



Pontificia Universidad Católica de Chile
Escuela de Ingeniería
Departamento de Ciencia de la Computación

DOCUMENTO DE DISEÑO

Nombre del proyecto:
Organización

SmartBoard Project Manager
Pontificia Universidad Católica de Chile

Fecha:
Versión:

04 de diciembre, 2013
2.0

Historia del Documento

Versión	Fecha	Autor(es)	Razón del Cambio
0.1	31/10/2013	Thomas Büchi	Primer borrador
1.0	04/11/2013	Thomas Büchi	Versión Terminada
1.4	07/11/2013	Thomas Büchi	Versión Revisada
1.5	27/11/2013	Thomas Büchi	Versión Actualizada
2.0	04/12/2013	Thomas Büchi	Versión Final

Equipo de Desarrollo

Nombres y Apellidos	Rol	Contacto
Nicolás Risso	Administrador del Proyecto	narisso@puc.cl (56 9) 8818-6497
Valentina Ibaseta	Desarrollador/Analista	vjibaset@puc.cl (56 9) 9497-5956
José Tomás Marquinez	Desarrollador/Analista	jtmarquinezv@puc.cl (56 9) 9020-0720
Thomas Büchi	Desarrollador/Diseñador	tbuchi@puc.cl (56 9) 9599-6990
Santiago Larraín	Desarrollador/Diseñador	slarrain@puc.cl (56 9) 8248-2759
Nicolás Escobar	Desarrollador/Tester	niescoba@puc.cl (56 9) 8824-6141
Fernando González	Desarrollador/Tester	fagonza6@puc.cl (56 9) 6727-1956

Contraparte del Proyecto

Nombres y Apellidos	Rol	Contacto
José Ignacio Benedetto	Estudiante	jibenedettoc@gmail.com (56 2) 2354-2000
Andrés Chacón	Estudiante	afchacon2@gmail.com (56 2) 2354-2000

Tabla de Contenidos

Historia del Documento	i
Equipo de Desarrollo	ii
Contraparte del Proyecto	ii
Tabla de Ilustraciones	1
1. Descripción General	1
1.1 Propósito del Sistema	1
1.2 Alcance del Proyecto	2
1.3 Contexto	3
1.4 Definiciones, Acrónimos y Abreviaturas	4
1.5 Referencias	5
2. Vista Lógica	6
2.1.1 Arquitectura del Sistema	6
2.1.2 Arquitectura Lógica	7
2.1.2.1 Interfaz de Usuario	7
2.1.2.2 Servicios de Sistema	8
2.1.2.3 Infraestructura	9
2.1.3 Arquitectura de los Módulos	9
2.1.3.1 Interfaz de Usuario	20
2.1.3.2 Servicios de Sistema	48
3. Vista de Implementación	49
3.1.1 Estructura de la Aplicación	49
3.1.2 Arquitectura de Implementación	49
4. Vista de Datos	1
4.1.1 Modelo de Datos	1
4.1.2 Servicios de Persistencia	1
4.1.3 Servicios de Transaccionalidad	1
5. Vista de Deployment	2
5.1.1 Arquitectura Técnica	2
5.1.2 Tecnología requerida	3
5.1.3 Deployment	3

Tabla de Ilustraciones

Ilustración 1 Diagrama de la arquitectura simplificada del sistema	6
Ilustración 2 Modelo Vista Controlador	7
Ilustración 3 Módulos.....	8
Ilustración 4 Módulos Externos.....	9
Ilustración 5 Controlador de Bugs.....	9
Ilustración 6 Controlador de Roles de Usuario	10
Ilustración 7 Controlador de Commits de Github	10
Ilustración 8 Controlador de Metas	10
Ilustración 9 Controlador de las tareas por usuario.....	11
Ilustración 10 Controlador de Usuarios	11
Ilustración 11 Controlador de las invitaciones para usuarios	11
Ilustración 12 Controlador de los roles de usuario por proyecto	12
Ilustración 13 Controlador de Documentos por proyecto	12
Ilustración 14 Controlador de Documentos por Tarea	13
Ilustración 15 Controlador de los Estados de las tareas por proyecto	13
Ilustración 16 Controlador de Dropbox	13
Ilustración 17 Controlador de los estados	14
Ilustración 18 Controlador de los Casos de Uso.....	14
Ilustración 19 Controlador de Proyectos	15
Ilustración 20 Controlador de los requerimientos de los Templates.....	15
Ilustración 21 Controlador del Logeo de facebook y google.....	15
Ilustración 22 Controlador del Kanban	16
Ilustración 23 Controlador de las Tareas	16
Ilustración 24 Controlador de los Templates de Casos de Uso.....	17
Ilustración 25 Controlador de la Aplicación	17
Ilustración 26 Controlador de los Grupos de Caso de Uso.....	17
Ilustración 27 Controlador de las Sesiones	18
Ilustración 28 Controlador del API	18
Ilustración 29 Controlador de los Comentarios	18
Ilustración 30 Controlador de los Casos de Tests	19
Ilustración 31 Controlador de Github	19
Ilustración 32 Controlador de las etiquetas	19
Ilustración 33 Modelo de Navegación.....	20
Ilustración 34 Interfaz de Ingreso	21
Ilustración 35 Flujo posible de ingreso.....	22
Ilustración 36 Interfaz de Registro	22
Ilustración 37 Flujo posible de Registro	23
Ilustración 38 Interfaz de Proyectos	23
Ilustración 39 Interfaz de creación de Proyectos.....	24
Ilustración 40 Interfaz editar Proyectos	25
Ilustración 41 Diagrama de Flujo de Proyectos.....	26
Ilustración 42 Intefaz del Kanban.....	30
Ilustración 43 Diagrama de flujo de acciones posibles en el kanban.....	31
Ilustración 44 Interfaz de creación de Tareas	32
Ilustración 45 Interfaz ver Tarea	33
Ilustración 46 Interfaz ver Comentarios.....	34

Ilustración 47 Interfaz ver Commits	35
Ilustración 48 Interfaz Reportar Horas.....	36
Ilustración 49 Interfaz ver Subtareas	37
Ilustración 50 Interfaz Crear Subtarea	38
Ilustración 51 Interfaz Agregar Columna	38
Ilustración 52 Interfaz ver Documentos del Proyecto.....	39
Ilustración 53 Interfaz de Casos de Uso	41
Ilustración 54 Diagrama del Flujo posible de en Casos de Uso	42
Ilustración 55 Interfaz Crear Caso de Uso.....	43
Ilustración 56 Interfaz Crear Template	46
Ilustración 57 Interfaz crear nuevo Grupo de Casos de Uso.....	46
Ilustración 58 Interfaz Agregar Usuario a Proyecto	47
Ilustración 59 Diagrama de flujo general de la aplicación	48
Ilustración 60 Arquitectura Ruby on Rails.....	49
Ilustración 61 Front End Módulos	1
Ilustración 62 Arquitectura técnica	2

1. Descripción General

Actualmente la mayoría de los organismos utilizan herramientas computacionales para su operación. Esto, con el objetivo de optimizar procesos, facilitar la comunicación y las tareas dentro de la institución, entre otros. Del mismo modo, las personas han comenzado a utilizar varias aplicaciones y software para organizar personalmente su día a día, lo que ha tenido como consecuencia un constante surgimiento de nuevas necesidades tecnológicas en los usuarios. Es por esto que se requiere crear nuevas herramientas y aplicaciones para satisfacer la creciente demanda.

No obstante, la creación de nuevas aplicaciones no es un trabajo sencillo. La mayoría de los casos requiere un equipo de desarrollo para ello, el que, a su vez, requiere de mucha coordinación y cooperación para poder realizar un buen trabajo en los plazos estimados. El desarrollo de un software está constituido de varias tecnologías y etapas: definición de requisitos de usuario, casos de uso, estructura de la aplicación, modelo de datos, tareas individuales de cada una de las personas en el desarrollo, entre otras cosas. Para abordar cada una de estas etapas en el desarrollo, existen distintas herramientas como GitHub, Dropbox, Kanbanery, Gmail; que permiten la comunicación, coordinación y trabajo confluente de cada una de las partes. Sin embargo, cada una funciona por separado. Esto entrega una dificultad adicional al momento de usarlas de manera integrada para la coordinación y comunicación del equipo de trabajo y con el cliente.

El objetivo de este proyecto es crear una herramienta que permita y facilite el desarrollo de software. Integrará aplicaciones para el manejo de versiones, compartición de archivos, asignación y monitoreo de proyectos, y creación de documentos de caso de uso y requisitos, entre otros.

