerca de contingut.

A internet es fàcil trobar diferents aplicacions, llibreries, frameworks o plugins que fan servir gestors de continguts. Ens alguns casos es fa servir wikis per mostrar informació sobre el projecte, preguntes freqüents o tècniques de desenvolupament. Altres aplicacions o projectes fan servir un gestor de continguts propi per mostrar als desenvolupadors com treballar-hi.

Podem indicar que els gestors de continguts faciliten la reutilització de codi per la seva facilitat d'edició i accés al contingut. Poder gestionar la informació per pàgines i agrupar-les permet tenir un estructura a seguir per cercar informació sobre el codi.

Exemples:
(wiki) wiki.rubyonrails.com on es detalla informació del framework Ruby on Rails i com a començar a treballar amb aquest.

(gestor de continguts propis) api.jquery.com i developer.apple.com .

2.6 Valoració d'alternatives

En aquest apartat es mostren possibles solucions que no seran desenvolupades en aquest projecte.

2.6.1 Creació o ampliació d'un generador de documentació

Aquesta possible solució no es viable donada la complexitat de la creació un generadors de documentació amb els requisits determinats. Aquesta solució hauria de ser un generador de documentació que per cada canvi a cada projecte repassés tot el codi i tornés a generar la documentació. La complexitat d'aquest solució incrementa en el moment que el contingut ha de ser de varies fonts, dinàmic i permetre una gestió d'usuaris.

2.6.2 Creació d'una extensió per un gestor de continguts

La creació d'una extensió per un gestor de contingut permetria no hagué de desenvolupar algunes parts com la gestió de pàgines o la d'usuaris. Per altre banda la extensió hauria d'incloure un centralització a codi de programació i una vinculació amb un gestor de sistema de versions per visualitzar el codi. Aquesta solució no s'ha considerat viable donat que no s'ha trobat un gestor de continguts fàcilment extensible i alhora no sobrecarregui el projecte amb funcions no necessàries.

2.7 Solució escollida

La solució escollida es desenvolupar una aplicació tipus gestor de continguts des de cero. Les eines descrites a l'estat de l'art tenen unes característiques molt bones que poden ajudar a la reutilització de codi però no compleixen els requisits del projecte. Es per això el gestor a desenvolupar portarà incorporades algunes de les característiques d'aquestes eines.

2.8 Planificació

L'apartat de planificació mostra els recursos dels quals es disposa per la creació del projecte, descriu i quantifica les diferents etapes i planifica les diferents etapes segons els recursos.

2.8.1 Recursos

2.8.1.1 Personals

Els recursos personals amb els quals es comptaran son els següents:

- **Cap de projecte**

  El cap de projecte s'encarrega de distribució de tasques i la creació dels documents.

- **Analista**

  L'analista es la persona de captar les necessitats i reflectir-les al document d'anàlisi del sistema. També serà la encarregada del disseny del sistema.

- **Desenvolupador**

  El desenvolupador es la persona encarregada de programar el sistema dissenyat.
- **Dissenyador**

  El dissenyador es l'encarregat de l'aspecte visual de l'aplicació final.

2.8.1.2 Materials

Son els recursos material els quals son necessaris pel desenvolupament i implementació del projecte.

Pel desenvolupament del projecte serà necessària un equip on els diferents actors del projecte puguin crear la seva part corresponent del projecte.

Per la posada en producció es necessari un hardware i software amb el qual es pugui fer servir de servidor.

2.8.2 Llistat d'etapes

El següent esquema mostra les diferents etapes del projecte amb les hores de dedicació per cada una.

	Etapes	Responsable	Hores
1	**Definició del projecte**	**Cap de projecte**	**16 hores**
	Objectius generals		
2	**Estudi de viabilitat**	**Cap de Projecte**	**75 hores**
	Definició de l'abast del sistema		
	Realització del detall dels objectius		
	Anàlisi dels requisits		
	Estudi de la situació actual		
	Valoració d'alternatives		
	Definició de la solució		
	Definició dels recursos i les etapes		
	Anàlisi cost-benefici		
3	**Anàlisi del sistema**	**Analista**	**60 hores**
	Anàlisi dels diferents apartats		

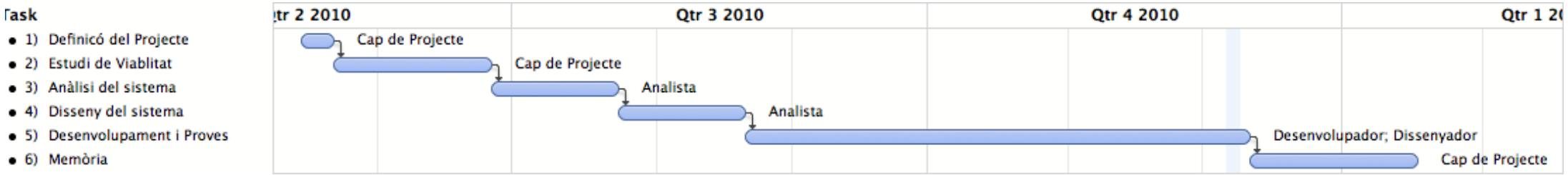
	Anàlisi de les característiques comunes		
4	**Disseny del sistema**	**Analista**	**60 hores**
	Arquitectura i Tecnologia		
	Disseny de l'aplicació		
	Planificació del desenvolupament		
5	**Desenvolupament i proves**	**Desenvolupador Dissenyador**	**480 hores**
	Desenvolupament		
	Creació de les proves		
6	**Realització de la memòria**	**Cap de projecte**	**80 hores**
		Total:	**771 hores**

2.8.3 Gantt

El següent quadre mostra la planificació de les diferents tasques descrites.

Task	Duration	Start	End	Dependencies	Assigned
1) Definicó del Projecte	2d	15/05/10 08:00	22/05/10 09:00		Cap de Projecte
2) Estudi de Viablitat	1w 4d 3h	22/05/10 09:00	26/06/10 09:00	1	Cap de Projecte
3) Anàlisi del sistema	1w 2d 4h	26/06/10 09:00	24/07/10 09:00	2	Analista
4) Disseny del sistema	1w 2d 4h	24/07/10 09:00	21/08/10 09:00	3	Analista
5) Desenvolupament i Proves	6w	21/08/10 09:00	11/12/10 09:00	4	Desenvolupador; Dissenyador
6) Memòria	2w	11/12/10 09:00	17/01/11 20:00	5	Cap de Projecte

Al següent diagrama de gantt es mostra la planificació del projecte a realitzar segons les seves etapes.


Task
1) Definicó del Projecte
2) Estudi de Viablitat
3) Anàlisi del sistema
4) Disseny del sistema
5) Desenvolupament i Proves
6) Memòria
tr 2 2010
Qtr 3 2010
Qtr 4 2010
Qtr 1 2(
Cap de Projecte
Cap de Projecte
Analista
Analista
Desenvolupador; Dissenyador
Cap de Projecte

2.9 Anàlisi Cost-Benefici

2.9.1 Costos Personals

El costos personals son aquells costos derivats de les hores dedicades per les persones que treballen al projecte. Ens aquest cas hi ha una única persona per tots els rols.

La persona encarregada es Carlos Saura Martinez. Estudiant d'Enginyeria Informàtica en Sistemes amb experiència com a desenvolupador web.

Els costos per hora de dedicació pels diferents rols son :

- **Cap de projecte : 30€/hora**
- **Analista: 25€/hora**
- **Desenvolupador: 20 €/hora**
- **Dissenyador: 20€/hora**

La següent taula mostra per cada etapa les hores de dedicació requerides i el cost total de l'etapa.

	Etapes	Hores x preu/hora	Cost
1	Definició del projecte	16 hores x 30€	480€
2	Estudi de viabilitat	75 hores x 30€	2250 €
3	Anàlisi del sistema	60 hores x 25€	1500 €
4	Disseny del sistema	60 hores x 25€	1500 €
5	Desenvolupament i proves	480 hores x 20€	9600 €
6	Realització de la memòria	80 hores x 30€	2400 €
		Total:	17730 €

2.9.2 Costos Materials

Pel desenvolupament es faran servir el iMac i el Macbook Pro propietats dels projectistes. No serà necessària la compra de cap software ni llicència donat que serà desenvolupament serà realitzat amb programari lliure.

En canvi els costos materials que suposa la implantació de l'eina poden variar segons el material disponible i el volum de d'usuaris. Per una implantació mitjana en un grup d'entre 15 i 200 desenvolupadors es el següent:

- Servidor:

Mac mini con Snow Leopard Server

2.66 GHz : Dual 500 GB
Core 2 Duo de Intel a 2,66 GHz a 2.0 GHz
4 GB de memoria
Unidades de disco duro doble de 500 GB a 7.200 rpm[1]
Gráficos GeForce 320M de NVIDIA
Mac OS X Server Snow Leopard
Preu : **1000,00 €**

- Connexió a internet (Si realitza consultable via internet) :

Ono 50Mb + trucades
Preu: **45,00 €/mes**

2.9.3 Cost general

El cost general del sistema es la suma dels costos personals i del costos materials. El cost fixes per la creació del sistema son 18.730€ i els costos mensuals son 45€.

2.10 Relació Cost-Benefici

El benefici d'introduir una eina per la reutilització de codi es pot observar tant en temps com en cost. Podem veure la següent hipotètica situació en un marc de desenvolupament:

Desenvolupador_1 fa la funcionalitat funX pel projecte Proj1. En la realització d'aquesta funcionalitat ha gastat 8 hores en cerca d'informació i 10 en desenvolupament.

Cost de la funcionalitat funX = 18 hores i 540 € (cost hora desenvolupador 30€)

A les 2 setmanes el Desenvolupador_2 pel Projecte Proj2 requereix fer la funcionalitat funXY, que es pràcticament igual a la funcionalitat funX. Amb l'eina de documentació de codi podem estimar que el Desenvolupador_2 ha gastat 30 minuts en buscar la informació i 2 hores en adaptar-la al seu projecte.

Cost de la funcionalitat funXY = 2 hores i 30 minuts, i 75 €.

Com podem observar amb l'eina de reutilització de codi es pot estalviar en temps i cost de producció. Pel grup de desenvolupadors suposa un estalvi de mes de 15 hores de desenvolupament i 465 € en la situació esmentada.

El càlcul de l'amortització del desenvolupament depèn del número de desenvolupadors els quals el facin servir el sistema i del seu ús. El període d'amortització depèn del ús que es faci de l'aplicació i el nombre d'hores estalviades.

Per l'empresa creadora del sistema de reutilització de codi serà amortitzada quan el nombre d'hores recuperades arribi al cost de desenvolupament que son 18.730€. Però donat que el sistema serà alliberat amb llicència lliure qualsevol empresa diferent de la creadora tindrà beneficis des de l'inici de la implantació.

3 Anàlisi del sistema

El capítol que es presenta a continuació es un anàlisi de les funcionalitats que compondrà la eina de reutilització de codi. La eina de reutilització de codi de programació té com a objectiu principal el proveir al desenvolupadors de software d'un entorn que permeti guardar les seves creacions mes rellevants. Els elements guardats han de ser consultables per l'autor del element o per qualsevol desenvolupador amb accés que pugui treure profit.

La eina de reutilització de codi donarà accés a un conjunt de solucions a problemes de programació de les qual es podran beneficiar qualsevol desenvolupador amb accés. La eina preveu millorar augmentar el rendiment dels grups de desenvolupadors, proveint de solucions ja realitzades i implementables, i alliberant al programadors de tasques repetitives.

Els desenvolupadors que utilitzin practiques àgils de desenvolupament poden centra-se en altres parts del desenvolupament fora de la codificació funcional. Amb un esbós de solució es mes fàcil aplicar practiques de programació com el Desenvolupament Dirigit per Testos (TDD) o treure'n mes profit de les metodologies àgils com la Programació Extrema. Tot seguit es detalla quines millores aportaria una eina de reutilització de codi als exemples de practiques de desenvolupament i de metodologia àgil mencionades:

Desenvolupament dirigit per testos
Es una pràctica de desenvolupament que consta de dos parts, desenvolupar els testos primer i refactoritzar. Per fer servir aquesta practica s'escull un requisit, es redacta un test per aquest requisit, i es comprova que el test falli. Després s'implementa un codi que passi el test redactat i després es refactoritza el codi per disposar d'un codi net i funcional. La eina per reutilitzar codi es útil en el moment de realitzar el test, proporcionant d'una idea general per codificar el test. També pel cas disposar de la primera solució sense refactoritzar per després refactoritzar-la. En tots dos casos es permet guanyar un temps de desenvolupament i el desenvolupador pot mantenir un nivell mes alt de creativitat amb menys desgast

mental. Un cop refactoritzada la solució es pot tornar a pujar a l'eina de reutilització de codi i així contribuir a disposar sempre de les millors solucions.

Programació Extrema

La programació extrema va ser formulada l'any 1996 i es una de les metodologies àgils mes destacades. En aquesta metodologia es considera que els canvis de requisits sobre la marxa es un procés natural i inevitable. Esperar canvis de requisits a qualsevol punt de la vida d'un projecte i adapta-se aquests, es millor i mes realista que intentar definir tots els requisits al principi el projecte. Els valors de la programació extrema son: simplicitat, comunicació, retroalimentació, coratge i respecte. En el següent llistat s'indica diferents raons per es quals la eina de reutilització de codi es adient per la programació extrema:

- Permet **simplificar** els dissenys per agilitzar el desenvolupament basant-se en anteriors dissenys ja creats i correctament explicats.
- Si es dedica menys temps al desenvolupament perquè es mes àgil, aquest temps pot ser dedicat a la **comunicació** amb el client, que normalment a la programació extrema forma part d'equip de desenvolupament. També serveix com a eina per **comunicar** el codi de programació entre desenvolupadors.
- Donat que es disposa del client integrat al grup de desenvolupament aquest pot proporcionar la seva opinió sobre cada cicle o iteració. El desenvolupament es mes àgil perquè s'han reutilitzat codi els cicles poden ser mes curts i mantenir un control mes estricte al canvis. La mateixa eina per reutilització es una font de **retroalimentació** dins del grup de desenvolupadors.
- El **coratge** intervé en el moment en que a la programació extrema es decideix fer programació per parelles. Però si hi ha programació en parelles en un moment de bloqueig una de les persones pot cercar idees a l'aplicació de reutilització de codi mentre que l'altre continua ideant solucions.
- En reutilitzar una solució ja creada es disposa de temps per millorar-la, afegir mes documentació, augmentar el nombre de testos i altres tasques que serveixen per augmentar la qualitat del desenvolupat. Fer aquestes tasques i

afegir noves solucions a l'eina de reutilització es **respectar** a la resta de desenvolupadors amb els quals es treballa.

En els següents apartats es detalla la solució escollida a l'estudi de viabilitat com a eina de reutilització de codi. La solució serà dividia en seccions i subseccions descriure mes acuradament cada funcionalitat de l'eina. Per cada funcionalitat es farà un breu descripció i detallarà quins requisits ha d'acomplir.

3.1 Secció Gestor de Projectes

El gestor de projectes ha de permetre manipular la informació dels projectes afegits a l'aplicació.

Aquest gestor ha de disposar de 3 visualitzacions:

- Llistat

  Aquesta vista ha de llistar els diferents projectes afegits a l'aplicació en ordre descendent de data de creació. La informació de visualitzar per aquesta vista es:

  - Nom del projecte
  - Accions per del projecte:
    - Visualitzar pàgines
    - Visualització de la informació
    - Editar
    - Eliminar

  També ha d'estar disponibles un enllaç a crear un nou projecte. El nombre de projectes llistats s'incrementa molt, s'afegirà un paginador per mostrar les dades d'una forma mes usable.

- Edició

  Aquesta vista serveix per la creació de nous projectes i per l'edició de les dades de projectes ja donats d'alta. La vista ha d'estar constituïda per un formulari. La informació que ha de mostrar el formulari es:

  - Títol (obligatori): Camp de text
  - Descripció (obligatori): Caixa de text

- Autors: Caixa de text
- Ruta de repositori: Camp de text

Aquest últim determinarà si a les pàgines d'aquest projecte es podran afegir elements de repositori.

- Visualització

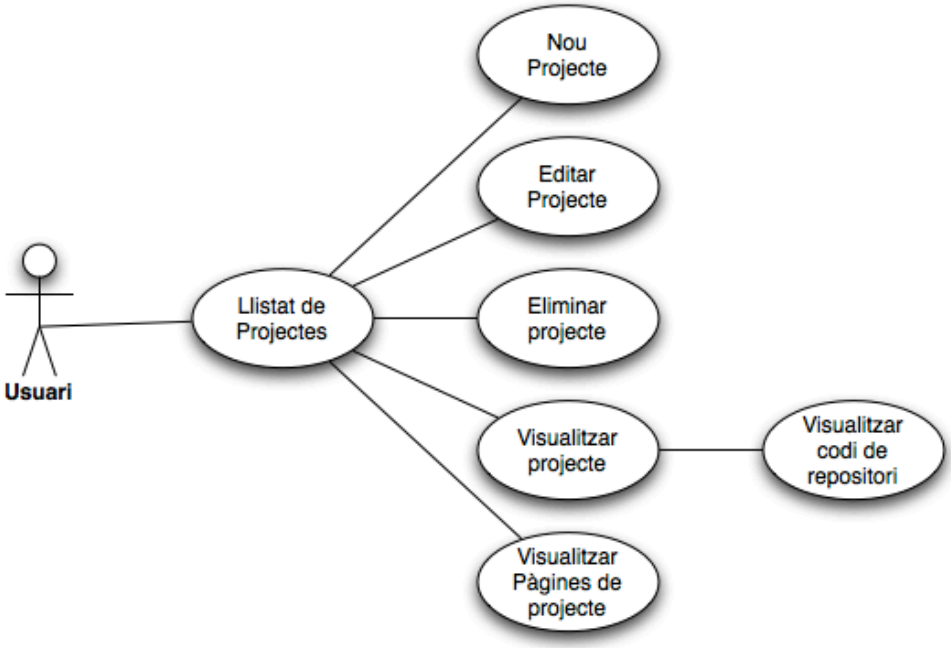
  La vista de visualització ha de mostrar la informació introduïda al formulari. Les dades mostrades en aquesta vista son:

  - Títol
  - Descripció
  - Autors
  - Enllaç a visualització de repositori

3.1.1 Consulta de repositori

Un projecte s'ha de poder relacionar amb un element d'un repositori gestor de versions des de el formulari. Els elements del repositori s'han de poder consultar i visualitzar. Mentre es consulta la informació del repositori sempre ha d'estar disponible la opció de tornar al projecte.

3.1.2 Cas d'ús



3.2 Secció Gestor de Pàgines

El gestor de pàgines serveix per gestionar les pàgines d'un projecte. El gestor ha de mostrar totes les pàgines relacionades amb el projecte que s'està consultant.