1.1 Propósito del Sistema

El sistema busca proveer una herramienta open source que permita al equipo de desarrollo llevar un seguimiento constante de su trabajo, de forma que pueda manejar los distintos aspectos de la construcción de su proyecto. El software, aparte de permitir el manejo de versionamiento que es indispensable para un grupo de trabajo, debe integrar secciones para el manejo de documentos como los casos de usos que estén asociados a diferentes requisitos del usuario. Así, se podrá generar constantemente tareas en base a dichos archivos.

Se pretende que la aplicación sea capaz de facilitar la coordinación y ejecución del equipo de desarrollo, así como también la comunicación con el cliente. Estos aspectos son muy relevantes, pues si estos aspectos no logran manejarse de manera adecuada, pueden ser la principal causa de retrasos en la implementación, o incluso de la reestructuración de un proyecto. Se busca generar valor en las reuniones de los clientes de los proyectos que vayan a usar el nuevo sistema, de manera que se le permita al cliente expresar sus ideas sin limitaciones computacionales, a través de un ambiente de trabajo de tablets sincronizados. De esta manera, los clientes, aprovechando las características de los dispositivos, podrán generar requisitos y casos de uso extraídos de lo que el lenguaje computacional exige.

SmartBoard Project Manager debe ser capaz de integrar todos los aspectos que son relevantes para el desarrollo de un proyecto, para que sea un medio de gestión y desarrollo. Ésta debe estar apuntada a cualquier equipo de desarrollo y debe ser simple en su utilización, pues debe facilitar el trabajo y no convertirse en un obstáculo más. Además, se desea que la arquitectura sea tal que pueda ser usada por equipos de estudiantes como medio de aprendizaje y experiencia de trabajo en un desarrollo de software real.

En resumen, se generará un ambiente que busca aumentar la usabilidad para el seguimiento del avance de cualquier proyecto. En particular, estará enfocado en proyectos de desarrollo de software, al buscar estar integrado con diversas plataformas y servicios webs tecnológicos que favorecen la integración de códigos, archivos, calendarios, entre muchos otros. Por otro lado, su ambiente móvil estará integrado fuertemente con el área web, de manera que las reuniones tendrán un valor agregado para el cliente y para el equipo del proyecto.

1.2 Alcance del Proyecto

Las principales funcionalidades que se buscan integrar dentro de la aplicación son las siguientes:

- Sistema de versionamiento de código
- Bug Tracking.
- Project Tracking.
- Testing Management.
- Task Management.
- Agenda y calendario.
- Sistema de administración de mensajes, que permita enviar mails.
- Sistema de archivos en la nube.
- Wiki interna.
- Módulo de documentos, con la posibilidad de exportar a PDF.
- Storyboard sobre el flujo de documentos.
- Herramienta móvil para diagramar.
- API para el login con redes sociales.

Así, el proyecto consiste en integrar cada uno de estos sistemas en una aplicación. Ésta estará constituida por una parte web y otra parte móvil. La primera contempla la parte principal, pues tendrá la mayoría de las funcionalidades. De esta manera, la aplicación móvil estará enfocada en tareas de diagramación para documentos o mock-ups, en las reuniones con clientes.

La primera versión de la herramienta constará de un avance integrado entre las dos partes mencionadas. En lo que respecta de la parte web, ésta consistirá en una plataforma de administración de proyectos que facilite un seguimiento de las tareas o actividades realizadas, realizándose y por realizar. De esta manera, sus usuarios en sus distintos roles podrán trabajar en la gestión de proyectos para la facilitación de la visualización del grado de avance del proyecto.

Permitirá personalizar un tablero de tareas que dispondrá de las distintas actividades registradas en los diversos grados de avance o columnas que dispongan. De esta forma, los usuarios podrán informarse sobre las horas dedicadas en cada tarea, el flujo que han tenido las tareas en el transcurso del tiempo, los comentarios realizados para cada una de las tareas, los responsables de llevar a cabo las tareas, los archivos utilizados o generados para respaldar una actividad. Como la herramienta busca facilitar el trabajo en proyectos de desarrollo, la aplicación estará integrada con un sistema de administración de código fuente como Git. A su vez, el manejo de archivos estará sincronizado y respaldado por la integración que tendrá con un sistema para guardar archivos en la nube.

Para futuros alcances, la sección web de la herramienta podrá incluir módulos de bug tracking, ambientes de testing management, integración con calendarios como Google Calendar, sistema de administración de mensajería, un módulo de documentación de proyectos de la forma de una wiki interna, una API adecuada para el login con redes sociales como Google+, entre otras.

Por otro lado, la primera versión de la parte móvil estará dedicada únicamente a agregarle valor a la generación de casos de uso y requerimientos de los clientes. De esta manera, las reuniones entre el grupo de desarrollo y los clientes contarán con un ambiente de trabajo integrado en que cada integrante de la reunión podrá respaldar, a través de un dispositivo móvil, lo que se esté conversando y dibujando en las reuniones. En otras palabras, la aplicación no será una herramienta para la administración de proyectos. Contará con un ambiente de dibujo sincronizado entre todas las tablets, que permitirá generar diagramas libres y UML. A su vez, se podrá conectar al ambiente de sincronización un Smart TV, pues se sincronizará con un dispositivo conectado a un televisor o proyector que mostrará un lienzo general de lo conversado en las reuniones. Por último, el ambiente estará sincronizado mediante un sistema para guardar archivos en la nube con la parte web, por lo que el trabajo de cada reunión quedará respaldado para su trabajo y modificación en el ambiente web de la herramienta.

1.3 Contexto

SmartBoard Project Manager (SPM) busca apoyar y facilitar la manera en que actualmente se planifican los proyectos de desarrollo de software. Actualmente, los distintos software y servicios existentes en el mercado satisfacen sólo de forma parcial todas las necesidades que pueden surgir a la hora de gestionar un proyecto. Esto exige trabajar con más de uno para lograr tener un gran abanico de características a utilizar. Por ejemplo, para coordinar tareas que llevan a cabo un proyecto más general se deben utilizar herramientas como Kanbanery o Trello; pero para realizar storyboarding se deben utilizar servicios como Fieldtest y Luzmy.

Con la integración de diversos servicios que ayudan en el desarrollo de un proyecto y con la adición de una gran variedad de características que le agregan usabilidad, se pretende construir un sistema que satisfaga gran parte de las necesidades que surgen en el desarrollo de cualquier proyecto, principalmente de desarrollo de software. Así, se busca que sea eficiente en cuanto a la cantidad de plataformas que deben ser utilizadas para manejar correctamente el funcionamiento y avance de un proyecto.

Es por esta razón que los clientes buscan crear un sistema web y móvil que ayude la forma actual de llevar a cabo este tipo de proyectos. Buscan desarrollar desde cero una plataforma de forma sistemática, planificada y que cumpliera con características mínimas para un correcto grado de usabilidad e integración entre las funcionalidades que se deseen, con la finalidad de ocupar la menor cantidad de software disponibles en el mercado posible. Además, que sea escalable a otros proyectos fuera de lo que ellos actualmente realizan, a futuro.

Así, este nuevo sistema funciona a través de una plataforma web que asegura compatibilidad entre varios navegadores y permite organizar un proyecto desde la nube. Esta plataforma se construye sobre el framework RubyOnRails, el más utilizado actualmente para el desarrollo de sistemas por su gran variedad de gemas que permite integrar un gran número de funcionalidades requeridas. A su vez, la sección móvil de SPM trabaja sobre Android nativo aprovechando el SDK de los lápices que los tablets a utilizar incluyen. Así, se asegura una integración fácil con los dispositivos Android Mini PC, que permitirá una pantalla compartida en los ambientes de trabajo. La aplicación móvil que se desarrollará significará un gran avance tecnológico en el ambiente de reuniones que generan los requisitos de las distintas reuniones. Como se busca la integración con la pantalla compartida, el lenguaje utilizado tiene todas las bondades y ventajas que el ambiente genera, permitiendo un entorno sincronizado de trabajo en que todo queda registrado digitalmente de manera casi instantánea. Otra de las razones que favorecen la utilización de dispositivos Android son su popularidad y economía frente a otras alternativas.

1.4 Definiciones, Acrónimos y Abreviaturas

- **Administrador:** Es quien maneja muchos proyectos, y tiene interés en ver el avance y trabajo realizado por cada equipo de trabajo.
- **API:** Interfaz de programación de aplicaciones, en sus siglas en inglés.
- **Bug Tracking:** Sistema de seguimiento de errores de una aplicación informática diseñado para ayudar a asegurar la calidad de software.
- **Cliente:** Es la parte interesada en que el proyecto se lleve a cabo, porque le traerá valor una vez terminado.
- **Desarrollador:** Es quien implementa la aplicación o proyecto. Puede ser tester, desarrollador de código, analista o arquitecto.
- **Git:** Software de sistema de gestión de código fuente.
- **Github:** una plataforma de desarrollo colaborativo de software para alojar proyectos utilizando el sistema de control de versiones Git
- **Jefe de Proyecto:** Es el responsable de que un proyecto se lleve a cabo adecuadamente.
- **Kanban:** Se entiende como el sistema de información que muestra todas las tareas necesarias para llevar a cabo un proyecto, y el estado en el que se encuentra cada tarea, para controlar el avance de dicho proyecto. Está formado por varias columnas que indican el estado de avance de que tiene cada tarea o actividad
- **SDK:** Kit de desarrollo de software, en sus siglas en inglés.