Aquest gestor ha de disposar de 3 visualitzacions:

- Llistat

La visualització de llistat ha de mostrar totes les pàgines de pàgines que s'han creat per el projecte en el qual s'ha entrat. El llistat ha de mostrar la següent informació:

- Títol de pàgina
- Accions
    - Visualitzar/editar
    - Eliminar

- Creació

La vista de creació ha de ser un formulari per determinar si es vol crear la pàgina o no. Aquest formulari tindrà un sol camp de text obligatori pel títol de la pàgina. Un cop creada la pàgina l'aplicació es redirigirà a l'usuari a la vista de edició i visualització. L'adreça de la pàgina es composarà l'adreça de la pàgina amb el títol de la pàgina.

- Edició i visualització

La vista d'edició i visualització ha de ser una vista molt dinàmica donat que per defecte la vista ha de ser la de visualització. Aquesta vista mostra els diferents continguts de la pàgina de forma ordenada i amigable. La vista d'edició ha de ser els diferents formularis de cada porció de contingut de la pàgina.

Per poder registrar el mes acuradament una funcionalitat o codi de programació a una pàgina aquesta vista ha de permetre guardar contingut variat. Les porcions de contingut s'han de poder afegir sense recarregar la pàgina i de manera unitària. De manera que un canvi en la descripció no pot afectar a codi de programació.

3.2.1 Contingut de pàgina

Diferent contingut que ha de poder guardar una pàgina ha de ser el següent:

3.2.1.1 Text

El contingut tipus text ha de permetre visualitzar el text amb format. Donat que una eina orientada a desenvolupadors els formats de text tipus negreta, capçalera o cursiva han de poder ser inserits per marques al text. La raó de fer servir aquest tipus d'editora facilitar una edició àgil sen se hagué de recorre a botons. Com els usuari de l'aplicació seran desenvolupadors, aquests ja tenen la facilitar per relacionar marcatges al text per donar un sentit determinat, una edició amb aquest tipus d'eina es molt mes fàcil i ràpid.

3.2.1.2 Codi

Codi com a element principal de la reutilització de codi s'ha de poder visualitzar correctament. Per això es necessari disposar d'un ressaltat de codi que faci mes fàcil la visualització. En la edició d'aquest tipus d'informació ha de permetre indicar quin llenguatge de programació es mostra. El llistat de llenguatges de programació acceptats ha de ser com a mínim 15 llenguatges.

3.2.1.3 Elements de repositori

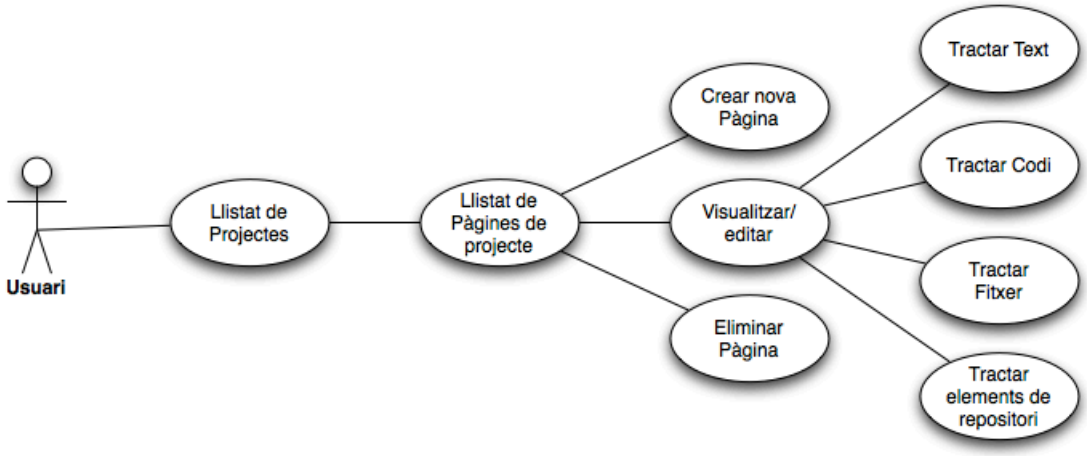
Els elements de repositori es una millora respecte els requisits inicials. Aquesta opció només s'ha de mostrar si el projecte al qual pertany la pàgina que s'està editant s'ha vinculat amb un repositori d'un sistema gestor de versions. Per afegir elements de repositori s'ha de permetre navegar pel contingut del repositori i afegir la visualització a la pàgina. La navegació pel contingut del repositori sempre serà per el contingut mes actualitzat i per la branca principal. Aquests elements no seran modificables donat que es guarden en un estat del repositori.

3.2.1.4 Fitxers

El contingut de les pàgines ha de permetre afegir fitxers, tant imatges com documents i fitxers. Els fitxers s'han de poder pujar des de un formulari dins de la mateixa web. Al guardar el formulari aquest ha de mostrar una previsualització en cas de ser una imatge o un enllaç per descarregar el fitxer o

document. Per permetre identificar el fitxer pujat el formulari ha de disposar d'un camp per afegir-hi títol. Tots els fitxers seran emmagatzemats a l'aplicació.

3.2.2 Cas d'ús



3.3 Secció Gestor d'usuaris

El gestor d'usuaris es la secció de l'aplicació on es podran donar d'alta nous usuaris, editar els ja creats i desactivar els que ja no faran servir l'aplicació.
El gestor d'usuaris ha de tindre 2 visualitzacions:

- Llistat

La visualització de llistat ha de mostrar tots els usuaris que s'han creat. El llistat ha de mostrar la següent informació:

- Nom, Cognoms
- Correu electrònic
- Està actiu?
- Es administrador?
- Accions (Editar, Eliminar)

- Edició

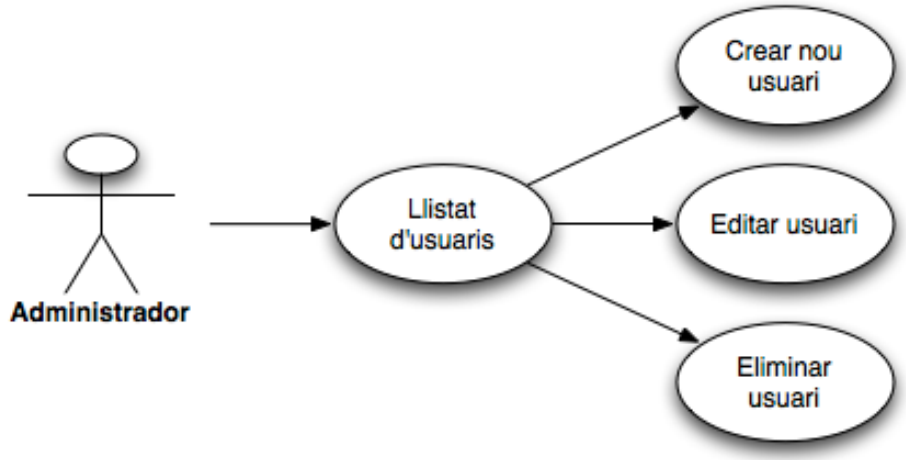
La visualització d'edició serveix per la creació de nous usuaris i per la modificació dels usuaris existents. Aquesta vista mostrarà un formulari per permetre la modificació de les dades. Les dades del formulari han de ser:

- Nom
- Cognoms
- Correu electrònic
- Contrasenya
- Confirmació de contrasenya
- Està actiu?
- Es administrador?
- Telèfon
- Adreça
- Aniversari

3.3.1 Perfils d'usuari

Hi hauran 2 perfils d'usuari, el perfil d'administrador i el perfil d'usuari. Els usuaris disposaran d'accés a totes les seccions menys a la de gestió d'usuaris. La secció d'usuaris estarà restringida únicament als perfils administradors.

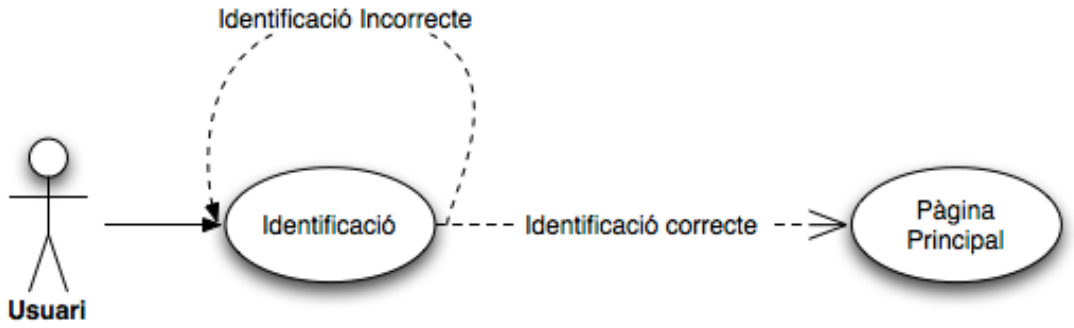
3.3.2 Cas d'ús



3.4 Secció d'identificació

Aquesta secció serveix per accedir a l'aplicació amb el nom d'usuari (correu electrònic) i la paraula clau. La informació per accedir a l'aplicació la ha de facilitar l'administrador. Si un usuari no registrat intenta accedir a una plana o secció, aquest serà redirigit a la secció d'identificació. Quan un usuari s'ha identificat a l'aplicació l'ha de redirigir a la plana principal. En el cas de que un usuari que ja s'ha identificat intenti accedir a la secció d'identificació, aquest serà redirigit a la plana principal.

3.4.1 Cas d'ús



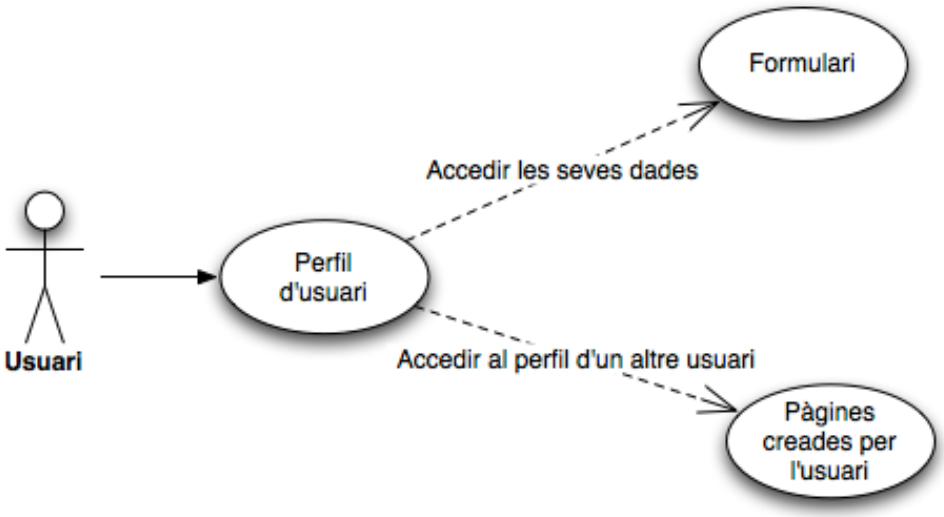
3.5 Secció d'usuari

Un usuari ha de poder gestionar la seva informació personal. Per això ha d'existir una secció on hi hagi un formulari on un usuari pot gestionar les seves dades. La única dada no modificable si no es administrador es el nom d'usuari.

En el cas de que un usuari no administrador intenti accedir a la secció d'usuari d'un altre usuari, no ha de mostrar el formulari i només les pàgines que aquest ha creat.

En aquesta secció després del formulari de modificació de dades s'ha de mostrar el llistat de pàgines creades per l'usuari.

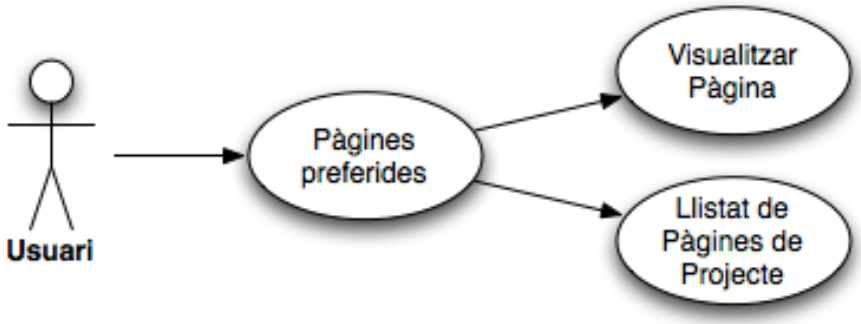
3.5.1 Cas d'us



3.6 Secció de pàgines preferides

Cada pàgina podrà ser afegida com a pàgina preferia. Aquesta opció proporciona als usuaris guardar les pàgines amb un contingut mes rellevant pel seu treball. La secció de pàgines preferides mostrarà el llistat de les pàgines que un usuari ha marcat com a preferida. Al llistat es mostrarà el títol de la pàgina i el projecte al que pertany, tots dos sent enllaços a la seva vista.

3.6.1 Cas d'ús

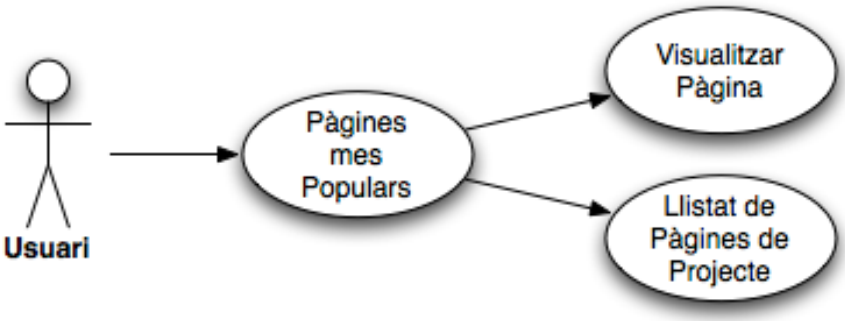


3.7 Secció de pàgines mes populars

La secció de pàgines mes populars ha de mostrar un llistat de les 20 pàgines mes afegides com a preferides. D'aquestes pàgines s'ha de mostrar el següent contingut:

- Títol de la pàgina sent un enllaç a la pàgina
- Número de vegades que s'ha afegit com a favorita
- Nom del projecte al que pertany la pàgina sent un enllaç al projecte

3.7.1 Cas d'ús

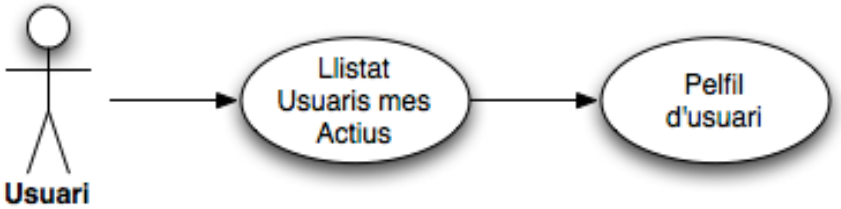


3.8 Secció d'usuaris mes actius

La secció d'usuaris mes actius ha de mostrar un llistat amb els noms dels usuaris que mes pàgines han creat. El llistat ha de mostrar la següent informació:

- Nom d'usuari sent un enllaç a la secció d'aquest usuari
- Número de pàgines creades

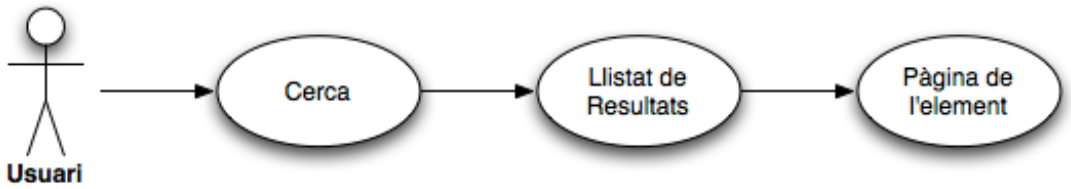
3.8.1 Cas d'ús



3.9 Secció de resultats de cerca

La secció de resultats de cerca ha de mostrar elements de l'aplicació que tenen relació com el contingut enviat. Cada element del resultat ha d'enviar te a la seva pàgina.

3.9.1 Cas d'ús



3.10 Trets comuns a les seccions

Les seccions esmentades i descrites disposaran d'un conjunt de funcionalitats i enllaços. Aquestes funcionalitats i característiques comunes proporcionen a la interfície mes contingut i usabilitat. El següent llistat mostra els diferents trets comuns:

Un usuari ha d'estar correctament identificat per accedir a una secció, en cas contrari serà redirigit a la secció d'identificació.

Quan un usuari s'ha identificat, la identificació serà persistent per mitjà de sessions per la resta de seccions. Una sessió només es tancarà en el cas de que l'usuari es desconnecti o es netegi les dades del navegador.

Sempre ha d'estar disponible un enllaç al llistat de projectes.

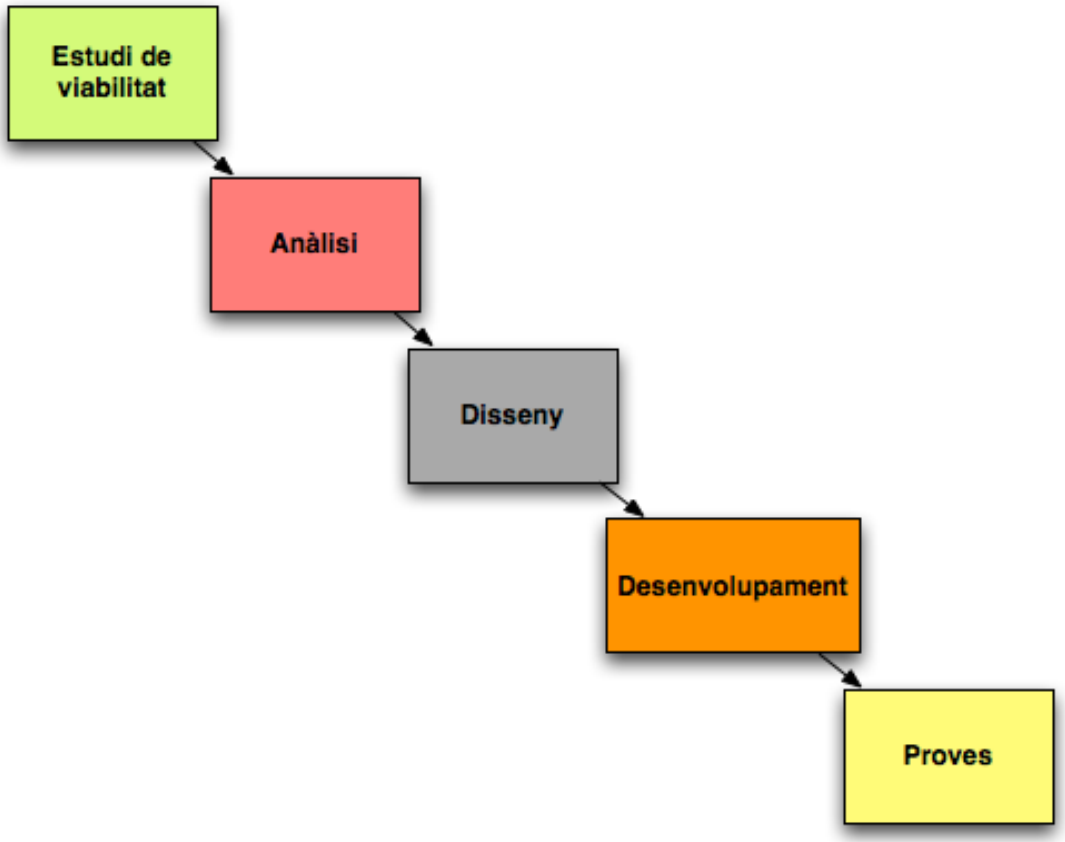
La secció de pàgines ha de mostrar en tot moment un enllaç al projecte sobre el qual s'està treballant.

L'accés a la secció d'usuari ha d'estar sempre disponible per mitjà d'un enllaç amb el nom de l'usuari.

3.11 Model de desenvolupament

El model de desenvolupament escollit per el projecte es el model seqüencial o de salt d'aigua.

Al model de desenvolupament seqüencial es descriuen per fases les diferents etapes de requerides per la creació de l'aplicació. A cada fase es descriuen diferents característiques que serveixen per la creació de la nova fase.



3.12 Arquitectura de software

L'arquitectura de software defineix de manera abstracta el nivell mes alt de l'estructura del sistema. L'arquitectura dona a conèixer els components que fan alguna tasca de computació, les seves interfases i la comunicació entre elles.

Les arquitectures més universals son:

- **Monolítica:** on el software s'estructura en grups funcionals molt units.
- **Client-Servidor:** el software fa un repartiment de la computació en dos parts independents, però sense un repartiment clar de les funcions.
- **3 nivells:** es una especialització de la arquitectura client-servidor on la carrega es reparteix en 3 parts o capes. L'arquitectura a 3 nivells fa un clar repartiment de les funcions: una capa de presentació (interfície d'usuari), una altre pel càlcul i una última per l'emmagatzematge. Entre les capes només hi ha una relació amb l'adjacent.

El sistema a desenvolupar farà servir una arquitectura de 3 nivells, donat que es ideal per projectes web. El primer nivell presentarà el codi HTML que mostra el contingut. El segon nivell atendrà les peticions del navegador i actuarà en conseqüència. El tercer i últim nivell guardarà relació amb les bases de dades on serà emmagatzemada la informació.

3.13 Tecnologia i software

Per la creació del sistema es faran servir les tecnologies pròpies de d'una aplicació web HTML, javascript i peticions get, post, put, i delete. Per el tractament de dades i el control del sistema de 3 capes es farà servir el framework Ruby on Rails.

Ruby on Rails es un framework de desenvolupament de software lliure fet amb el llenguatge de programació Ruby, seguint l'arquitectura Model, Vista i Controlador. Els principis fundamentals de Ruby on Rails inclouen no et repeteixis (en anglés DRY: Don't Repeat Yourself) i convenció per sobre de la configuració.

Prototype el framework de software lliure de codi javascript el qual ve per defecte al instal·lar el framework Ruby on Rails i per això serà el que es farà servir en el sistema.

Prototype proporciona eines per la realització de peticions Ajax i tractament del DOM de la pàgina web.

Les dades de l'aplicació seran guardades en una base de dades SQL dins d'un Sistema Gestor de Bases de Dades. El framework Ruby on Rails proporciona 3 entorns Desenvolupament, Test i Producció i per cada entorn es fa servir una base de dades. Per els entorns de desenvolupament i test es farà servir el sistema gestor de base de dades SQLite que proporciona un gran facilitat d'ús sense cap configuració. Per l'entorn de producció es farà servir el sistema gestor de base de dades MySQL que proporciona un sistema mes potent i amb mes opcions de configuració. Els dos sistema de gestor de base de dades escollits son gratuïts i de llicència lliure.

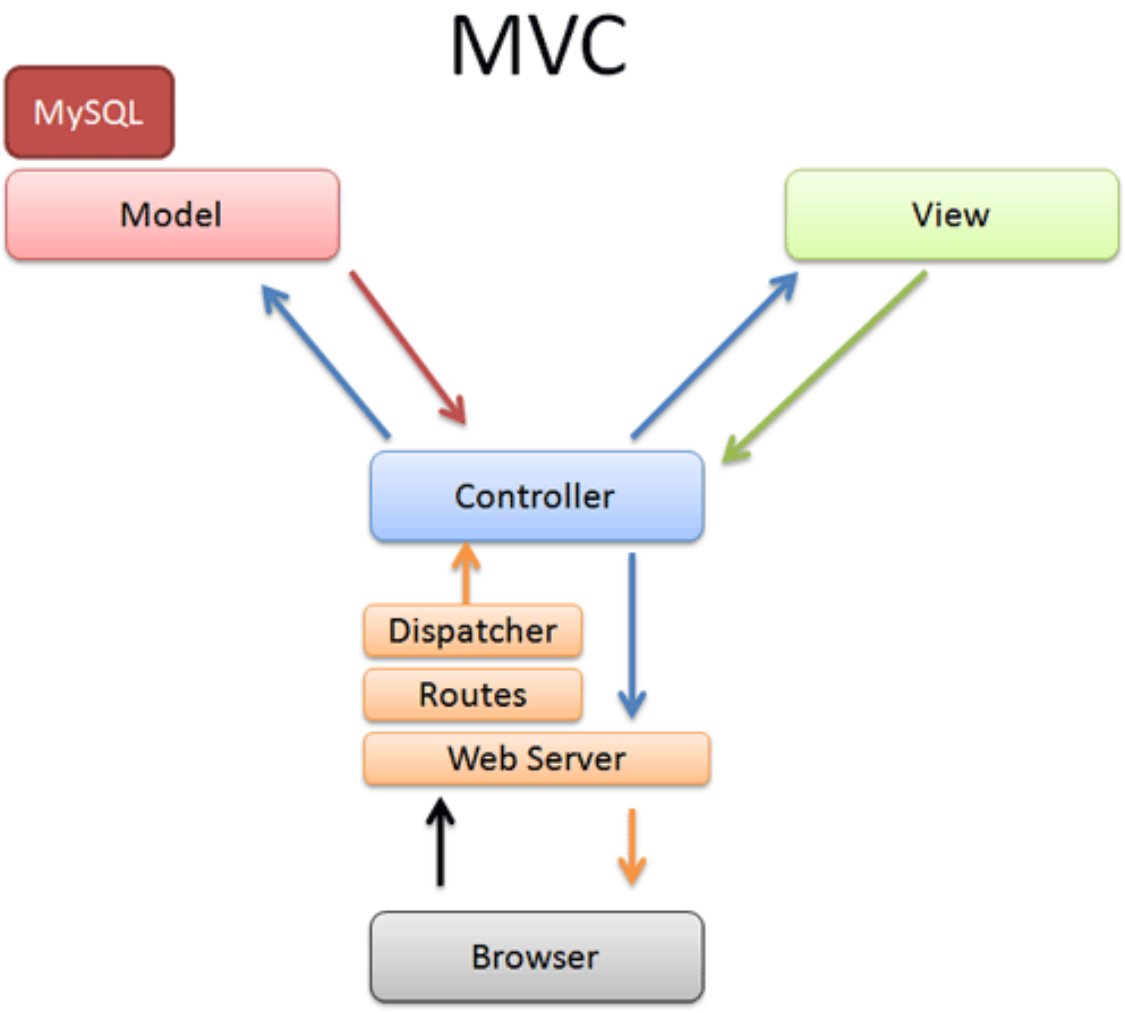
Per la creació del sistema es faran servir les següents eines de codi lliure:

- Mozilla Firefox 3
- Vim 7
- Terminal

4 Disseny del sistema

4.1 Model-Vista-Controlador

Per la creació del sistema de reutilització de codi es farà servir una arquitectura de 3 capes. Les capes que formaran part del sistema seran Vista, Model i Controlador, aquesta arquitectura sol ser anomenada per la abreviació MVC que significa: Model, View and Controller.



La capa de Vista es la que s'encarrega de mostrar les dades recollides del controlador. Les dades son mostrades per mitjà del fitxers que contenen el codi HTML i les referències als diferents recursos. La capa de vista proporciona diferents elements per llençar esdeveniments per interactuar amb l'aplicació

La capa de Controlador es la encarregada de recollir els esdeveniments del navegador llençats des de la capa de Vista. La capa de Controlador identifica l'esdeveniment i executa la acció corresponent. Segons l'acció pot només mostrar un a vista en concret o fer una petició al model per la recol·lecció de dades a mostrar a la capa de vista.

La capa de Model es la encarregada de tractar les dades i les seves relacions. El model afegeix una capa d'abstracció sobre una taula de la base de dades i defineix un objecte amb el atributs de la taula. Es la capa de model on es defineixen les diferents relacions entre taules com d'un a molts, d'un a un o de molts a un. Si el controlador fa una petició de dades sobre una taula, es el model l'element que s'encarrega de recol·lectar els resultats i retornar-los en forma de vector.

La arquitectura de Model, Vista i Controlador es molt útil per aplicacions web on els Models encapsulen les diferents taules de la base de dades, el controlador atén les peticions GET, PUT, POST i DELETE al servidor i envia al navegador d'internet les diferents vistes HTML amb el contingut demanat.

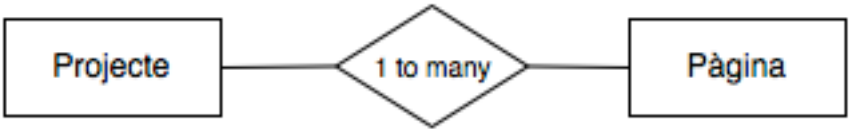
4.2 Disseny del Gestor de Projectes

El Gestor de projectes permet gestionar els diferents projectes del sistema de reutilització de codi. La taula que ens permet guardar les dades del projectes té el nom de Projectes i conté la següent informació:

4.2.1 Model Projecte

Nom del atribut	Tipus
Id	Integer
Nom	String
Descripció	Text
Autors	Text
url_slug	String
Data_creació	Datetime
Data_actualització	Datetime
Git_path	String

El següent diagrama d'entitat relació mostra les diferents relacions del model Projecte amb la resta de models.



4.3 Disseny del Gestor de Pàgines

El gestor de pàgines serveix per gestionar les diferents pàgines on serà guardada la informació del codi de programació. La taula Pagina emmagatzema les dades segons el següent quadre:

4.3.1 Model Pàgina

Nom del atribut	Tipus
Id	Integer
Títol	String
Project_id	Integer
url_slug	String
Data_creació	Datetime
Data_actualització	Datetime
Git_path	String

Per gestionar el diferent contingut de les pàgines es farà servir les següent taules:

4.3.1.1 Model slot de pàgina

Nom del atribut	Tipus
Id	Integer
Pàgina_id	Integer
Tipus_objecte_relacionat	String
Id_objecte_relacionat	Integer
Data_creació	Datetime
Data_actualització	Datetime
Posició	Integer

4.3.1.2 Model anotació de text

Nom del atribut	Tipus
Id	Integer
Titol	String
Contingut	Text
Data_creació	Datetime
Data_actualització	Datetime

4.3.1.3 Model anotació de codi

Nom del atribut	Tipus
Id	Integer
Llenguatge	String
Contingut	Text
Data_creació	Datetime
Data_actualització	Datetime

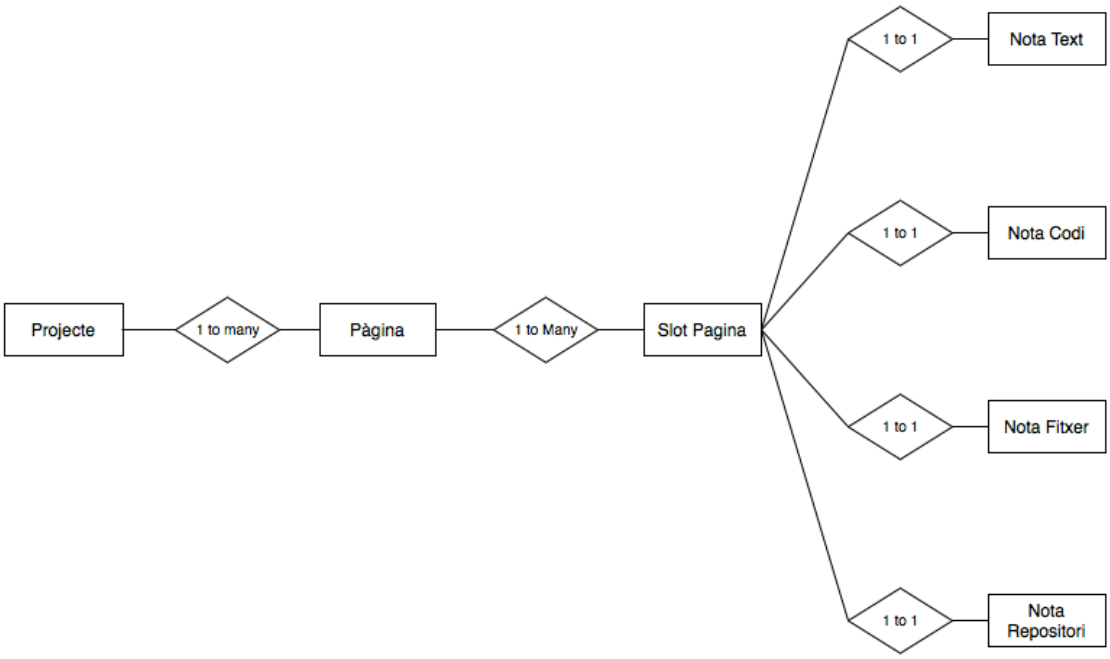
4.3.1.4 Model anotació de fitxer

Nom del atribut	Tipus
Id	Integer
Titol	String
Nom_fitxer	String
Tipus_contingut	String
Tamany_fitxer	Integer
Data_actualització_fitxer	Integer
Data_creació	Datetime
Data_actualització	Datetime

4.3.1.5 Model anotació d'element de repositori

Nom del atribut	Tipus
Id	Integer
Titol	String
Git_id	String
Git_tipus	String
Contingut	Text
Llenguatge	String
Data_creació	Datetime
Data_actualització	Datetime

El següent diagrama mostra les diferents relacions dels models descrits



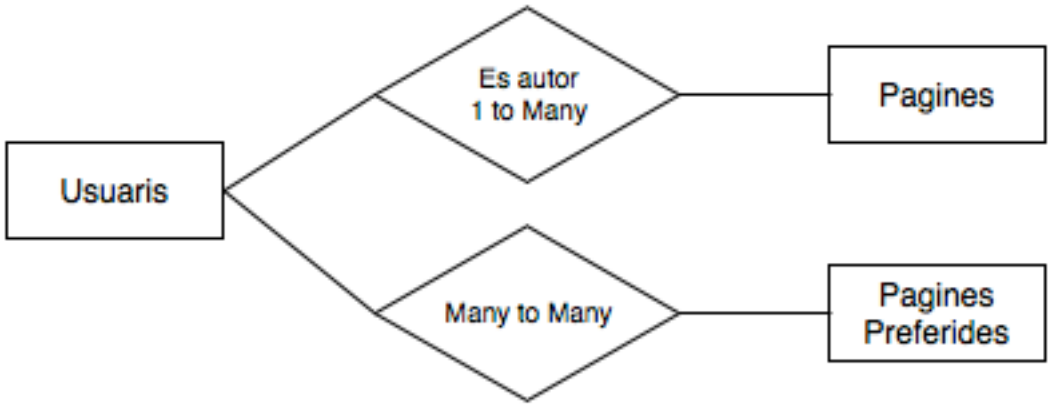
4.4 Disseny de Gestor d'usuaris

El gestor d'usuaris guarda tota la informació referent als usuaris del sistema. Les accions con registre, identificació o canvi de dades agafen les dades de la taula Usuaris.

4.4.1 Model usuari

Nom del atribut	Tipus
Id	Integer
Nom	String
Cognoms	String
Email	String
Password_encryptada	String
Salt	String
Es_actiu	Boolean
Es_administrador	Boolean
Telefon	Integer
Adreça	String
Data_naixement	Date
Data_creació	Datetime
Data_actualització	Datetime

El següent esquema mostra les diferents relacions amb la taula Usuaris.



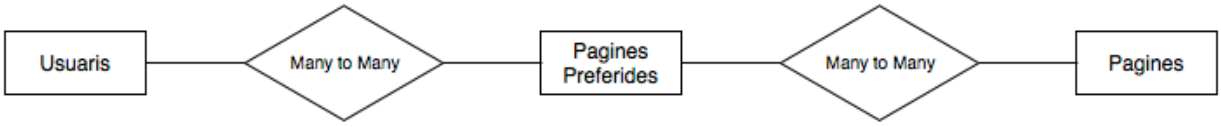
4.5 Disseny de Pàgines Preferides

Pàgines preferides es l'apartat del sistema on un usuari pot guardar pàgines amb anotacions per ser visualitzades mes tard. Aquest apartat té el seu suport a una taula intermitja amb el següent esquema.