- **Storyboard:** Se entiende como un dibujo libre, que podría utilizarse para mostrar el flujo de un web.

1.5 Referencias

1. Heroku (www.heroku.com/features)
2. Ruby on Rails: Documentation (www.rubyonrails.org/documentation)
3. Guía de estilo Rails (www.github.com/bbatsov/rails-style-guide)
4. The Elements of UML 2.0 Style, Scott W. Amber (2005), Primera edición.
5. IIC2154 2013-2 (Grupo 2), Documento de Perfil de Proyecto de Especialidad (v1.0)

2. Vista Lógica

Esta vista presenta tres niveles de arquitectura para el sistema open source de administración de proyectos SmartBoard.

2.1.1 Arquitectura del Sistema

La aplicación web se encuentra instalada en Heroku, un servicio de hosting que esta sobre Amazon Web Services. La configuración esta en modo Infraestructura como servicio, lo que provee una máquina virtual que ejecuta el servidor de Ruby on Rails y una base de datos PostgreSQL. Esta configuración permite facilitar la escalabilidad del software, ya que a mayor uso aumenta la cantidad de máquinas virtuales, lo que significa que el sistema sólo usa los recursos que necesita.

La arquitectura simplificada del sistema, consta de una cantidad N de usuarios que se conectan a un servidor en la nube y comparten la misma base de datos.

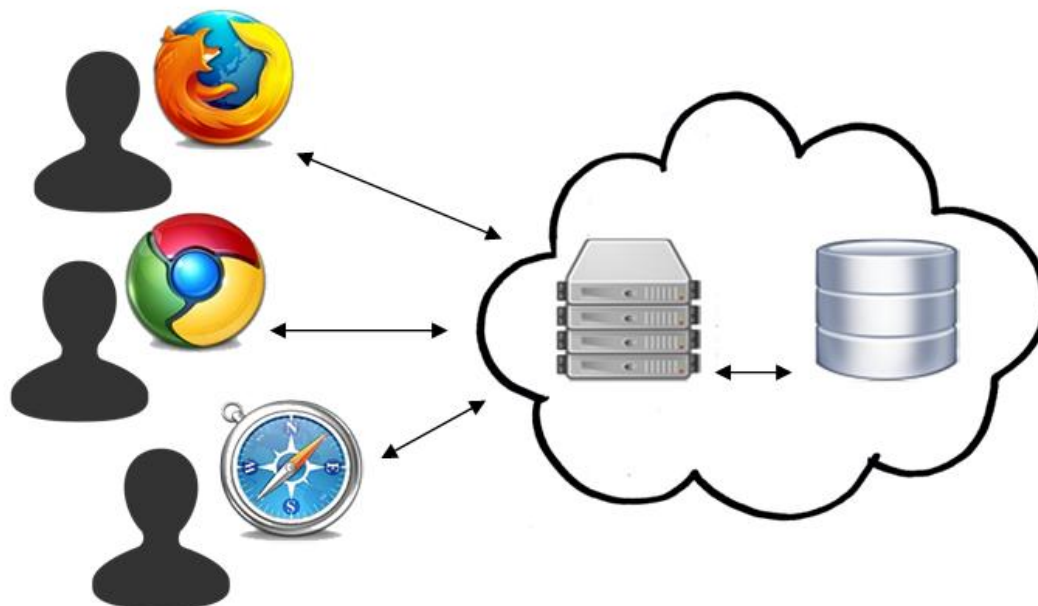


Ilustración 1 Diagrama de la arquitectura simplificada del sistema

La arquitectura de software fue implementada usando un patrón de diseño MVC (Modelo, Vista, Controlador), que es el estándar de las aplicaciones hechas en Ruby On Rails. En el caso de RoR el modelo no sólo incluye una imagen de la base de datos sino también en las migraciones (expresan los cambios hechos en la base de datos), los observadores, y las emigraciones. La vista constituye a la presentación de la información en código html, y es con la cual el usuario interactúa. Controlador se encarga de conectar el modelo con la vista, haciendo consultas, validaciones, creaciones, modificaciones, etc...

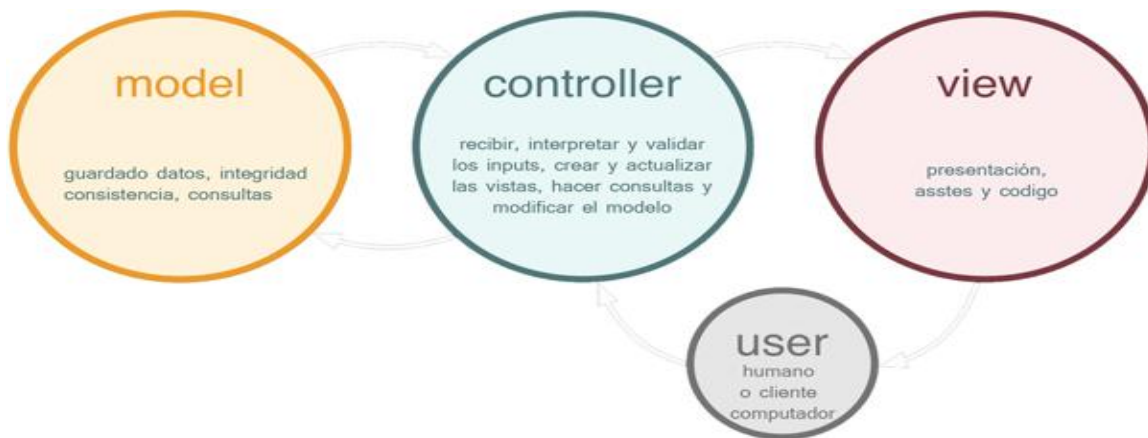


Ilustración 2 Modelo Vista Controlador

2.1.2 Arquitectura Lógica

2.1.2.1 Interfaz de Usuario

La Vista de Casos de Uso muestra el front-end del sistema. El mismo es generado dinámicamente utilizando tecnología de contenido web dinámico. Desde el punto de vista del back-end se tiene un conjunto de páginas dinámicas generadas a partir de los procesos llevados a cabo por el sistema.

La aplicación tiene 7 grandes módulos que contienen todas las funcionalidades de la aplicación web. Estos módulos están directamente asociados a los modelos más importantes del proyecto, excepto el de reportes que es una vista importante y no un objeto en sí.

Los módulos identificados y sus interdependencias se presentan en la *Ilustración 3*, en la siguiente sección.

2.1.2.2 Servicios de Sistema

El siguiente diagrama presenta los módulos identificados y sus interdependencias.