4.5.1 Model Pàgina Preferida

Nom del atribut	Tipus
Id	Integer
Pàgina_id	Integer
Usuari_id	Integer
Data_creació	Datetime
Data_actualització	Datetime

El següent esquema mostra les diferents relacions amb l'anterior taula

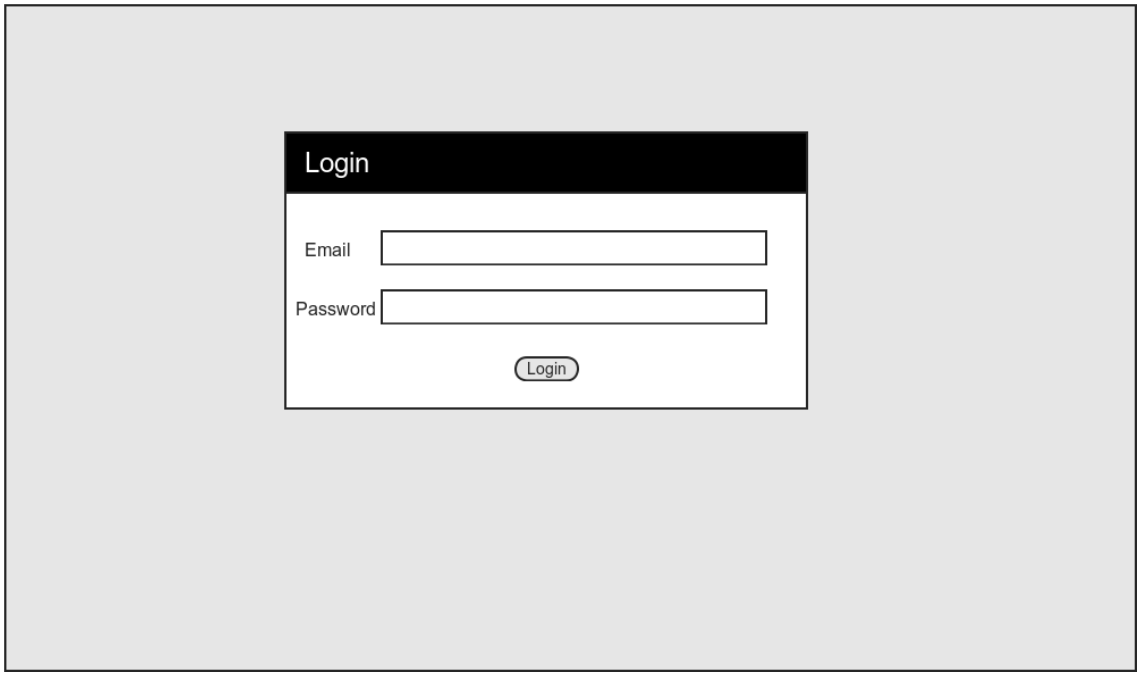


4.6 Interfícies del sistema

Els següents apartats descriuran l'aspecte del sistema i la estructuració de de les dades. Per descriure els diferents apartats es realitzaran esquemes amb un estil que proporcioni una idea general de l'aspecte final de l'aplicació.

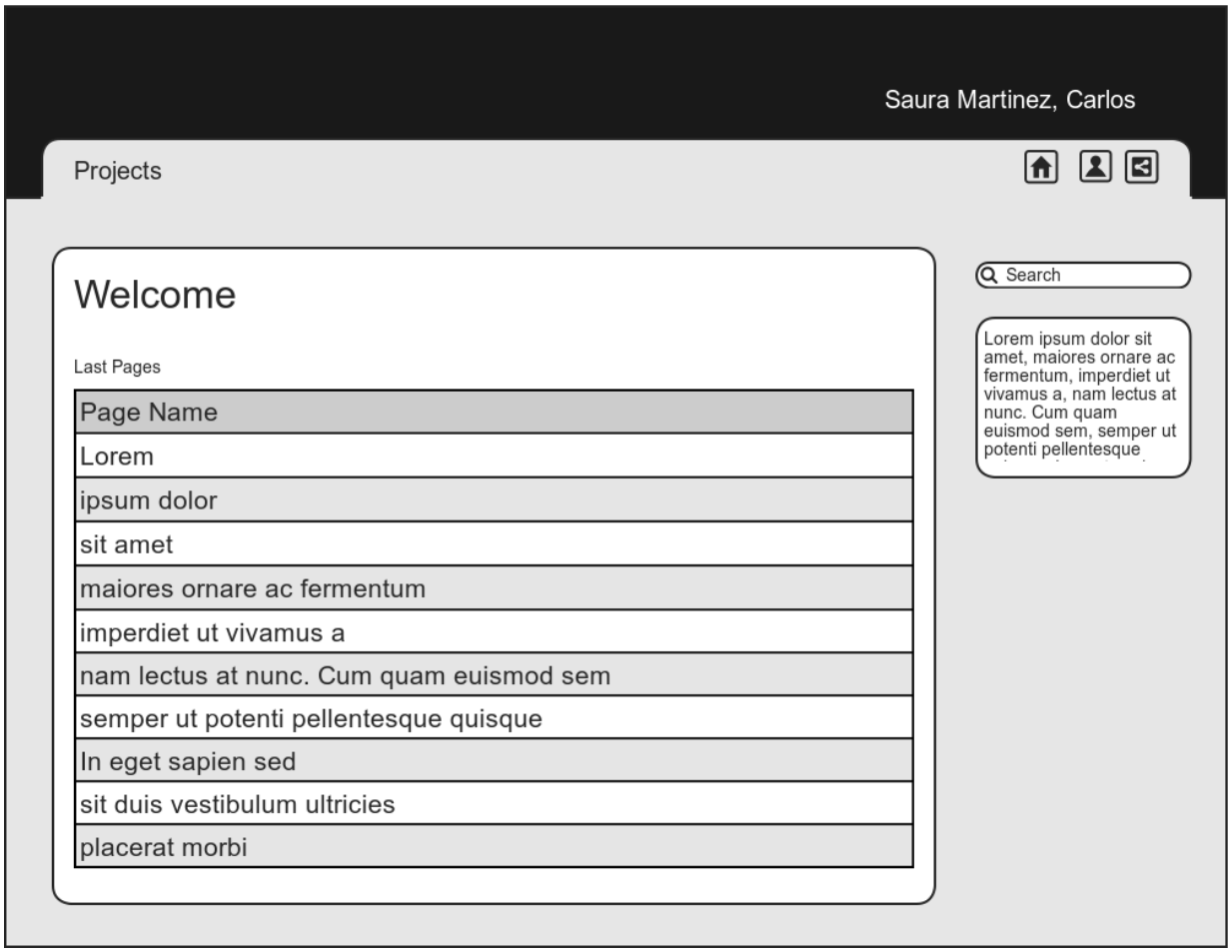
4.6.1 Identificació

El següent diagrama mostra la pàgina d'accés a l'aplicació.



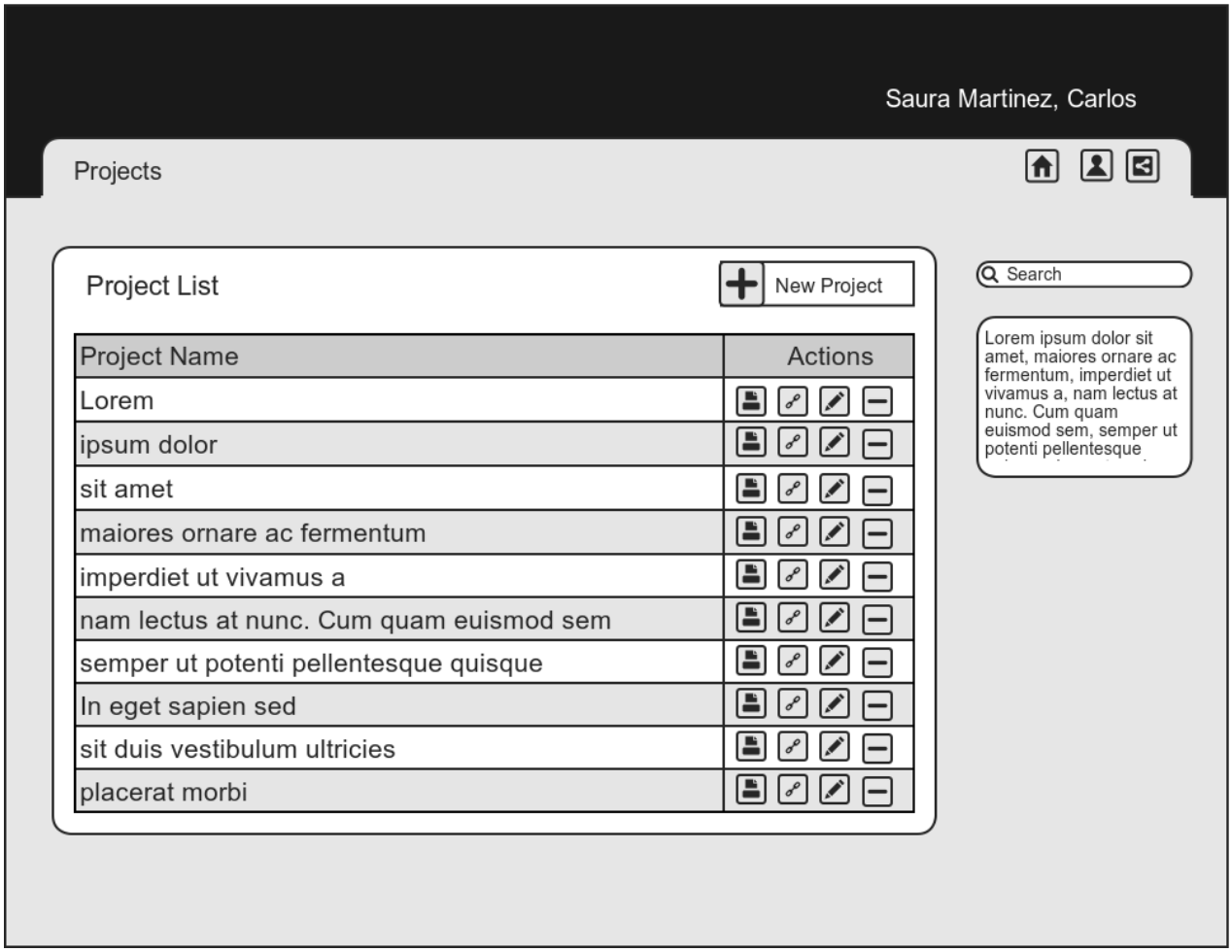
4.6.2 Pàgina Principal

El següent esquema mostra quina es la primera pàgina que es mostra al accedir a l'aplicació. Es podrà visualitzar el nom de l'usuari connectat i les últimes pàgines creades a sistema.



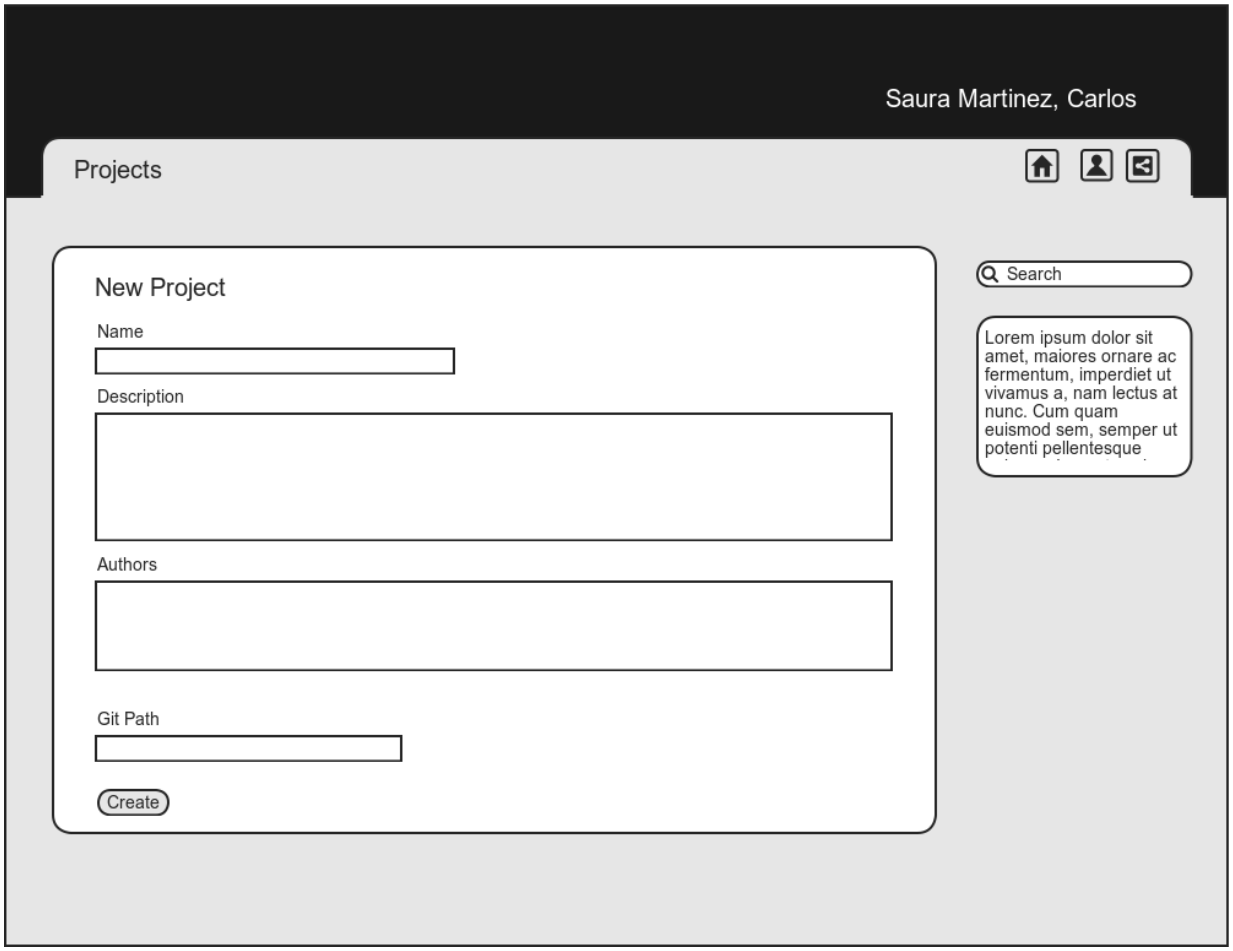
4.6.3 Llistat de Projectes

El mostra els diferents projectes creats al sistema i les accions disponibles. Les accions son veure pàgines de projecte, veure projecte, editar projecte, eliminar projecte i crear projecte nou.



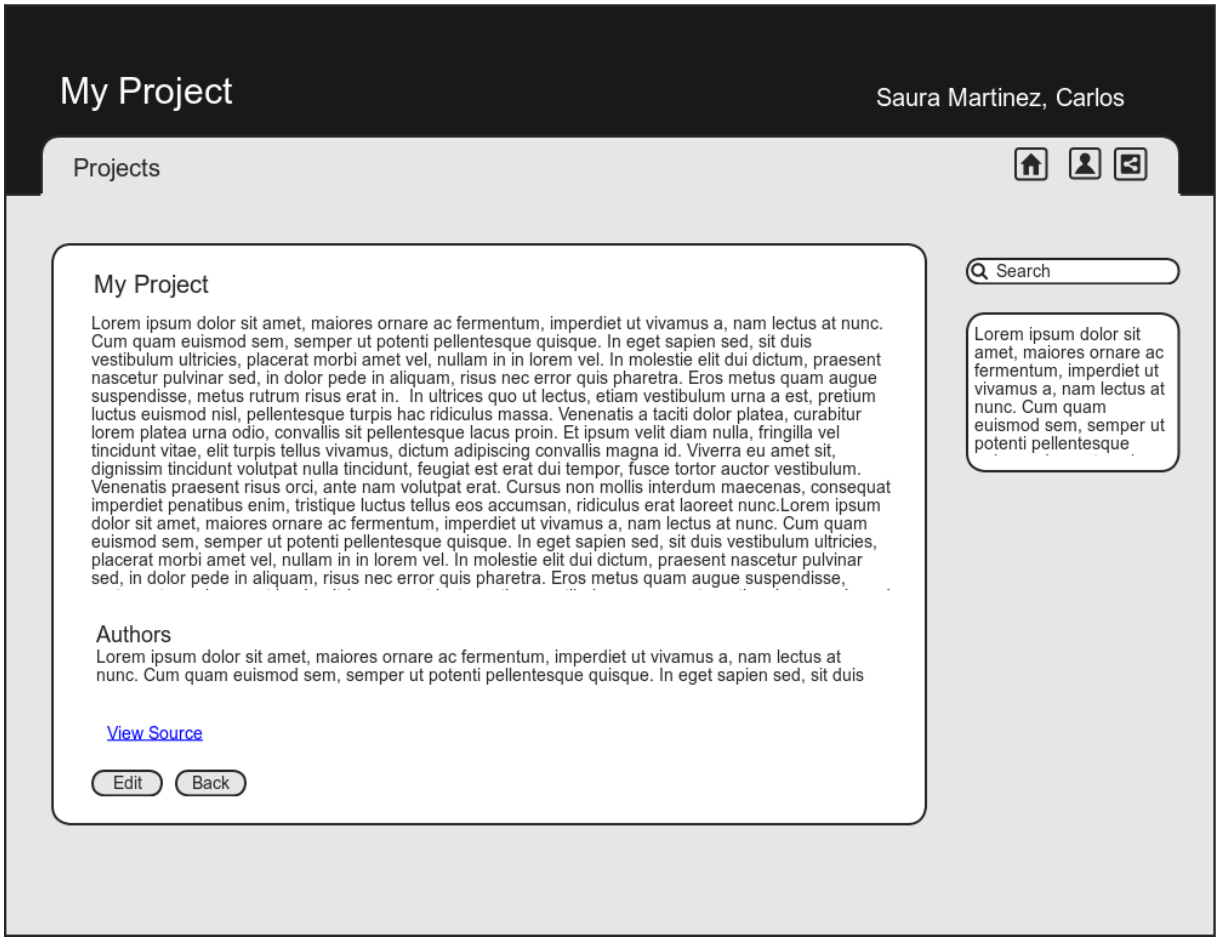
4.6.4 Formulari de creació i edició de projecte

El següent esquema mostra quin aspecte tindrà el formulari de creació i edició de projectes. A la barra lateral es mostrarà ajuda sobre el formulari.