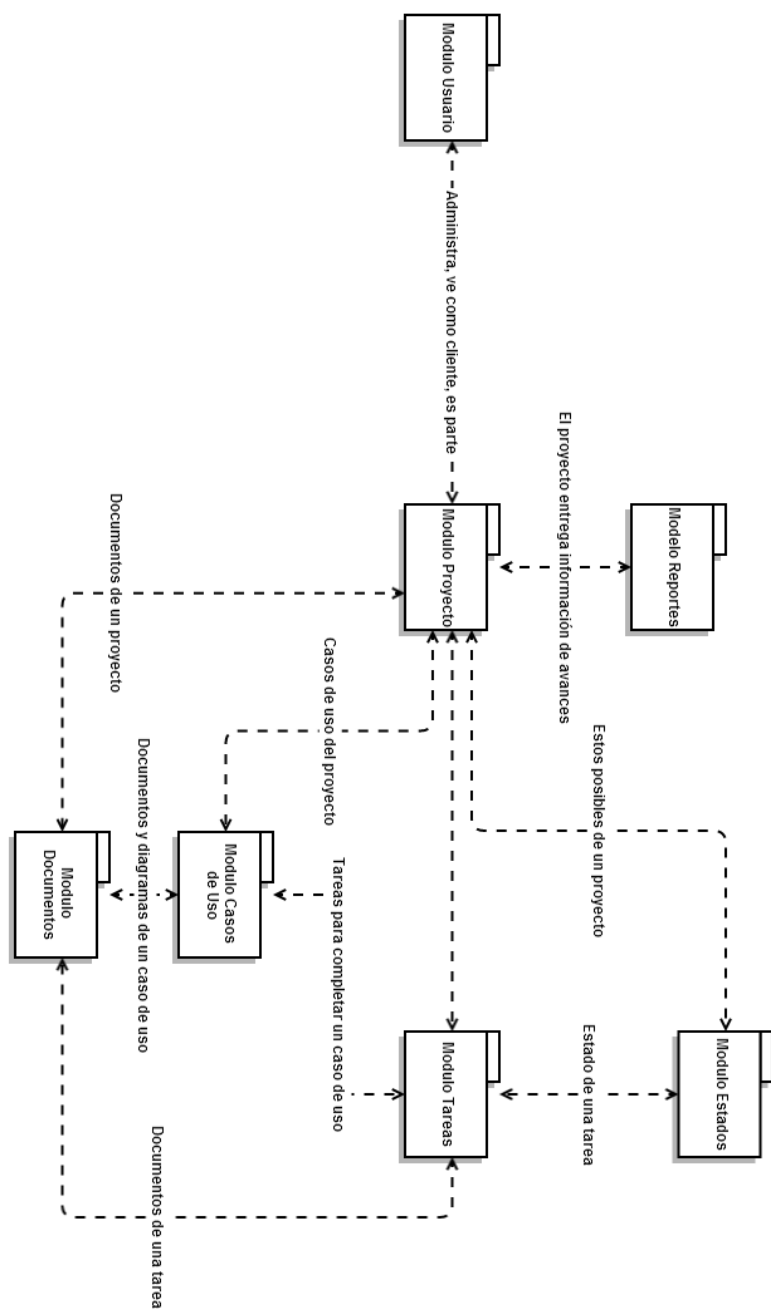


Ilustración 3 Módulos

2.1.2.3 Infraestructura

La aplicación contiene dos módulos para uso de aplicaciones externas, el primero es uno proveído por el framework de Ruby On Rails que es el Action Mailer. Esto permite mandar mails desde la aplicación a los distintos servicios de correo.

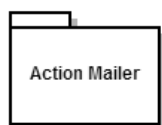


Ilustración 4 Módulos Externos

La segunda es el API para la parte móvil, que es detallada en el documento de diseño móvil.

2.1.3 Arquitectura de los Módulos

La Arquitectura de los Módulos presenta un refinamiento de la Arquitectura Lógica. En nuestra aplicación corresponden a los controladores de la aplicación de Ruby On Rails. Estos son presentados a continuación.

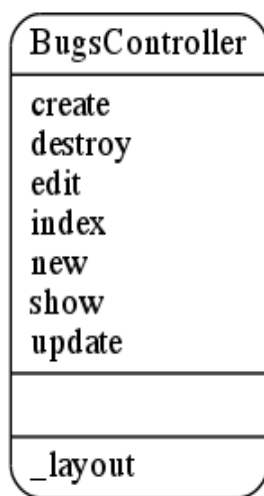


Ilustración 5 Controlador de Bugs

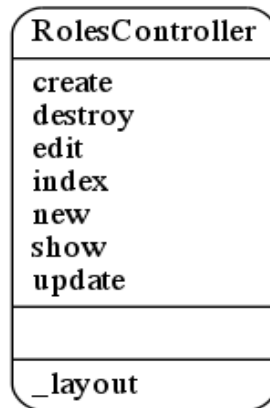


Ilustración 6 Controlador de Roles de Usuario

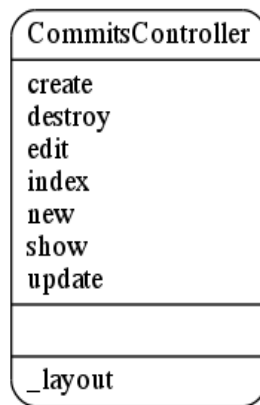


Ilustración 7 Controlador de Commits de Github

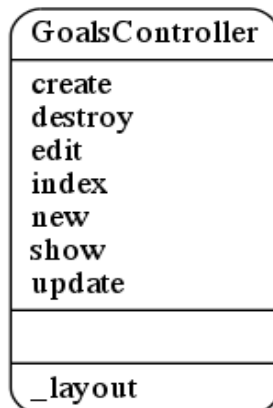


Ilustración 8 Controlador de Metas

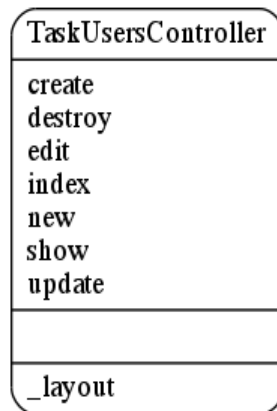


Ilustración 9 Controlador de las tareas por usuario



Ilustración 10 Controlador de Usuarios

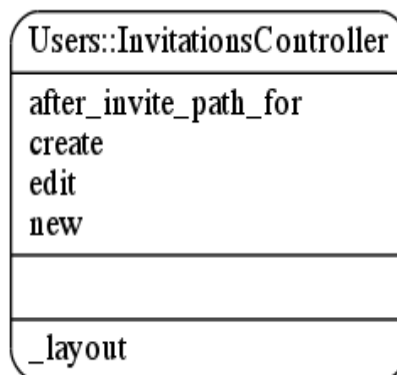


Ilustración 11 Controlador de las invitaciones para usuarios

ProjectRoleUsersController
create destroy edit index new show update
_layout

Ilustración 12 Controlador de los roles de usuario por proyecto

ProjectInviteController
_one_time_conditions_valid_374? accept confirm_password_invitation decide invite reject send_invitation submit_password_invitation
_layout