4.6.5 Visualització de Projecte

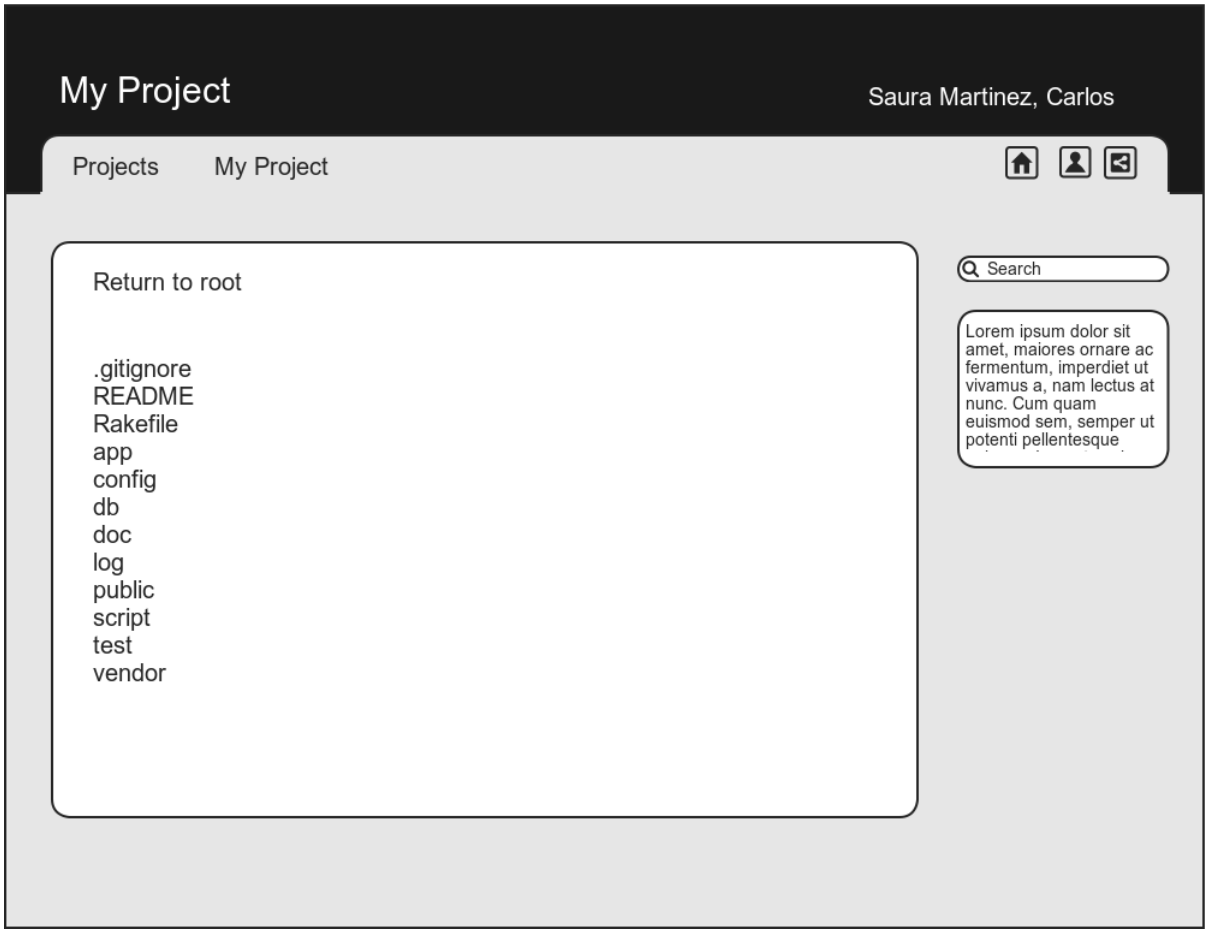
La visualització del projecte mostra la informació introduïda sobre el projecte i la possibilitat de visualitzar el contingut del repositori GIT.



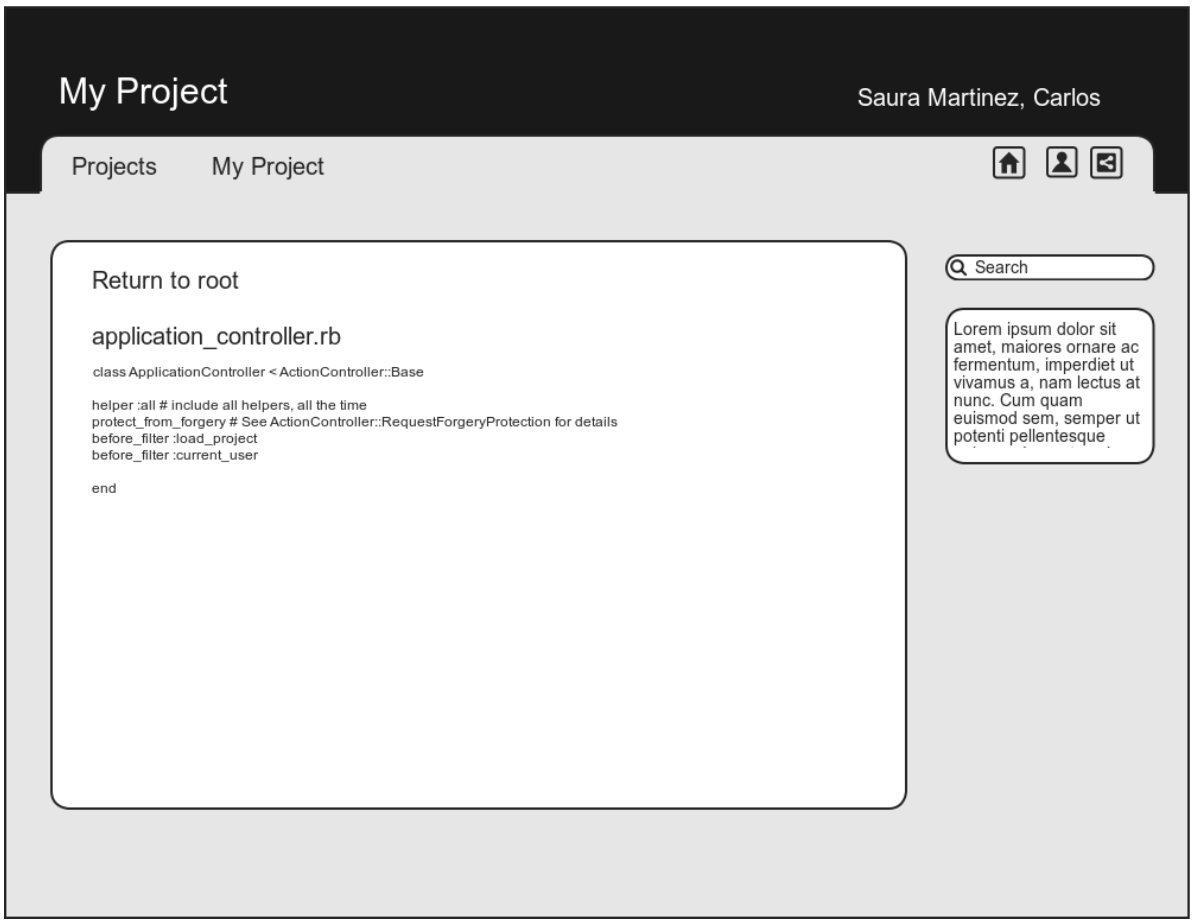
4.6.6 Visualització de codi de repositori d'un projecte

El següent diagrama mostra la informació del repositori de codi en GIT en cas del projecte tingui repositori.

Visualització d'un directori

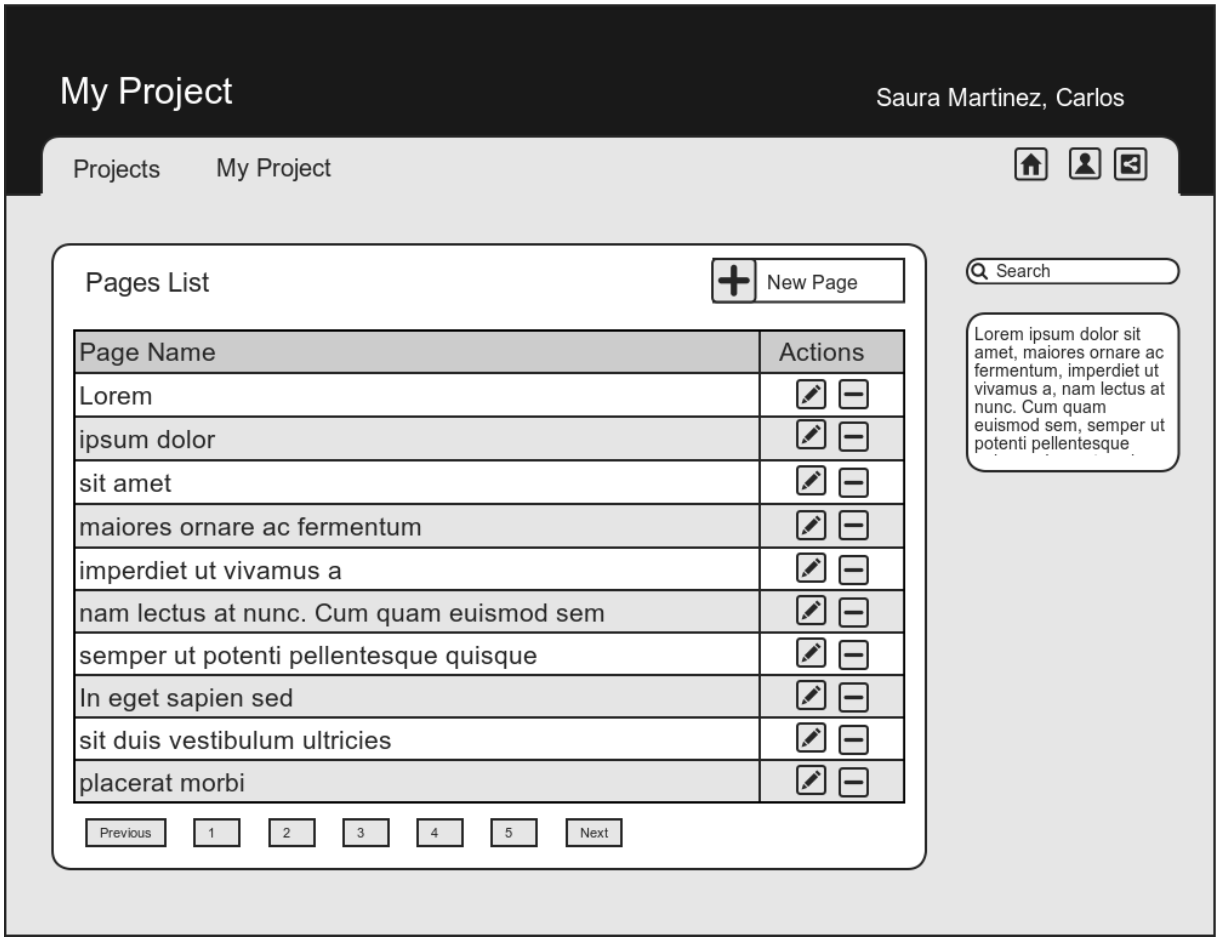


Visualització d'un fitxer



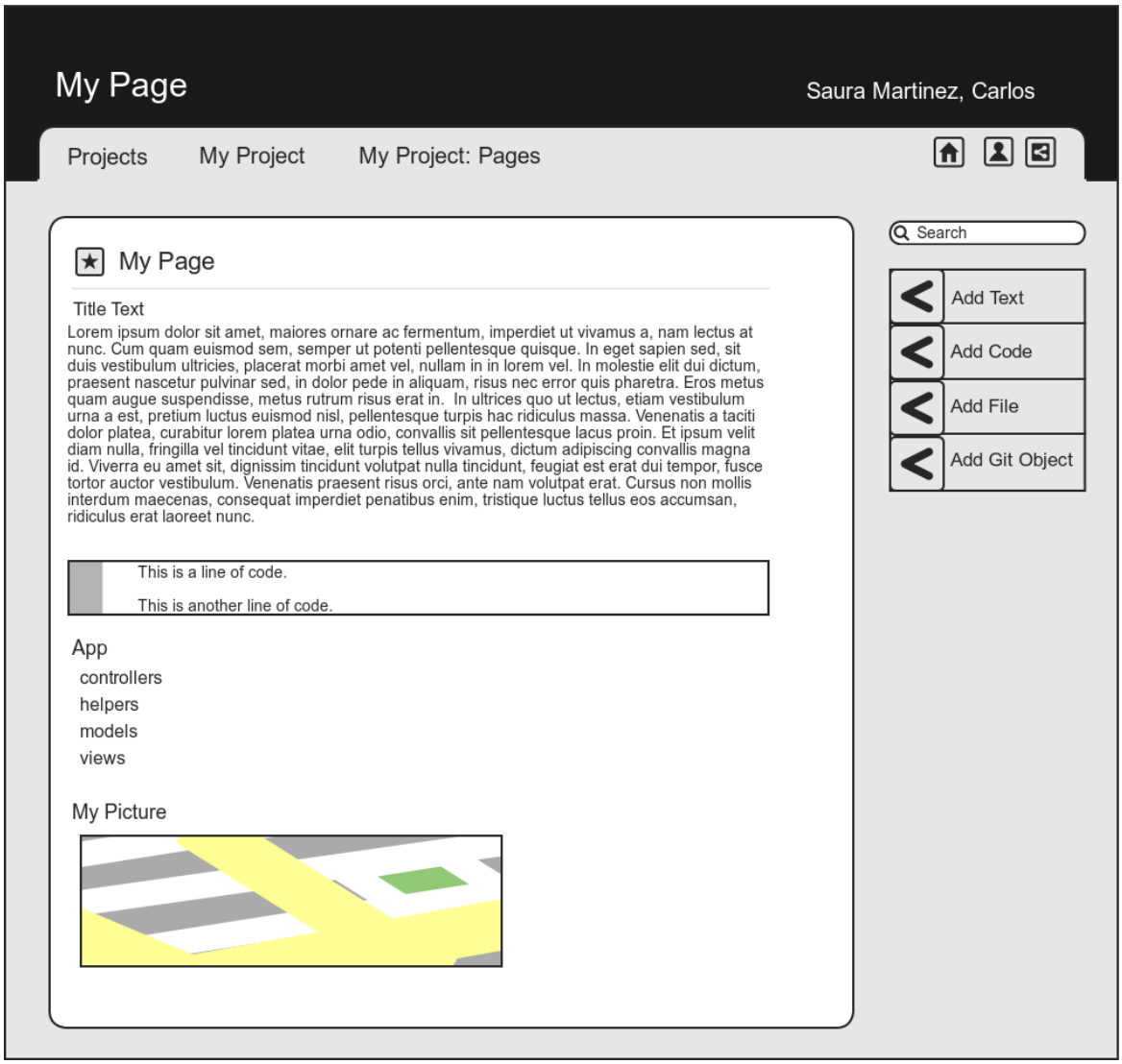
4.6.7 Llistat de Pàgines de projecte

El llistat de pàgines de projectes mostra totes les pàgines que pertanyen a un projecte. Dona la opció de crear una nova pàgina pel projecte, visualitzar una pàgina, editar-la i eliminar-la.

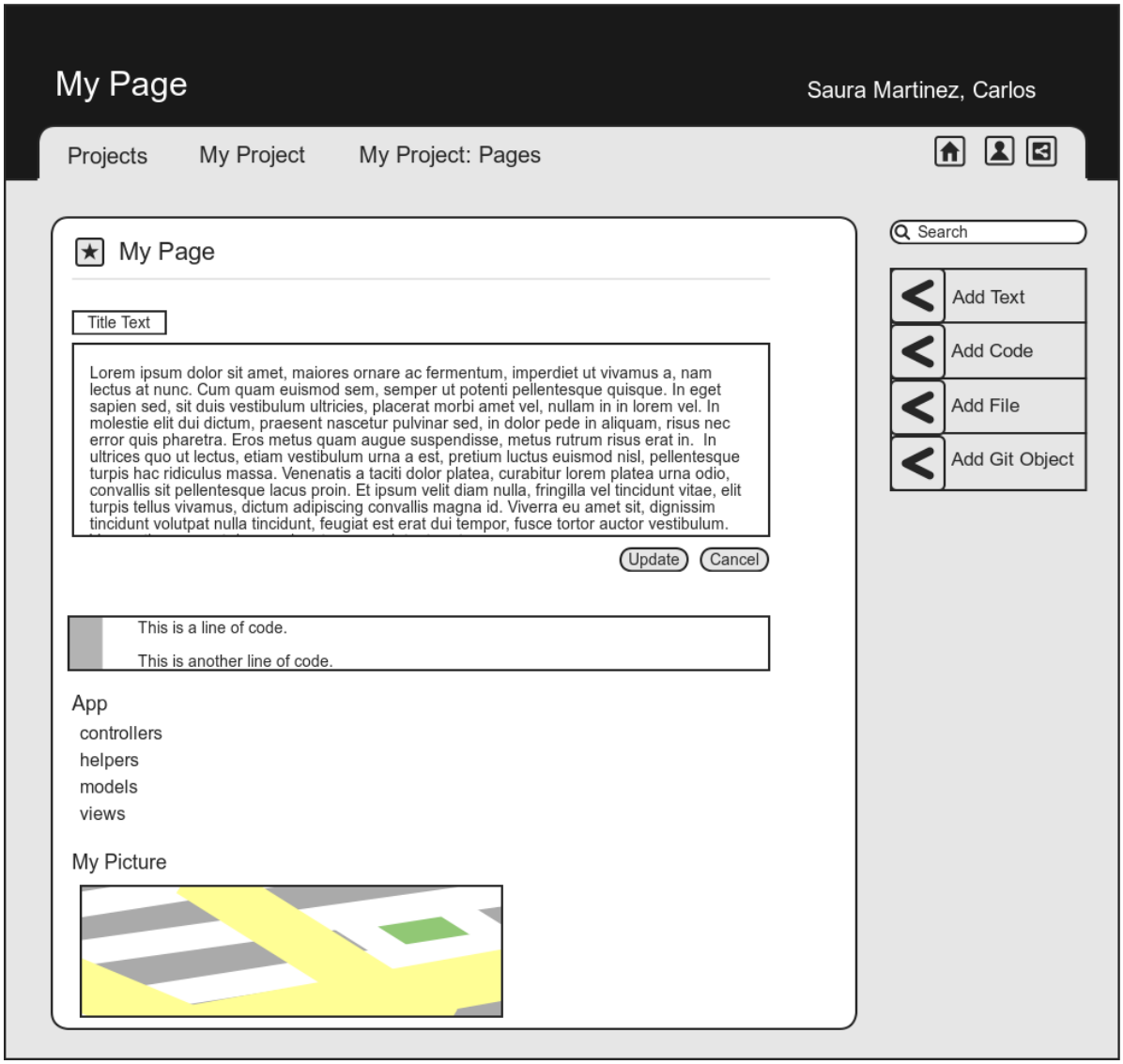


4.6.8 Visualització de Pàgina

El següent esquema mostra la informació introduïda d'una pàgina. Dona la opció d'afegir una anotació de text, codi, fitxer o objecte de repositori a la pàgina.

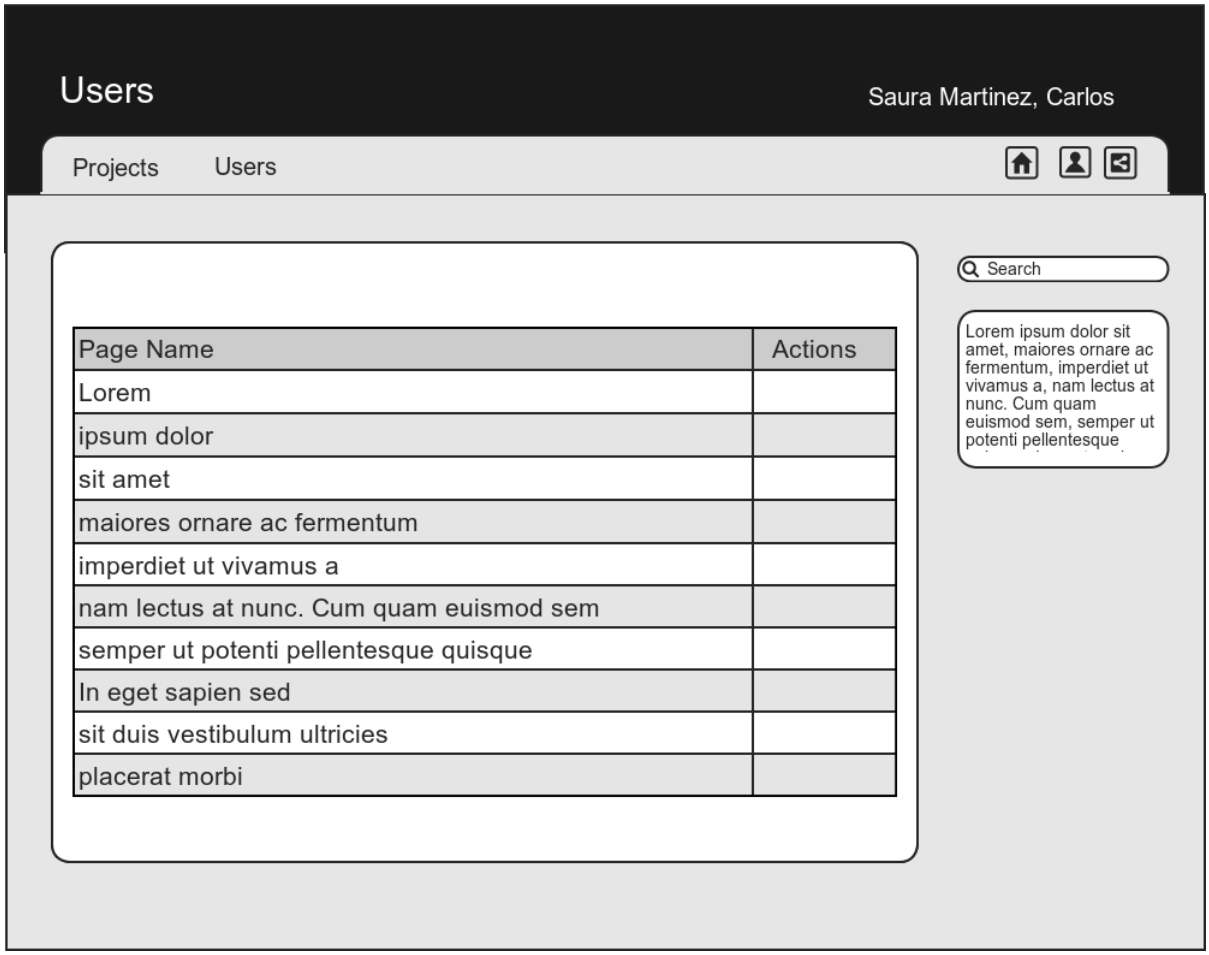


El mateix esquema anterior però amb l'edició de l'anotació per text via AJAX.



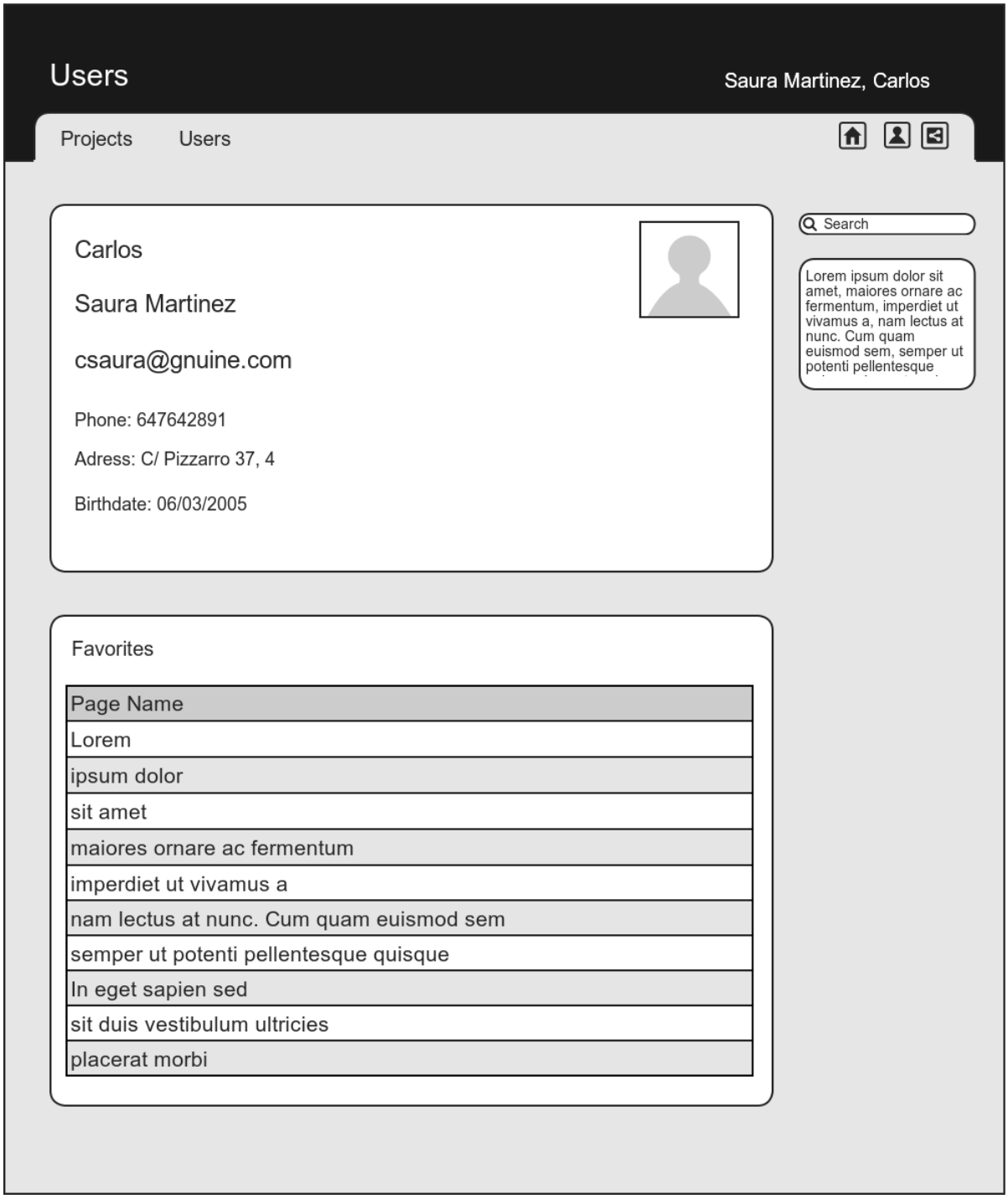
4.6.9 Llistat d'usuaris

El següent esquema mostra com es mostrarà la informació del llistat d'usuaris del sistema.



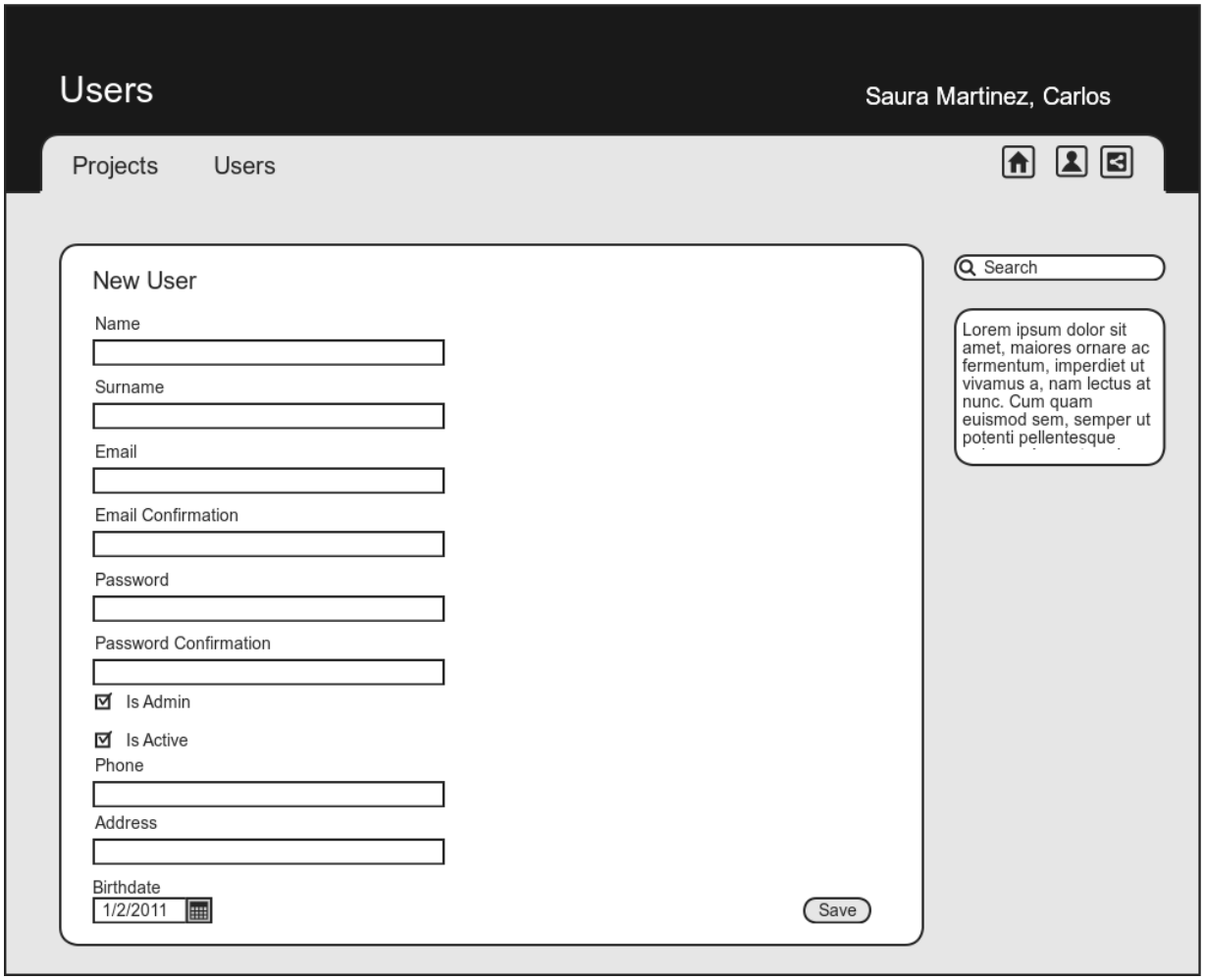
4.6.10 Visualització de les dades d'un usuari

El següent esquema mostra la informació introduïda sobre un usuari. En el cas que un usuari mostri la seva fitxa es mostrarà les seves pàgines preferides.



4.6.11 Formulari de creació i edició d'un usuari

El següent esquema mostra el formulari que permet la creació de nous usuaris i la edició d'usuaris ja creats.



4.7 Gantt de tasques de desenvolupament

5 Desenvolupament i Proves

En aquest apartat es mostra com s'ha desenvolupat el projecte. Per convenció tot el projecte s'ha desenvolupat en Anglès, per tant el model Projecte es diu Project i de la mateixa manera els controladors i les vistes.

5.1 Gestor de Projectes

5.1.1 Model

El model *Project* estableix el vincle amb la taula de la base de dades PROJECTS. Al model s'ha descrit la seva relació amb el model pàgina de la següent manera:

has_many :pages

En el model Project s'han definit les següents validacions.

- Validació de presencia: **validates_presence_of :name, :description**
- Validació d'únic: **validates_uniqueness_of :url_slug**

El camp *url_slug* serveix per identificar de manera mes descriptiva el projecte en el qual s'està treballant. Com el camp *url_slug* ens serveis per identificar el projecte quest ha de ser únic i es per això que el camp es inicialitzat cada vegada que el projecte es modificat amb: **before_validation :initialize_slug**

La funció *inicialitze_slug* comprova el nom de projecte i el transforma en un identificador únic al sistema separat per guions.

5.1.2 Controlador

El controlador atén les peticions del navegador d'internet sobre els projectes, sent identificades per el fitxer de rutes i enviant la petició a l'acció corresponent. Les accions de les que disposa el controlador projectes son:

1. **index:** l'acció index es la encarregada de recopilar tots els projectes disponibles al sistema.

2. **show:** l'acció show cerca per el identificador *:url_slug* passat a la ruta, si es trobat es crea una variable per poder ser visualitzat a la vista. Aquesta acció també comprova si el projecte disposa d'un repositori GIT.

3. **new:** l'acció new inicialitza en una variable un projecte buit amb *Project.new*, aquesta variable serveix per que la vista pugui crear un projecte mes fàcilment.

4. **edit:** l'acció edit identifica el projecte que es vol editat per mitjà d'un *:url_slug,* si es trobat el guarda en una variable per ser tractat a la vista.

5. **create:** l'acció create s'encarrega de crear un projecte amb les dades enviades per el formulari de l'acció *:new*. Si les dades son correctes i el projecte es guardat correctament a la base de dades ens redirigeix a la vista *:show* del projecte creat mostrant una notificació. En el cas de que les dades no siguien correctes l'usuari serà redirigit a formulari un altre cop indicant quines dades no son vàlides.

6. **update:** l'acció update fa les mateixes accions que la acció *:create* però en aquest cas s'actualitza un projecte ja existent.

7. **destroy:** l'acció destroy busca el projecte corresponent i l'elimina. Un cop eliminat el projecte ens redirigeix al llistat de tots els projectes del sistema.

5.1.3 Vistes

Les vistes es corresponen amb les accions del controlador, però no totes les accions només les que mostren informació. En el cas dels projectes totes les vistes son el tipus HTML i son les següents:

1. **index:** aquesta vista mostra un llistat amb informació de tots els projectes del sistema. Per cada projecte es mostra el nom de projecte i les accions disponibles: veure, editar, destruir i crear un de nou.

2. **show:** aquesta vista mostra la informació del projecte . La informació mostrada es el nom, la descripció, els autors i si s'ha introduït informació sobre un repositori de codi GIT un enllaç a veure el codi. Desde la vista show es pot accedir anar a la vista **:edit** i **:index**.

3. **new:** la vista new facilita un formulari per la introducció de dades d'un projecte nou.

4. **edit:** la vista edit mostra la informació d'un projecte en un formulari que permet la modificació de les dades d'aquest.

5.1.4 Tests

Per el desenvolupament d'aquest apartat s'ha realitzat els següents testos que comproven que tot vagi segons el previst.

5.1.4.1 Tests unitaris

Els següents testos comproven que les restriccions i validacions introduïdes al model Project es complexin.

1. Comprovació de la correcta creació d'un projecte
2. Comprovació de la correcta inicialització del *:url_slug*
3. Comprovació de que es requereix el atribut *:name*
4. Comprovació de que es requereix el atribut *:description*

5.1.4.2 Tests Funcionals

Els següents testos funcionals comprovaran que la resposta del controlador de Projetes sigui la esperada depenent de les entrades.

1. Test de l'acció **:index**

   Comprova que respongui a una petició GET amb una resposta HTTP 200. Comprova que la variable que ha de contenir tots els projectes de l'aplicació no sigui *nil*.
2. Test de l'acció **:new**

   Comprova la resposta sobre la petició GET sigui HTTP 200.
3. Test de l'acció **:create**

   Comprova que sobre una petició POST amb els paràmetres necessaris per la creació d'un projecte sigui creat un nou projecte i l'usuari sigui redirigit a la vista show del projecte
4. Test de l'acció **:show**

   Comprova que donada una petició GET i enviant com a paràmetre el *:url_slug* d'un projecte el controlador dona una resposta HTTP 200.
5. Test de l'acció **:edit**

   Comprova que donada una petició GET i enviant com a paràmetre el *:url_slug* d'un projecte el controlador dona una resposta HTTP 200.
6. Test de l'acció **:update**

   Comprova que donada una petició PUT amb paràmetres dona resposta i redirigeix al llistat de projectes
7. Test de l'acció **:destroy**

   Comprova que donada una petició DELETE enviant com a paràmetre el identificador d'un projecte el nombre de projectes disminueix en 1. També comprova que ens redirigeix cap al llistat de projectes.

5.2 Visualització del codi del repositori GIT

Per la visualització del codi en un repositori GIT s'han realitzat 2 controladors un per veure carpetes i un per veure fitxers. Com la informació no es troba la base de dades per aquest apartat no cal un model.

5.2.1 Controlador de carpetes

Aquest controlador té el nom de *trees_controller* en referencia a que una carpeta a un repositori git té la classe *Tree*. Aquest controlador només té una acció.

1. **show:** aquesta acció detecta el projecte sobre el qual es busca el codi. Quan s'obté el projecte, s'inicialitza una variable per indicar quin repositori té el projecte i amb el identificador passat per paràmetres es detecta quina es la carpeta que es vol mostrar.

5.2.2 Vistes del controlador de carpetes

Donat que només hi ha una acció al controlador només hi ha una vista per visualitzar les carpetes.

1. **show:** aquesta vista mostra un enllaç per tornar a l'arrel i mostra tot el contingut de la carpeta amb a enllaç. Segons el tipus de cada element de la carpeta l'enllaç t'envia al controlador de carpetes o al de fitxers.

5.2.3 Controlador de fitxers

El controlador de fitxer té el nom de *blob_controllers* això es degut a un fitxer a dins d'un repositori GIT rep el nom **blob**. Aquest controlador només té una acció que es la visualitzar.

1. **show:** identifica el projecte sobre el qual volem veure el codi font, i cerca al repositori el fitxer segons el codi enviat per paràmetres.

5.2.4 Vistes del controlador de fitxers

Donat que només hi ha una acció al controlador només hi ha una vista per visualitzar els fitxers.

1. **show:** aquesta vista mostra un enllaç per tornar a l'arrel i mostra el contingut del fitxer que hem indicat.

5.3 Gestor de Pàgines

5.3.1 Model

El model *Page* estableix el vincle amb la taula de la base de dades PAGES. Al model s'han descrit les següents relacions:

belongs_to :project

belongs_to :author, :class_name => "User", :foreign_key => "author_id"
has_many :slots, :class_name => 'PageSlot', :order => 'position', :dependent => :destroy
has_many :user_pages
has_many :users, :through => :user_pages

En el model Page s'han definit les següents validacions:

- Validació de la presencia de títol i de identificador de projecte: **validates_presence_of :title, :project_id**
- Validació de la uniquisitat del atribut **:url_slug: validates_uniqueness_of :url_slug**

El camp *url_slug* serveix per identificar de manera mes descriptiva la pàgina en el qual s'està treballant. Com el camp *url_slug* ens serveis per identificar les pàgines quest ha de ser únic i es per això que el camp es inicialitzat cada vegada que la pàgina es modificada amb: **before_validation :initialize_slug**

La funció **inicialitze_slug** comprova el nom de la pàgina i el transforma en un identificador únic al sistema separat per guions.