Ilustración 13 Controlador de Invitaciones de un proyecto

DocumentProjectsController
create destroy edit index new show update
_layout

Ilustración 14 Controlador de Documentos por proyecto

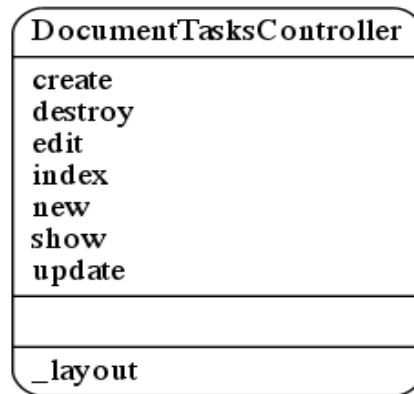


Ilustración 15 Controlador de Documentos por Tarea

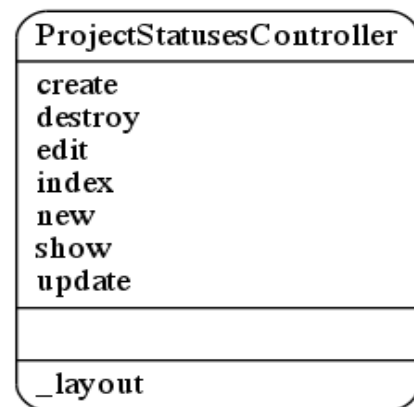


Ilustración 16 Controlador de los Estados de las tareas por proyecto

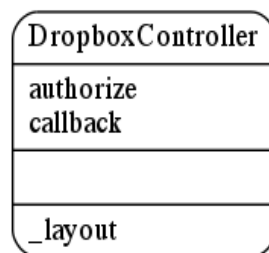


Ilustración 17 Controlador de Dropbox

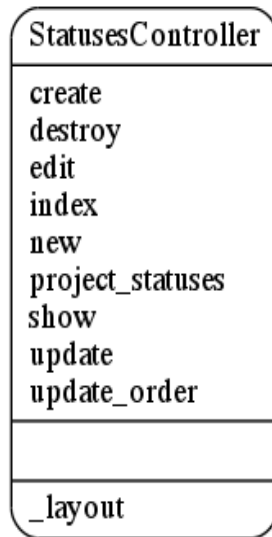


Ilustración 18 Controlador de los estados

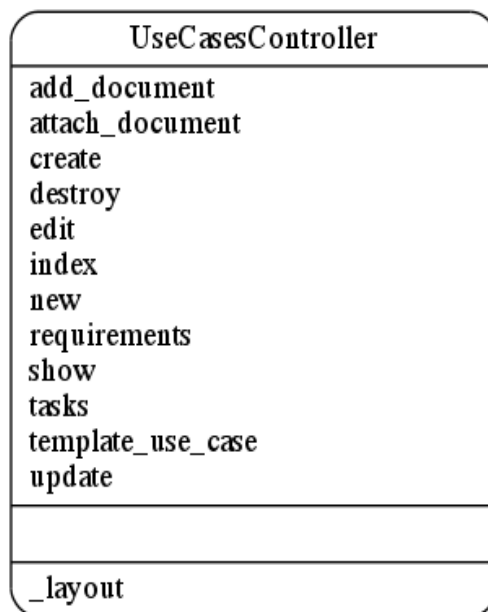


Ilustración 19 Controlador de los Casos de Uso

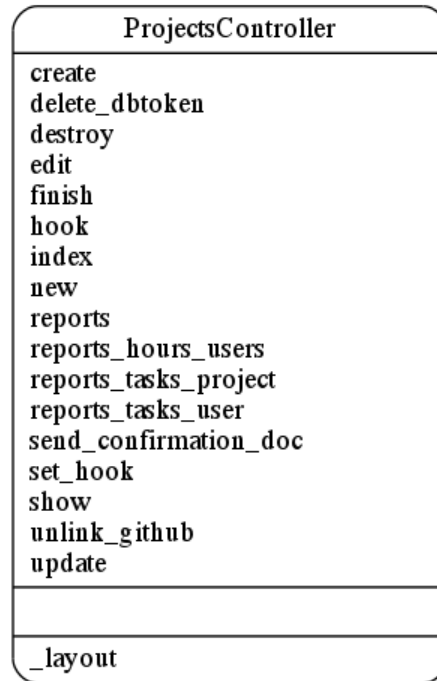


Ilustración 20 Controlador de Proyectos

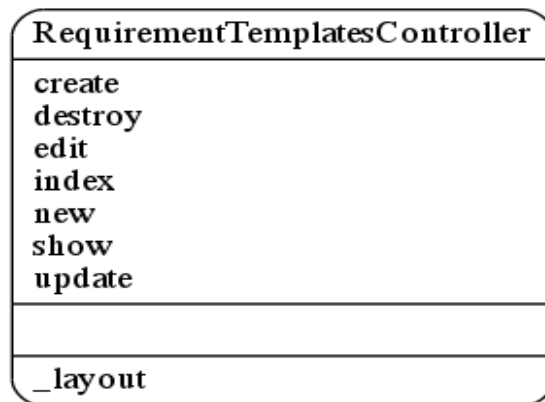


Ilustración 21 Controlador de los requerimientos de los Templates

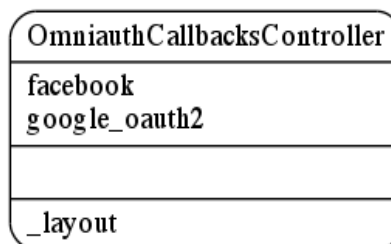


Ilustración 22 Controlador del Logeo de facebook y google

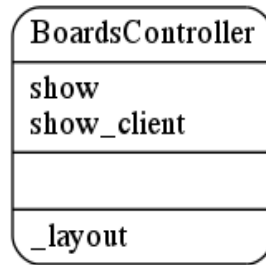


Ilustración 23 Controlador del Kanban



Ilustración 24 Controlador de las Tareas

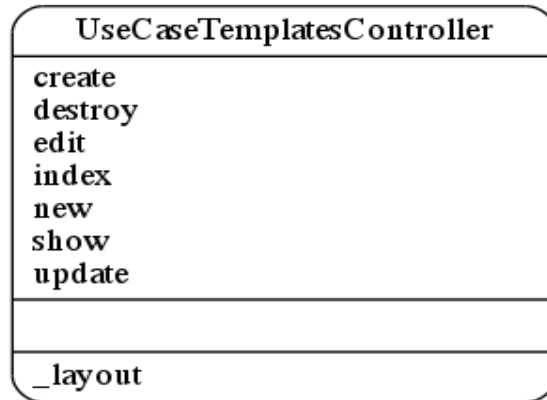


Ilustración 25 Controlador de los Templates de Casos de Uso

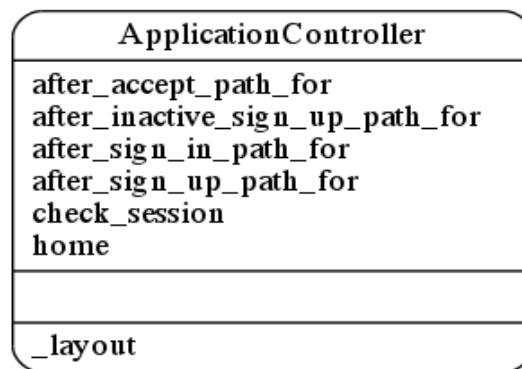


Ilustración 26 Controlador de la Aplicación

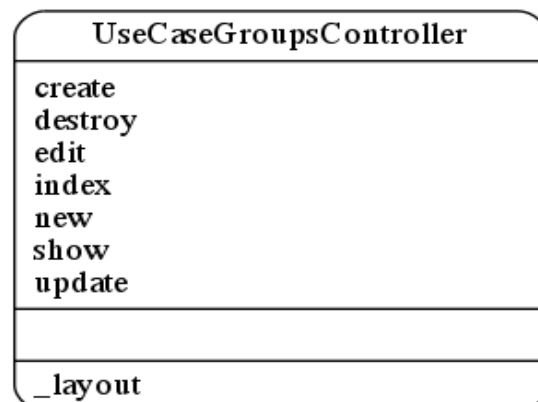


Ilustración 27 Controlador de los Grupos de Caso de Uso

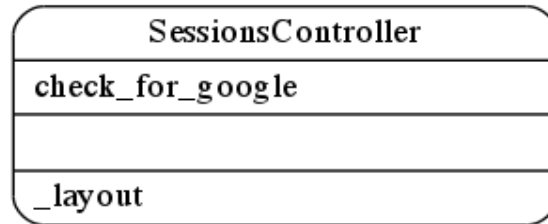


Ilustración 28 Controlador de las Sesiones

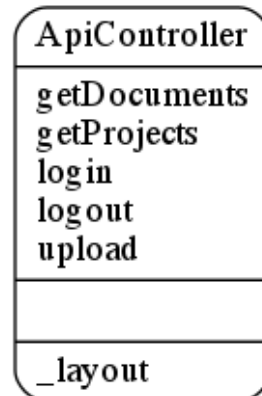


Ilustración 29 Controlador del API

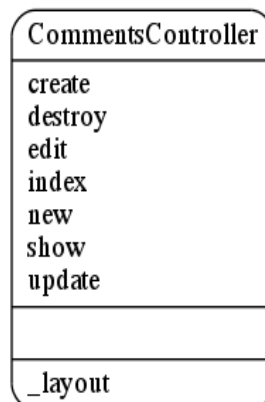


Ilustración 30 Controlador de los Comentarios

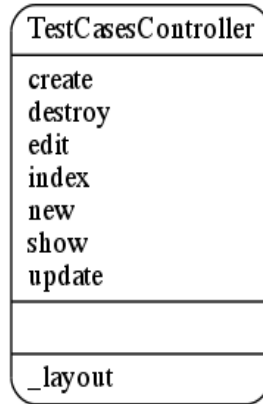


Ilustración 31 Controlador de los Casos de Tests

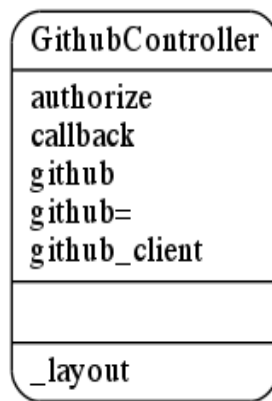


Ilustración 32 Controlador de Github

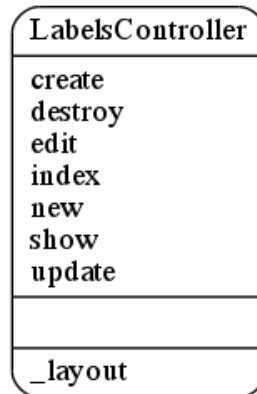


Ilustración 33 Controlador de las etiquetas

2.1.3.1 Interfaz de Usuario

Modelo de Navegación del Sistema

Actualmente esta implementado sólo un tipo de permiso y vista para los distintos usuarios. Por lo que existe solo un modelo de navegación que se presentara a continuación.