5.3.2 Controlador

El controlador de pàgines atén les peticions del navegador d'internet segons son identificades a les rutes. Les rutes de pàgines van precedides per el *:url_slug* d'un projecte i la paraula pages. Les accions que atén el controlador son:

1. **index:** l'acció index es identifica el projectes en el qual estem treballant i busca totes les pàgines relacionades amb aquest projecte.

2. **show:** a l'acció show se li envia per paràmetres el *url_slug* d'una pàgina i cerca la pàgina que correspon per ser mostrada. També es crea una variable per identificar si la pàgina ha estat marcada com a preferida per l'usuari.

3. **new:** l'acció new inicialitza en una variable una pàgina buit amb **Page.new**, aquesta variable serveix per que la vista pugui crear una pagina mes fàcilment.

4. **edit:** l'acció edit identifica la pàgina que es vol editat per mitjà del *:url_slug* passat per paràmetres. Una porció de codi html es renderitzada com a text pla i guardada a una variable amb el nom de **@form**. El codi html guardat a la variable **@form** conté el formulari per modificar el títol de la pàgina.

5. **create:** l'acció create es la encarregada de crear una pàgina a partir del paràmetres enviats desde la vista de la acció **:new**. Per afegir l'autor de la pàgina identifica l'usuari per mitjà de la sessió. Si la pàgina es guardada correctament l'usuari es redirigit a l'acció **:show** de la pàgina. En el cas que no sigui correcte es torna a mostrar l'acció **:new**.

6. **update:** l'acció update identifica la pàgina i l'actualitza per mitjà de les dades enviades per paràmetres des de el formulari de l'acció **:edit**. Tant si la pàgina es correctament guardada com si no la vista es actualitzada amb notificacions.

7. **destroy:** l'acció destroy busca la pàgina corresponent i l'elimina. Un cop eliminada la pàgina ens redirigeix al llistat de totes les pàgines del projectes.

5.3.3 Vistes

Als següents punts es descriuen les diferents vistes que es corresponen amb accions del controlador de pàgines. La majoria de les vistes que es descriuen son HTML.

1. **index:** aquesta vista mostra un llistat amb totes les pàgines d'un projecte paginades. Per cada pàgina es mostra el títol i les opcions d'anar a la pàgina o eliminar-la. En aquesta vista també proporciona un enllaç per la creació de noves pàgines.

2. **show:** la vista show mostra la informació introduïda a una pàgina. La informació d'una pàgina por provenir d'una nota de text (model *Note*), una anotació de codi (model *CodeNote*), fitxer (model *FileNote*) i si el projecte té associat un repositori de codi un objecte de repositori (model *GitNote*). Els cada

element de la pàgina esta relacionat amb un slot de pàgina (model *PageSlot*), aquest slot son els que mantenen la posició i permet que siguin ordenables. Aquesta vista proporciona un enllaç per la creació de nous elements a la pàgina, que seran afegit per ajax.

3. **new:** la vista new mostra un formulari per la creació d'una nova pàgina del projecte.

4. **edit:** la vista edit es una vista javascript en la qual es substitueix el títol de la pàgina per un formulari i es centra el cursor en aquest formulari.

5.3.4 Tests

Pel desenvolupament d'aquest apartat s'ha realitzat els següents testos que comproven que tot vagi segons el previst.

5.3.4.1 Tests unitaris

Els següents testos comproven que les restriccions i validacions introduïdes al model Project es complexin.

1. Comprovació de la correcta creació d'un projecte
2. Comprovació de la correcta inicialització del **:url_slug**
3. Comprovació de que es requereix el atribut **:name**
4. Comprovació de que es requereix el atribut **:project_id**

5.3.4.2 Tests Funcionals

Els següents testos funcionals comprovaran que la resposta del controlador de Projectes sigui la esperada depenent de les entrades.

1. Test de l'acció **:index**

   Comprova que respongui a una petició GET amb una resposta HTTP 200 indicant el projecte en el qual treballa. Comprova que la variable que ha de contenir tots les pagines del projecte no sigui **nil**.

2. Test de l'acció **:new**

   Comprova que donada una petició GET la ruta corresponent enviant per paràmetres l'identificador d'un projecte aquest es rep una resposta HTTP 200.

3. Test de l'acció **:create**

   Comprova que donada una petició POST amb els paràmetres per crear una pàgina i l'identificador d'un projecte, el nombre de pàgines creix en 1. Comprova que un cop creada la nova pàgina l'usuari sigui redirigit a l'acció show de la pàgina.

4. Test de l'acció **:show**

   Comprova que enviant per paràmetres l'identificador **:url_slug** d'una pàgina per una petició GET es rep una resposta HTTP 200.

5. Test de l'acció **:edit**

   Comprova que enviant per paràmetres l'identificador **:url_slug** d'una pàgina per una petició GET es rep una resposta HTTP 200.

6. Test de l'acció **:destroy**

   Comprova que enviant per paràmetres l'identificador **:url_slug** d'una pàgina per una petició DELETE es veu reduït el nombre de pàgines en 1. En aquest test també es comprova que un cop es eliminat el projecte l'usuari es redirigit al llistat de pàgines del projecte.

5.4 Contingut de pàgina

5.4.1 Slot de pàgina

5.4.1.1 Model

El Model *PageSlot* es l'element que relaciona una pàgina amb els seu contingut. Al model *PageSlot* se li han afegit capacitats polimòrfiques per acceptar relacions amb els diferents contenidors d'informació. Per disposar d'aquestes capacitats polimòrfiques s'ha afegit un camp per guardar la classe de l'element relacionat

(**:rel_objecj_type**) i un cap per guardar l'identificador de l'element relacionat (**:rel_object_id**), la relació s'acaba establint al model amb la següent línia:

belongs_to :rel_object, :polymorphic => true

Com el model també té relació amb pàgina, hi ha la següent línia per identificar la relació existent:

belongs_to :page

5.4.1.2 Plugin: acts_as_list

El plugin *acts_as_list* serveix per donar a un element capacitats d'ordenació. En aquest cas s'han donat aquestes capacitats al l'slot de pàgina per permetre ordenar el contingut de la forma mes fàcil possible. Per funcionar el plugin té els següents requeriments:

1. Disposar d'un camp amb el nom de **:position** on guardar la posició dins d'un llistat
2. Afegir la següent línia al model: **acts_as_list :scope => :page**
3. Creació d'una acció per ordenar els elements. En aquest cas s'ha afegit una acció **:sort** al controlador de pages. Quan un element es reordenat es fa una crida ajax que guarda el nou ordre dels elements de la pàgina.

5.4.2 Nota de Text

5.4.2.1 Model

El model *Note* es l'encarregat de emmagatzemar la informació textual de les pàgines a la taula *NOTES*. Com s'ha explicat a l'apartat anterior el contingut de la informació d'una pàgina es relacionada amb aquesta per mitjà d'un slot. Per realitzar aquest relació s'ha definit el següent codi al model.

has_one :page_slot, :as => :rel_object, :dependent => :destroy

La relació definida marca que es polimòrfica com atribut *:rel_object*.

5.4.2.2 Controlador

El controlador de notes de text atén les peticions ajax del navegador d'internet realitzades des de la vista d'una pàgina. Les per peticions son identificades per les rutes definides. Les accions que atén el controlador son

1. **create:** la acció create recull les dades de la pàgina que s'està editant i inicia una transacció. La transacció crea un nou slot i una nova nota i les insereix a la pàgina.

2. **edit:** la acció edit recull la informació de la nota i genera la vista per editar la nota via ajax.

3. **update:** l'acció update recull les dades enviades des del formulari de la vista **:edit** i actualitza la informació de la nota de la pàgina. Si la nota s'ha pogut actualitzar be, el formulari de d'edició de la nota que es substituït per mostrar el nou contingut de la nota.

4. **destroy:** l'acció destroy identifica la nota i la elimina de la base de dades. Des del controlador també s'elimina la slot que conté la nota a la pàgina per ajax.

5.4.2.3 Vistes

Als següents punts es descriuen les diferents vistes que es corresponen amb accions del controlador de notes. Donat que es totes les peticions son del tipus ajax, les vistes son del tipus javascript amb codi ruby embedit.

1. **create:** la vista create s'encarrega d'introduir un nou slot al llistat de la pàgina i d'insertar la informació de la nota dins del slot amb un formulari per introduir la informació sobre la anotació.

2. **edit:** la vista edit reemplaça la els elements que mostren la informació del slot per un formulari on editar la nota. El formulari un cop emplenat s'enviarà via ajax a l'acció **:update**.

5.4.2.4 Tests

5.4.2.4.1 Tests unitaris

Els tests unitaris corresponen al model *Note* on es comproven les relacions i la correcta creació dels elements. Els tests unitaris sobre el model *Note* son els següents:

1. Comprovació que quan una nota d'una pàgina es creada es accessible per la seva relació d'slot.

5.4.2.4.2 Tests funcionals

Els següents tests funcionals comproven la resposta del controlador de notes de text *notes_controller*. Els tests funcionals son els següents:

1. Test de l'acció **:create**
   Comprova que donada una petició POST enviada via ajax una nova nota sigui creada, que la nota disposi d'un slot a la pàgina i que la nota hagi estat inserida a la pàgina.

2. Test de l'acció **:edit**
   Comprova que la nota que vol ser modificada sigui identificada correctament i la visualització de la nota sigui modificada pel formulari. També es comprovat que la resposta HTTP es 200.

3. Test de l'acció **:update**
   Comprova que la nota es actualitzada amb els paràmetres enviats i la resposta HTTP es 200.

4. Test de l'acció **:destroy**

Comprova que la nota es identificada i eliminada juntament amb el seu slot i la resposta HTTP es 200 per la petició DELETE enviada per ajax.

5.4.3 Nota de codi

5.4.3.1 Model

El model *CodeNote* es l'encarregat de guardar les anotacions de codi d'una pàgina a la taula CODE_NOTES. Per poder relacionar una anotació de codi amb una pàgina, al model *CodeNote* se li ha afegit la següent relació amb el model *PageSlot*:

has_one :page_slot, :as => :rel_object, :dependent => :destroy

La relació definida marca que es polimòrfica com atribut *:rel_object*.

5.4.3.2 Controlador

El controlador de notes de codi atén les peticions ajax del navegador d'internet realitzades des de la vista d'una pàgina. Les per peticions son identificades per les rutes definides per adreçar les peticions a les accions del controlador. Les accions que atén el controlador son

1. **create:** la acció create recull les dades de la pàgina que s'està editant i inicia una transacció. La transacció crea un nou slot i una nova nota de codi i les insereix a la pàgina.

2. **edit:** la acció edit recull la informació de la nota de codi i genera la vista per editar la nota via ajax.

3. **update:** l'acció update recull les dades enviades des del formulari de la vista **:edit** i actualitza la informació de la nota de codi de la pàgina. Si la nota s'ha pogut actualitzar be, el formulari es substituït per mostrar el nou contingut de la nota de codi amb el codi ressaltat.

4. **destroy:** l'acció destroy identifica la nota de codi i la elimina de la base de dades. Des del controlador també s'elimina el slot que conté la informació de l'anotació de codi dins de la pàgina.

5.4.3.3 Vistes

Als següents punts es descriuen les diferents vistes que es corresponen amb accions del controlador de notes. Donat que es totes les peticions son del tipus ajax, les vistes son del tipus javascript amb codi ruby embedit.

1. **create:** la vista create s'encarrega d'introduir un nou slot al llistat de la pàgina i d'insertar un formulari per la introducció de les dades d'un nota de codi. Al enviar el formulari es fa un petició ajax PUT per la acció **:update**. A la petició s'enviaran les dades de la nota, el llenguatge que es va servir i el codi.

2. **edit:** la vista edit reemplaça la els elements que mostren la informació del slot per un formulari on editar la nota de codi. El formulari un cop emplenat s'enviarà via ajax a l'acció **:update** amb una petició PUT.

5.4.3.4 Tests

5.4.3.4.1 Tests unitaris

Els tests unitaris corresponen al model *CodeNote* on es comproven les relacions i la correcta creació dels elements. Els tests unitaris sobre el model *CodeNote* son els següents:

1. Comprovació que quan una nota de codi d'una pàgina es creada es accessible per la seva relació d'slot.

5.4.3.4.2 Tests funcionals

Els següents tests funcionals comproven la resposta del controlador de notes de codi code*_notes_controller*. Els tests funcionals son els següents:

1. Test de l'acció **:create**

   Comprova que donada una petició POST enviada via ajax una nova nota de codi sigui creada, que amb la nota de codi sigui creat un slot a la pàgina i que la nota hagi estat inserida a la pàgina.

2. Test de l'acció **:edit**

   Comprova que la nota de codi que vol ser modificada sigui identificada correctament i la visualització de la nota sigui modificada pel formulari. També es comprovat que la resposta HTTP es 200 al respondre a una petició GET.

3. Test de l'acció **:update**

   Comprova que la nota de codi es actualitzada amb els paràmetres enviats i la resposta HTTP es 200 donada la petició PUT.

4. Test de l'acció **:destroy**

   Comprova que la nota de codi es identificada i eliminada juntament amb el seu slot i la resposta HTTP es 200 per la petició DELETE enviada per ajax.

5.4.4 Nota de Fitxer

5.4.4.1 Model

El model *FileNote* es l'encarregat de guardar les anotacions de fitxer d'una pàgina a la taula FILE_NOTES. Per poder relacionar una anotació de fitxer amb una pàgina, al model *FileNote* se li ha afegit la següent relació amb el model *PageSlot*:

has_one :page_slot, :as => :rel_object, :dependent => :destroy

La relació definida marca que es polimòrfica com atribut *:rel_object*.

Per poder afegir fitxers al model s'ha instal·lat el plugin per Ruby on Rails *paperclip* i per determinar que aquest model té un fitxer adjunt s'ha afegit la següent instrucció:

has_attached_file :file

5.4.4.2 Controlador

El controlador de notes de fitxer atén les peticions ajax del navegador d'internet realitzades des de la vista d'una pàgina. Les per peticions son identificades per les rutes definides per adreçar les peticions a les accions del controlador. Les accions que atén el controlador son:

1. **create:** la acció create recull les dades de la pàgina que s'està editant i inicia una transacció. La transacció crea un nou slot i una nova nota de fitxer i les insereix a la pàgina.

2. **edit:** la acció edit recull la informació de la nota de fitxer i genera la vista per editar la nota via ajax.

3. **update:** l'acció update recull les dades enviades des del formulari de la vista **:edit** i actualitza la informació de la nota de fitxer de la pàgina. Si la nota s'ha pogut actualitzar be, el formulari es substituït per mostrar el nou contingut de la nota de fitxer mostrant el fitxer i un enllaç per descarregar-lo.

4. **destroy:** l'acció destroy identifica la nota de fitxer i la elimina de la base de dades. Des del controlador també s'elimina el slot que conté la informació de l'anotació de codi dins de la pàgina.

5.4.4.3 Vistes

Als següents punts es descriuen les diferents vistes que es corresponen amb accions del controlador de notes de fitxer file_notes_controller. Donat que es totes les peticions son del tipus ajax, les vistes son del tipus javascript amb codi ruby embedit.

1. **create:** la vista create s'encarrega d'introduir un nou slot al llistat de la pàgina i d'insertar un formulari per la introducció de les dades d'un nota de codi. Al enviar el formulari es fa un petició ajax PUT per la acció **:update**. A la petició s'enviaran les dades de la nota, el llenguatge que es va servir i el codi.

2. **edit:** la vista edit reemplaça la els elements que mostren la informació del slot per un formulari on editar la nota de codi. El formulari un cop emplenat s'enviarà via ajax a l'acció **:update** amb una petició PUT.

5.4.4.4 Tests

5.4.4.4.1 Tests unitaris

Els tests unitaris corresponen al model *FileNote* on es comproven les relacions i la correcta creació dels elements. Els tests unitaris sobre el model *FileNote* son els següents:

1. Comprovació que quan una nota de fitxer d'una pàgina es creada es accessible per la seva relació d'slot.

5.4.4.4.2 Tests funcionals

Els següents tests funcionals comproven la resposta del controlador de notes de fitxer *file_notes_controller*. Els tests funcionals son els següents:

1. Test de l'acció **:create**
   Comprova que donada una petició POST enviada via ajax una nova nota de fitxer sigui creada, que amb la nota de fitxer sigui creat un slot a la pàgina i que la nota hagi estat inserida a la pàgina.

2. Test de l'acció **:edit**
   Comprova que la nota de fitxer que vol ser modificada sigui identificada correctament en una variable i la visualització de la nota sigui modificada pel formulari. També es comprovat que la resposta HTTP es 200 al respondre a una petició GET.

3. Test de l'acció **:update**

   Comprova que la nota de fitxer es actualitzada amb els paràmetres enviats i la resposta HTTP es 200 donada la petició PUT.

4. Test de l'acció **:destroy**

Comprova que la nota de fitxer es identificada i eliminada juntament amb el seu slot i la resposta HTTP es 200 per la petició DELETE enviada per ajax.

5.4.5 Nota d'element GIT

5.4.5.1 Model

El model *GitNote* es l'encarregat de guardar les anotacions d'elements git d'una pàgina a la taula GIT_NOTES. Per poder relacionar una anotació de fitxer amb una pàgina, al model *GitNote* se li ha afegit la següent relació amb el model *PageSlot*:

has_one :page_slot, :as => :rel_object, :dependent => :destroy

La relació definida marca que es polimòrfica com atribut *:rel_object*.

El model guardarà un referència a l'element git per poder ser mostrat a la pàgina.

5.4.5.2 Controlador

El controlador de notes d'elements git atén les peticions del navegador d'internet realitzades des de la vista d'una pàgina. Les per peticions son identificades per les rutes definides per adreçar les peticions a les accions del controlador. Les accions que atén el controlador son:

1. **show:** la acció show permet recorre tots el sistema de fitxers i carpetes d'un repositori GIT per escollir quin element es vol afegir.

2. **create:** la acció create recull les dades de la pàgina que s'està editant i inicia una transacció. La transacció crea un nou slot i una nova nota d'element de repositori i les insereix a la pàgina.

3. **index:** la acció index identifica el projecte, la pàgina i l'element inicial de repositori.

4. **destroy:** l'acció destroy identifica la nota d'element de repositori git i la elimina de la base de dades. Des del controlador també s'elimina el slot que conté la informació de l'anotació.

5.4.5.3 Vistes

Als següents punts es descriuen les diferents vistes que es corresponen amb accions del controlador de notes d'elements GIT. Les vistes son del tipus HTML amb codi ruby embedit (html.erb).