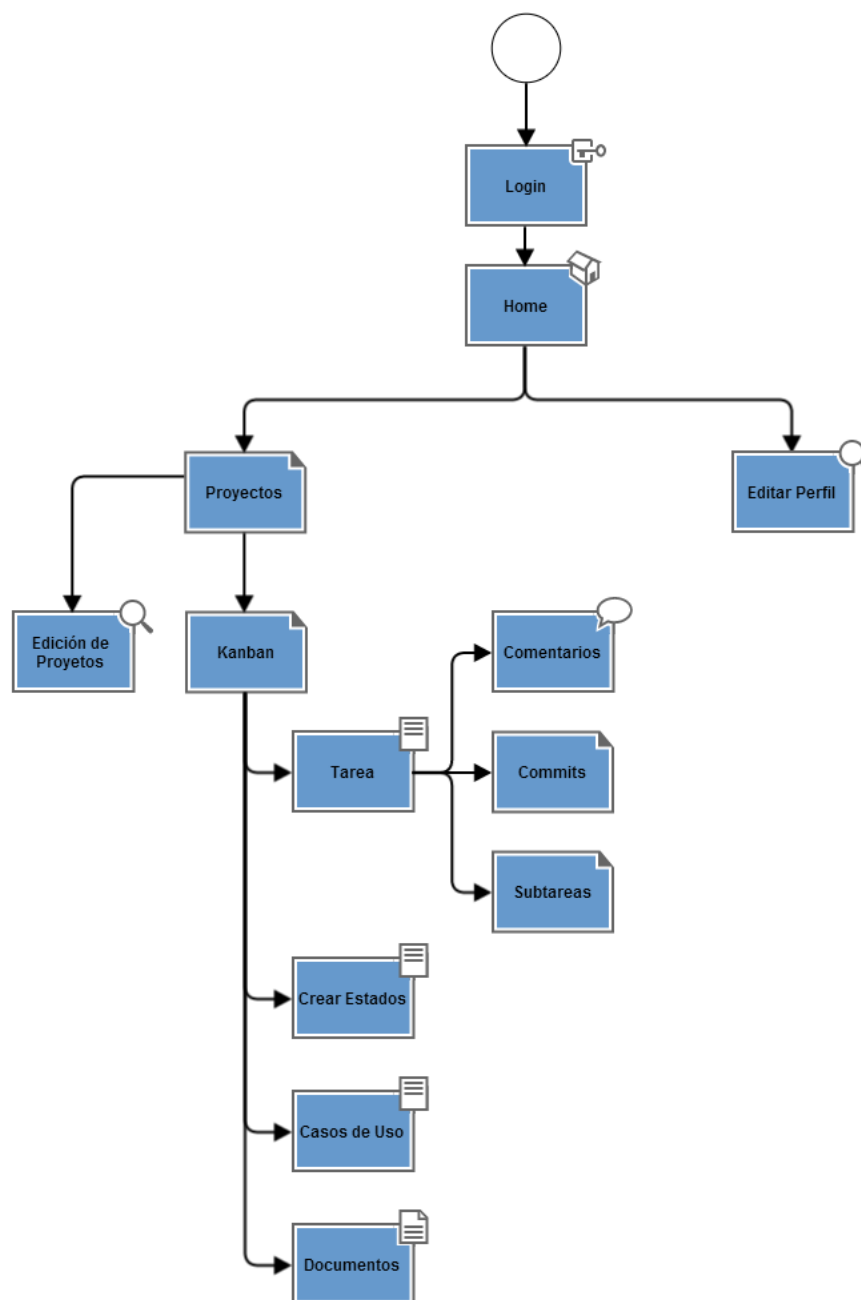


Ilustración 34 Modelo de Navegación

Simbología:

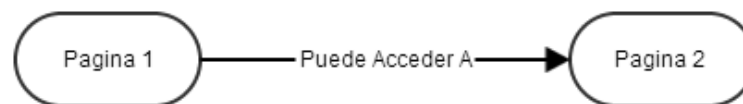


Ilustración 35 Simbología

Interfaz de Login del Usuario

SmartBoard Sign in Sign up

Sign in

Email

Password

Remember me ☐

[Sign in](#) [Google Login](#) [Facebook Login](#)

[Sign up](#)
[Forgot your password?](#)
[Didn't receive confirmation instructions?](#)
[Sign in with Google OAuth2](#)
[Sign in with Facebook](#)

Ilustración 36 Interfaz de Ingreso

En el siguiente diagrama se muestran las distintas posibilidades de ingreso del usuario a la página web.

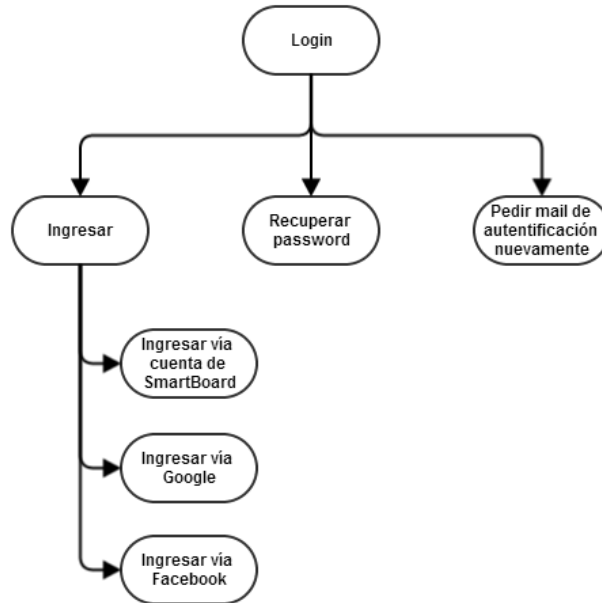


Ilustración 37 Flujo posible de ingreso.

Interfaz de Registro de Usuario

SmartBoard Sign in Sign up

Sign up

Email

Password

Password confirmation

[Sign up](#) [Google Login](#) [Facebook Login](#)

[Sign in](#)
[Forgot your password?](#)
[Didn't receive confirmation instructions?](#)
[Sign in with Google OAuth2](#)
[Sign in with Facebook](#)