1. **show:** la vista show mostra els elements del repositori GIT del projecte. La vista mostra tant el contingut de les carpetes (trees), com el dels fitxers (blobs). Per tots dos casos es mostrat un enllaç per guardar la vista com a anotació d'element git.

2. **index:** la vista index mostra el contingut de la carpeta contenidora de la arrel del repositori GIT. Cada element es un enllaç per mostrar el seu contingut.

5.4.5.4 Tests

5.4.5.4.1 Tests unitaris

Els tests unitaris corresponen al model *GitNote* on es comproven les relacions i la correcta creació dels elements. Els tests unitaris sobre el model *GitNote* son els següents:

1. Comprovació que quan una nota de fitxer d'una pàgina es creada es accessible per la seva relació d'slot.

5.4.5.4.2 Tests funcionals

Els següents tests funcionals comproven la resposta del controlador de notes de fitxer *git_notes_controller*. Els tests funcionals son els següents:

5. Test de l'acció **:create**
   Comprova que donada una petició POST una nova nota de respositori sigui creada, que amb la nota de repositori sigui creat un slot a la pàgina i que la nota hagi estat inserida a la pàgina.

6. Test de l'acció **:show**
   Comprova que la nota d'element git sigui identificada correctament en una variable. També es comprovat que la resposta HTTP es 200 al respondre a una petició GET.

7. Test de l'acció **:index**
   Comprova que la nota d'element git sigui identificada correctament en una variable. També es comprovat que la resposta HTTP es 200 al respondre a una petició GET.

8. Test de l'acció **:destroy**
   Comprova que la nota de fitxer es identificada i eliminada juntament amb el seu slot i la resposta HTTP es 200 per la petició DELETE enviada per ajax.

5.5 Gestió d'usuaris

5.5.1 Model

El model *User* es l'encarregat de obtindré i guardar les dades de la taula USERS de la base de dades. El model d'usuaris esta relacionat amb pàgines com autor i com per guardar les pàgines preferides. Les relacions son les següents:

has_many :user_pages
has_many :pages, :foreign_key => "author_id"
has_many :favorites, :through => :user_pages, :source => :page

En el model *User* s'han definit les següents validacions sobre les dades:

validates_presence_of :name
validates_presence_of :surname
validates_presence_of :email
validates_presence_of :email_confirmation, :if => :mail_changed?
validates_presence_of :password, :if => :password_required?
validates_presence_of :password_confirmation, :if => :password_required?
validates_confirmation_of :password, :if => :password_required?
validates_confirmation_of :email, :if => :mail_changed?
validates_length_of :password, :within => 6..12, :if => :password_required?
validates_uniqueness_of :email
validates_format_of :email, :with => /\A([^@\s]+)@((?:[-a-z0-9]+\.)+[a-z]{2,})\Z/i

Com el password de l'usuari es guarda en encriptat es crea un atribut virtual:

attr_accessor :password

Per encriptar el password just abans de guardar les dades d'un usuari el password es encriptat amb la funció *:encrypt_password*.

before_save :encrypt_password

5.5.2 Controladors

El controlador atén les peticions del navegador d'internet per obtindré o modificar les dades dels usuaris. Per atendre les peticions sobre els usuari s'ha creat el controlador *users_controller* i per les sessions *sessions_controler*.

Les accions sobre el controlador d'usuaris son:

1. **index:** l'acció index recol·lecta tots els usuaris creats a l'aplicació.

2. **show:** a l'acció show se li envia per paràmetres el *identificador* de l'usuari del qual es volen veure les seves dades.

3. **new:** l'acció new inicialitza en una variable un usuari i comprova que l'usuari que vol accedir es del tipus administrador.

4. **edit:** l'acció edit identifica quin es l'usuari del qual es volen modificar les dades per ser tractades a la vista. Es comprova que l'usuari que vol modificar les dades es un usuari administrador

5. **create:** l'acció create crea un usuari nou amb les dades introduïdes per un usuari administrador.

6. **update:** l'acció update identifica l'usuari del qual es vol modificar les dades. L'usuari es actualitzat amb les dades enviades per paràmetres des del formulari de la vista *edit*. Per actualitzar un usuari l'usuari que fa la petició ha de ser del tipus administrador.

7. **destroy:** l'acció destroy busca l'usuari que es vol eliminar i esborra de la base de dades i del sistema l'usuari sense eliminar les pàgines creades.

Les accions sobre el controlador de sessions son:

1. **create:** l'acció create rep per paràmetres login d'usuari i la paraula clau. En el cas que les dues dades siguin correctes es crea una variable de sessió per poder identificar l'usuari des de qualsevol part del sistema.

2. **destroy:** L'acció destroy elimina la variable de sessió corresponent a l'usuari i el redirigeix a la secció de login.

5.5.3 Vistes

Les vistes es corresponent amb accions del controlador, però no totes les accions tenen vista, donat que no han de mostrar informació, només tractar-la. Les següents vistes corresponent al controlador d'usuaris, donat que el controlador de sessions no te cap vista.

1. **index:** la vista de l'acció index mostra un llistat amb tots els usuaris de l'aplicació i les accions disponibles per cada usuari. Les accions disponibles son crear un de nou, editar, veure i eliminar.

2. **new:** la vista de l'acció new mostra un formulari amb els camp necessaris per la creació d'un nou usuari.

3. **show:** la vista de l'acció show mostra la majoria de les dades enregistrades de l'usuari, algunes dades son secretes com ara la paraula clau.

4. **edit:** la vista de l'acció edit mostra un formulari ja emplenat amb les dades de l'usuari. El formulari permet la edició de les dades per actualitzar l'usuari.

5.5.4 Tests

Pel desenvolupament d'aquest apartat s'han realitzat els següents testos on es comprova que el resultat de la codificació sigui l'esperat.

5.5.4.1 Tests unitaris

Els testos unitaris comproven les restriccions i validacions requerides al model *User*.

1. Comprovació de la correcta creació d'un usuari.

2. Comprovació del requeriment de l'atribut nom.
3. Comprovació del requeriment de l'atribut cognom.
4. Comprovació del requeriment de l'atribut correu electrònic.
5. Comprovació del requeriment de l'atribut paraula clau.
6. Comprovació de la confirmació de correu electrònic.
7. Comprovació de la confirmació de la paraula clau.
8. Comprovació de nombre de caràcters mínims d'una paraula clau
9. Comprovació de que el correu electrònic sigui únic.
10. Comprovació del format del correu electrònic.

5.5.4.2 Tests funcionals

Els testos funcionals comproven el correcte de l'aplicació simulant una petició al controlador i comprovant la resposta.

Els següents testos corresponent al controlador d'usuaris.

1. Test de l'acció **:index**
   Comprova que donada una petició GET la resposta HTTP sigui 200 i la variable amb tots els usuaris realment contingui tots els usuaris.

2. Test de l'acció **:show**
   Comprova que al passar per paràmetres l'identificador d'un usuari sigui instanciada una variable amb el se valor. Comprova que donada una petició GET la resposta HTTP sigui 200.

3. Test de l'acció **:new**
   Comprova que donada una petició GET la resposta HTTP sigui 200. Comprova que per accedir a l'acció new l'usuari registrat ha de ser un administrador.

4. Test de l'acció **:edit**
   Comprova que donada una petició GET la resposta HTTP sigui 200. Comprova que per accedir a l'acció edit l'usuari registrat ha de ser un administrador. Comprova que una variable sigui instanciada amb la informació del l'usuari.

5. Test de l'acció **:create**

   Comprova que donada una petició GET l'usuari sigui redirigit al llistat d'usuaris. Comprova que el nombre d'usuari hagi crescut en 1 al realitzar la petició.

6. Test de l'acció **:update**

   Comprova que les dades d'un usuari siguin modificades i l'usuari sigui redirigit al llistat.

7. Test de l'acció **:destroy**

   Comprova que donada una petició DELETE al controlador amb l'identificador d'un usuari, el nombre d'usuaris s'hagi reduït en 1.

5.6 Preferits

5.6.1 Model

El model *UserPage* estableix la relació entre un usuari i una pàgina com a preferida. Per la realització d'aquestes relacions s'han introduït al model el següent:

belongs_to :user
belongs_to :page

5.6.2 Controlador

Per atendre les peticions sobre preferits realitzades per l'usuari s'ha creat el controlador *favorites_controller* . Les accions per el controlador de preferits son les següents:

1. **index:** la acció index mostra les pàgines afegides com a preferides d'un usuari.
2. **create:** la acció create crea una nova relació entre usuari i pàgina com a preferida.

3. **destroy:** la acció destroy elimina una relació existent entre usuari i pàgina com a preferida.

5.6.3 Vistes

Les vistes es corresponent amb accions del controlador. Les següents vistes son del tipus HTML i javascript segon la seva funció:

1. **index:** la vista d'index mostra un llistat amb totes les pàgines marcades com a preferides per l'usuari.
2. **create:** la vista create modifica l'identificador de la pàgina sobre si ja esta marcada com a preferida o no.
3. **destroy:** la vista destroy modifica l'identificador de la pàgina sobre si ja esta marcada com a preferida o no.

5.6.4 Tests

Per la creació d'aquest aparat s'han realitzat els següents testos per comprovar que la resposta sigui la esperada.

5.6.4.1 Tests unitaris

Els tests unitaris comproven les relacions del model *UserPage*.

1. Comproven que des de una pàgina preferida es pugui accedir al seu usuari.
2. Comproven que des de una pàgina preferida es pugui accedir a la seva pàgina.

5.6.4.2 Tests funcionals

Els tests funcional simulen les entrades i sortides de les accions d'un controlador. Per comprovar el correcte funcionament del controlador favorites_controler es defineixen els següents tests:

1. Test de l'acció **:index**

   Comprova que donada una petició GET la resposta HTTP sigui 200 i siguin retornats tots els usuaris registrats a l'aplicació.
2. Test de l'acció **:create**

Comprova que donada una peticó POST la resposta HTTP sigui 200. Comprova que al realitzar la petició el nombre de favorits hagi crescut en 1.

3. Test de l'acció :**destroy**

   Comprova que donada una petició DELETE la resposta HTTP sigui 200. Comprova que al realitzar la petició el nombre de favorits s'hagi disminuït en 1.

5.7 Portada

5.7.1 Controlador

Per mostrar la pàgina principal del sistema un cop identificat l'usuari es crea el controlador *home_controller*. El controlador només té l'acció **:index** per recopilar la informació que es vulgui mostrar al entrar al sistema.

5.7.2 Vistes

Les vistes es corresponen amb les accions del controlador. El controlador *home_controller* nomes te l'acció **index** la qual ens mostra la informació de benvinguda i informació destacada si calgués.

5.7.3 Tests

5.7.3.1 Tests funcionals

Els tests funcionals comproven el correcte funcionament de l'aplicació per mitjà del controlador. Per l'acció **index** es comprova que donada una petició GET la resposta HTTP sigui 200.

5.8 Login principal

5.8.1 Controlador

El controlador de login es el definit com a inicial al fitxer de rutes, d'aquesta manera es el primer en aparèixer quan vols accedir a l'aplicació. Quan el navegador d'internet fa una petició de la url de l'aplicació es el *login_controller* el que l'atén.

L'acció **show** es únic acció del controlador de login i la seva funció es comprovar si hi ha un usuari identificat al sistema. Si l'usuari es identificat et redirigeix al *home_controller*, en cas contrari mostra la seva vista.

5.8.2 Vistes

Les vistes es corresponen a accions del controlador, en el cas del sistema de login el controlador només té la vista show que mostra un formulari per introduir el correu electrònic i la paraula clau per identificar-te.

5.8.3 Tests

5.8.3.1 Tests funcionals

Els tests funcionals comproven el funcionament del sistema realitzant peticions com ho faria l'usuari. En el tests es comprova que donada una entrada en concret la resposta es la esperada Pel controlador de login els tests son els següents:

1. Test de l'acció **:show**

Comprova que donada una petició GET la resposta HTTP es 200 si l'usuari no esta identificar i la resposta es una redirecció si l'usuari ja s'ha identificat.

5.9 Proves manuals

Les proves manuals comproven requeriments del sistema que no es poden comprovar amb tests automàtics. Proves manuals realitzades son les següents:

1. **Prova de cohesió de l'aspecte visual de l'aplicació:** s'ha comprova que totes les vistes del sistema tenen un aspecte visual similar que permet una fàcil identificació dels elements.
2. **Prova del correcte ressaltat de codi:** s'ha comprovat que donat diferents llenguatges de programació el ressaltat ha identificat i colorejat els diferents elements del codi.
3. **Prova de la ordenació d'elements:** s'ha comprovat que les diferents notes d'una pàgina poden ser reordenades arrossegant-los i l'ordre queda guardat.
4. **Prova de la navegació del codi GIT:** s'ha comprovat manualment que es pot crear un projecte amb un repositori GIT i aquest es navegable.
5. **Prova de la visualització de les notes de text amb estils per marcatge:** s'ha comprovar que es poden donar estils al text de les anotacions de text d'una pàgina amb el llenguatge de marcatge indicat. També s'ha comprovat que la visualització del text amb estils es veu correctament.
6. **Prova de que les peticions d'una anotació son ajax:** s'ha comprovat que per afegir, editar i eliminar anotacions d'una pàgina al navegador no li cal recarregar.

6 Conclusions

Finalitzat el projecte es pot avaluar si el aquest ha complert els seus objectius.
Amb els resultats obtinguts podem considerar que el objectiu principal ha estat satisfet. Si recordem l'objectiu principal era crear un sistema per ajudar a la reutilització de codi amb la finalitat d'augmentar la productivitat en la creació de software, el qual s'ha desenvolupat i es funcional amb el requeriments fixats.

Per la creació del sistema de reutilització m'he trobat en la situació que hi han diferents pràctiques ha seguir en el desenvolupament però molt poca informació d'aplicacions per ajudar la reutilització. Per solucionar aquesta falta de base o competència vaig prendre com a estat de l'art diferents aplicacions que fan servir els desenvolupadors. D'aquesta manera la solució creada finalment te com a base a gestors de continguts, wikis i sistemes de documentació.

La planificació s'ha pogut acomplir en el nombre d'hores de dedicació indicats amb petites variacions. Però en quan a les dades ho ha estat possible per falta de temps personal de dedicació fent enrederir el projecte uns mesos.

Finalment podem dir que l'aplicació resultant del projecte final de carrera es una bona eina de suport per la reutilització de codi de programació. He de matisar que per reutilitzar codi s'han de continuar fent servir tècniques com llibreries de software, patrons de disseny i frameworks, donat que la solució donada es una eina de suport a la reutilització i no la solució final per la reutilització de codi.

6.1 Possibles ampliacions

La eina de suport a la reutilització de codi desenvolupada es funcional però sobre tot un bona base per possibles ampliacions per augmentar la seva efectivitat. Les possibles ampliacions que es poden realitzar son les següents:

1. **Sistema de bloqueig d'edició**

   Afegir un sistema per controlar que no hi hagi cap altre usuari modificant una pàgina. En el cas que hi hagi un usuari modificant una pàgina aquesta hauria de quedar bloquejada per evitar conflictes de dades.

2. **Afegir webservice per la integració**

   Afegit les capacitats per convertir la eina en un webservice on el es poguin realitzar peticions i aquest retorni la resposta en XML o JSON. Amb aquest sistema es podrien realitzar plugins per facilitar l'accés a les pàgines on hi ha la informació de la reutilització de codi. El plugins poden ser per exemple NetBeans o Eclipse.

3. **Importació de documentació**

   La importació des de sistemes de documentació permetria incloure de forma ràpida el treball de documentació realitzat.

4. **Afegir suport per mes sistemes gestors de versions**

   Actualment la eina desenvolupada només suporta repositoris del sistema de versions GIT. Una bona ampliació es afegir el suport d'altres sistemes gestor de documentació com Subversion, Mercurial o Bazaar.

7 Bibliografia

Per la realització d'aquest projecte s'han fet servir les següents referències:

Monografies

1. Budd, Andy; Moll, Cameron; Collison, Simon. *CSS Mastery: Advanced Web Standards Solutions*. New York: Apress, 2006. 247pp.
2. Duran Toro, Amador; Bernández Jiménez, Beatriz. *Metodología para el Análisis de Requisitos de Sistemas Software.* Sevilla: Universidad de Sevilla, 2001. 78pp.
3. Otero García, Alberto. *Projecte web.* Barcelona: Eureca Media, 2006. 90pp.
4. Ruby, Sam; Thomas, Dave; Heinemeier, David; et al. *Agile Web Development with Rails*. Texas: The Pragmatic Programmers LLC, 2009. 766pp.

Articles

1. "Code Reuse". [en línia] A: Wikipedia, 2010 <http://en.wikipedia.org/wiki/Code_reuse > (Data d'actualització: 25/01/2011) (Data de consulta: 20/12/2010)
2. "Extreme Programing". [en línia] A: Wikipedia, 2010 <http://en.wikipedia.org/wiki/Extreme_Programming> (Data d'actualització: 28/01/2011) (Data de consulta: 22/10/2010)
3. "Scrum". [en línia] A: Wikipedia, 2010 <http://en.wikipedia.org/wiki/Scrum_(development)> (Data d'actualització: 28/01/2011) (Data de consulta: 22/10/2010)
4. "TDD". [en línia] A: Wikipedia, 2010 <http://en.wikipedia.org/wiki/Test-driven_development >(Data d'actualització: 22/01/2011) (Data de consulta: 22/10/2010)

Pàgines web

1. *Apple Developer Connection* [en línia] <developer.apple.com> (Data actualització: 2010) (Data d'última consulta: 05/08/2010)
2. *Git – Fas Version Control System* [en línia] <http://git-scm.com/> (Data actualització: 05/01/2011) (Data d'última consulta: 19/09/2010)

3. *Grit* [en línia] <http://grit.rubyforge.org/ > (Data actualització: 28/02/2009) (Data d'última consulta: 30/09/2010)
4. *jQuery* [en línia] <http://api.jquery.com/> (Data actualització: 2010) (Data d'última consulta: 26/10/2010)
5. *Paperclip* [en línia] <https://github.com/thoughtbot/paperclip> (Data actualització: 19/02/2011) (Data d'última consulta: 3/09/2010)
6. *RedCloth Textile markup language* [en línia]<http://redcloth.org/> (Data actualització: 2011) (Data d'última consulta: 15/10/2010)
7. *Ruby on Rails Guides* [en línia] <http://guides.rubyonrails.org/ >(Data actualització: 2011) (Data d'última consulta: 02/08/2010)
8. Ultraviolet Syntax Highlighting Engine. [en línia] <http://ultraviolet.rubyforge.org/> (Data d'última consulta: 10/10/2010)