Ilustración 38 Interfaz de Registro

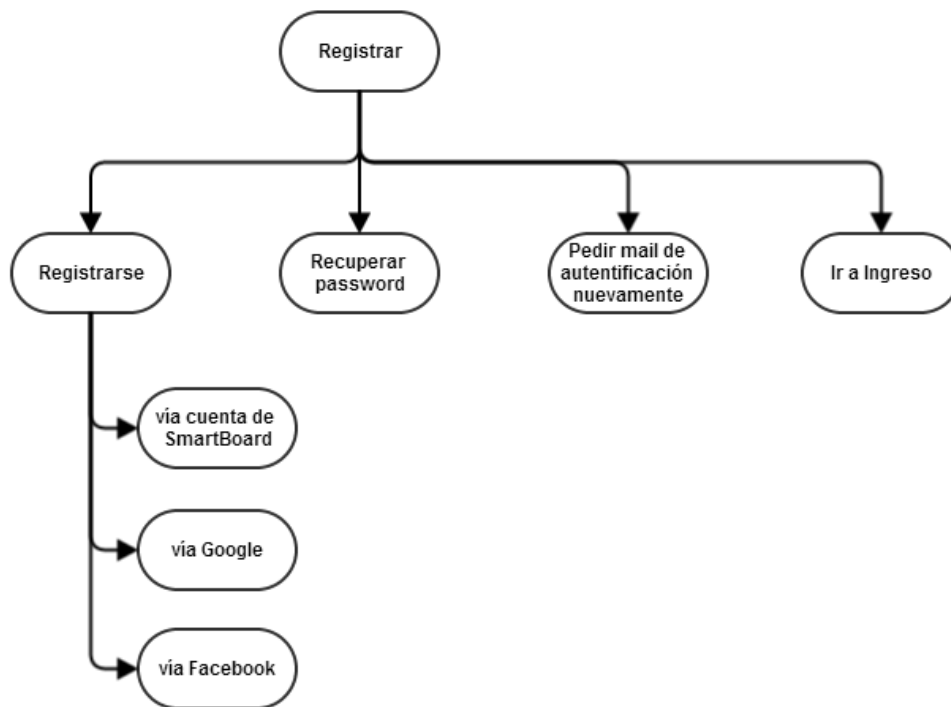


Ilustración 39 Flujo posible de Registro

Interfaz Proyectos de Usuario

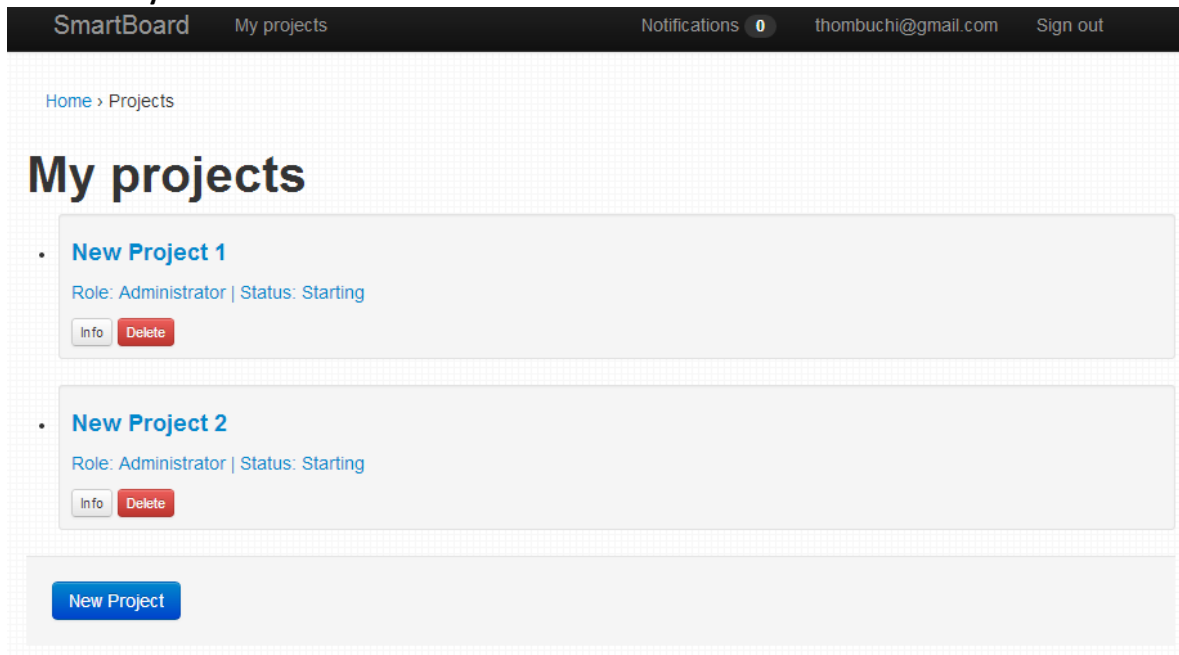


Ilustración 40 Interfaz de Proyectos

Interfaz Crear Proyecto

New

The image shows a web interface for creating a new project. The title "New" is prominently displayed at the top. Below it, there is a form with three main sections: "Name", "Description", and "Initial date". The "Name" field is a single-line text input. The "Description" field is a large, multi-line text area. The "Initial date" section consists of three dropdown menus for year, month, and day, currently showing "2013", "November", and "27". At the bottom of the form, there is a grey bar containing two buttons: "Create Project" (in blue) and "Cancel" (in grey).

Ilustración 41 Interfaz de creación de Proyectos

Interfaz Editar Proyecto

The image shows a web interface for editing a project. The top navigation bar includes the 'SmartBoard' logo, the current path 'My projects / New Project 1', an 'Add User' link, and user information 'thombuchi@gmail.com' with a 'Sign out' link. A notifications bell icon shows '0' notifications. On the left, a sidebar contains icons for a clipboard, documents, a user profile, a list, and a file. The main content area is titled 'Edit' and contains the following form elements:

- Name:** A text input field containing 'New Project 1'.
- Description:** A large, empty text area for project details.
- Initial date:** Three dropdown menus set to '2013', 'December', and '4'.
- Project status:** A dropdown menu set to 'Starting'.
- Buttons:** A blue 'Update Project' button and a grey 'Cancel' button.

Ilustración 42 Interfaz editar Proyectos

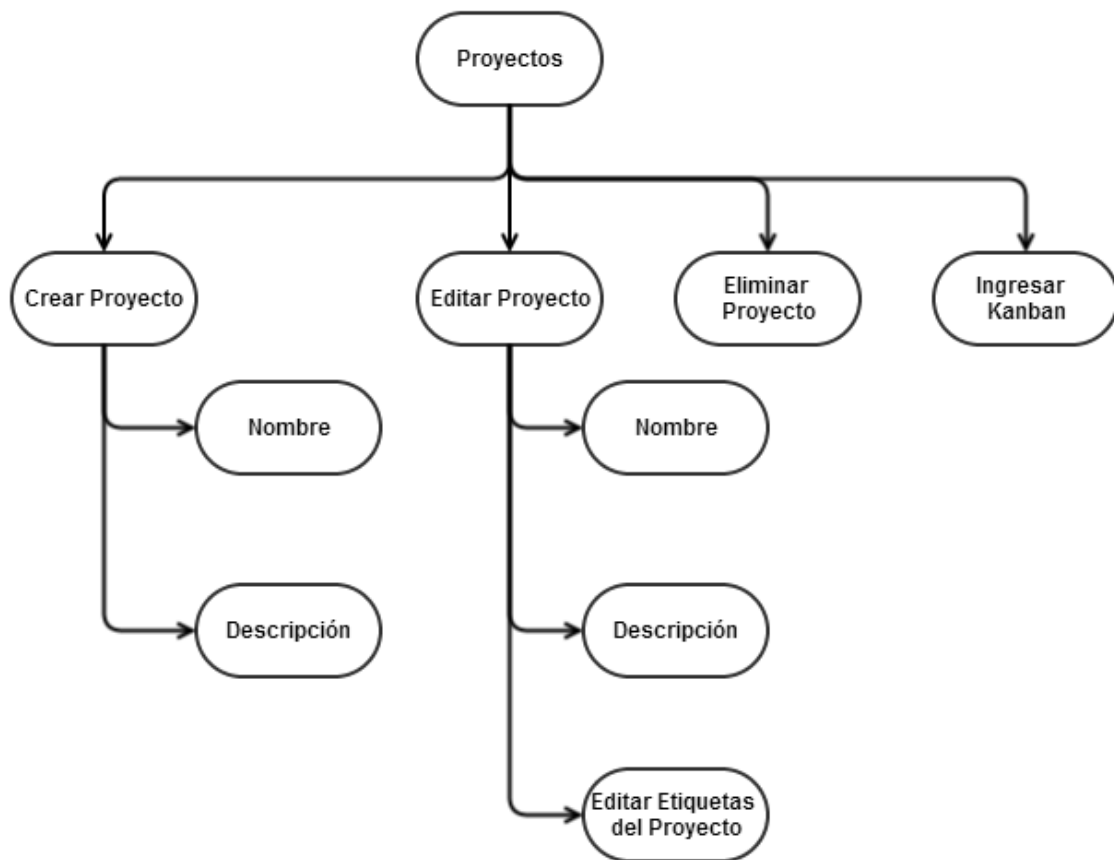


Ilustración 43 Diagrama de Flujo de Proyectos

Etiquetas

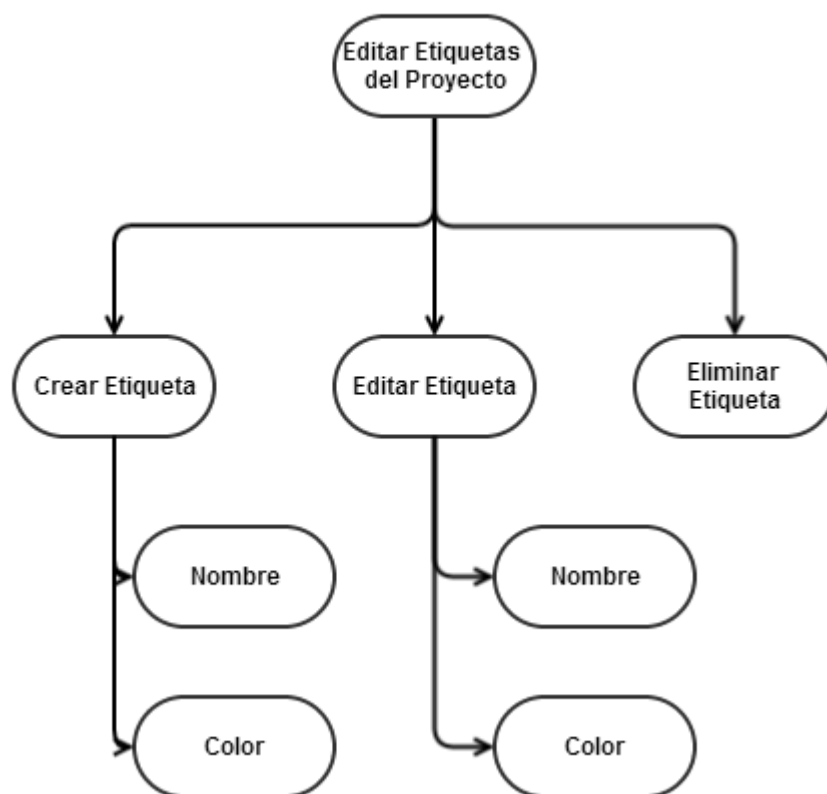


Ilustración 44 Diagrama de flujos de las Etiquetas

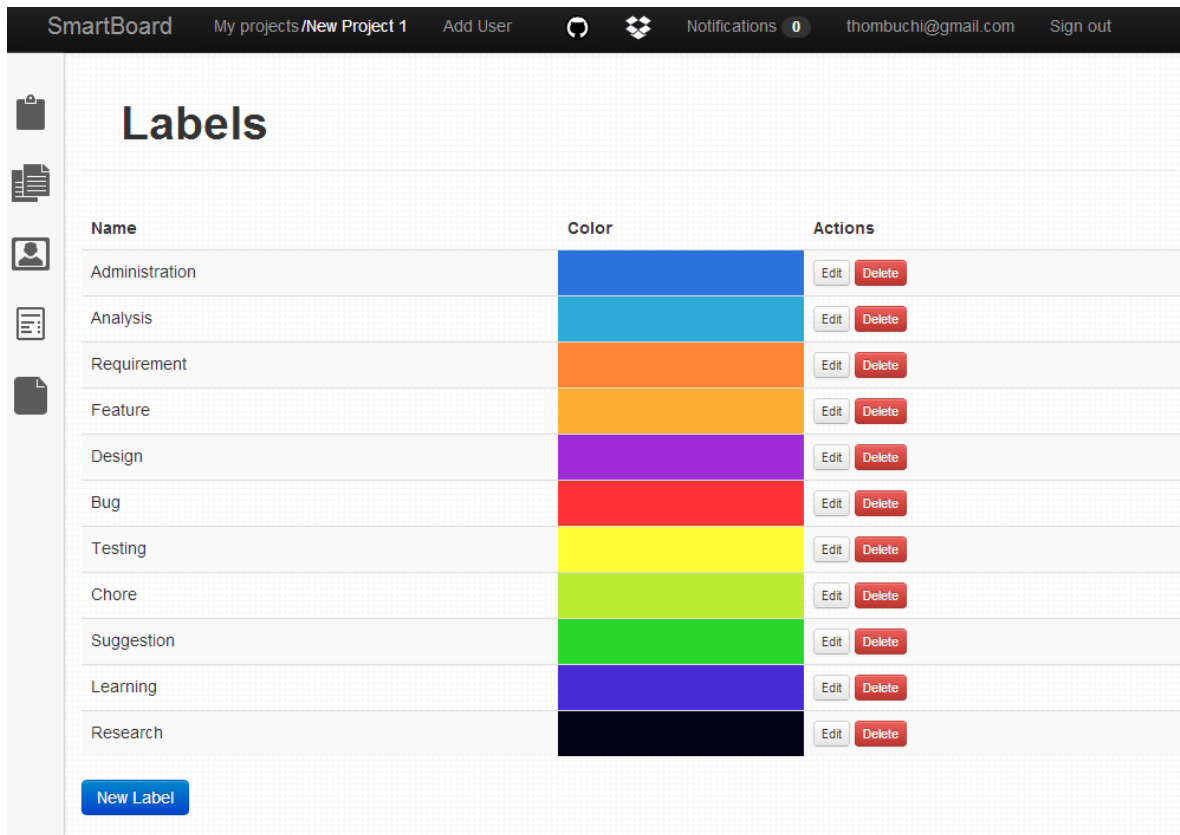


Ilustración 45 Interfaz de las Etiquetas del proyecto

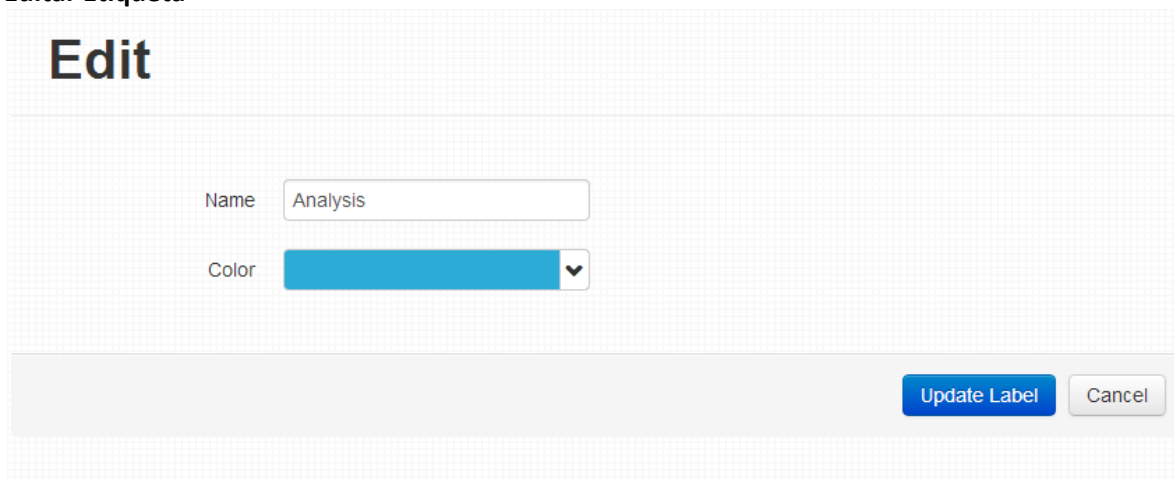
Crear Etiqueta



The 'Label' dialog box features a title bar with a close button (X). The main area contains a 'Name' text input field and a 'Color' dropdown menu currently set to blue. Below the dropdown is a color selection tool consisting of a square gradient preview and a vertical rainbow color bar with a slider. At the bottom right, there are two buttons: 'Create Label' (highlighted in blue) and 'Cancel'.

Ilustración 46 Interfaz de Cceación de una nueva Etiqueta

Editar Etiqueta



The 'Edit' dialog box has a title bar and a background grid. It contains a 'Name' text input field with the text 'Analysis' and a 'Color' dropdown menu set to blue. At the bottom right, there are two buttons: 'Update Label' (highlighted in blue) and 'Cancel'.

Ilustración 47 Interfaz de Editar una Etiqueta

Interfaz del Kanban

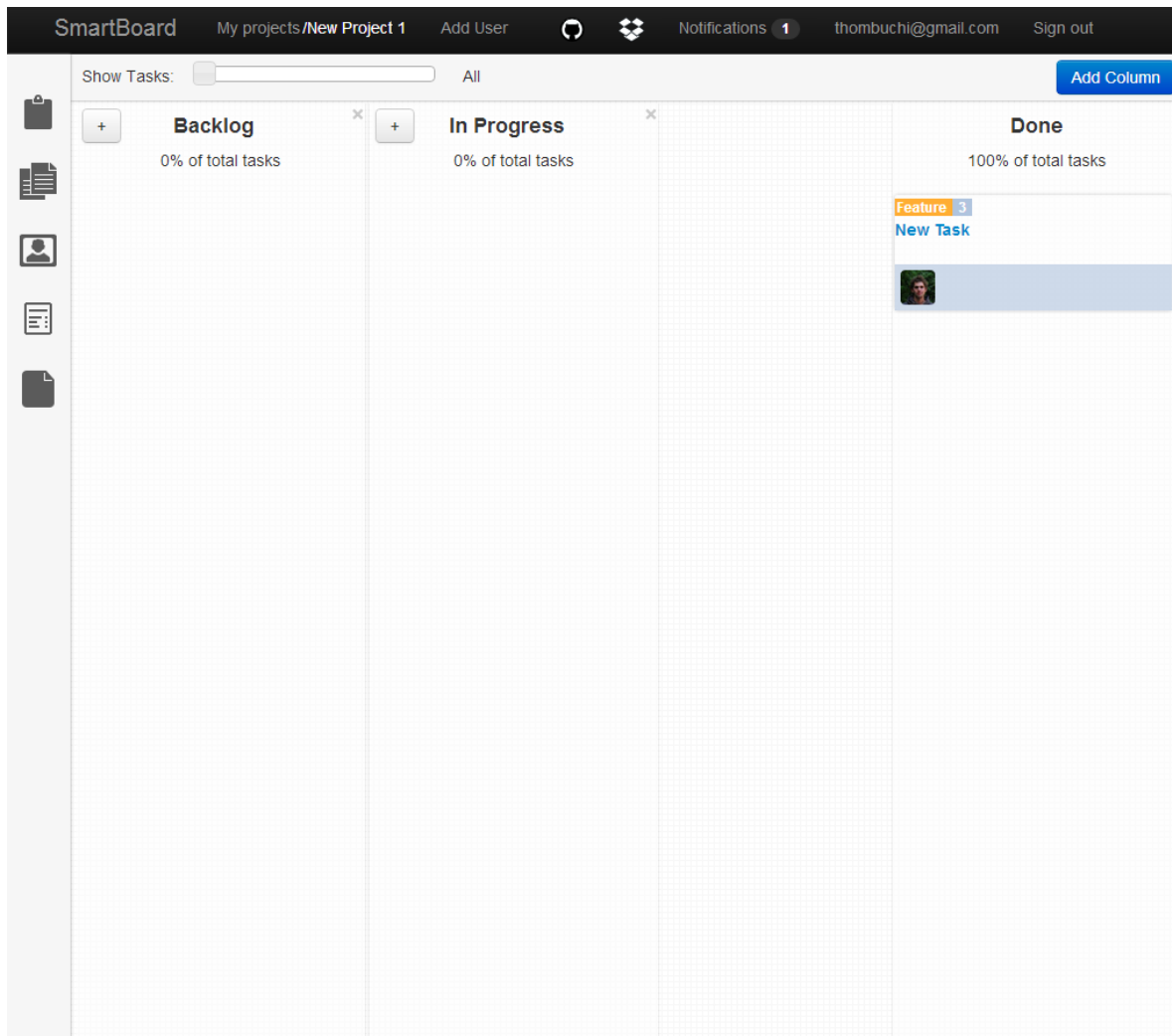


Ilustración 48 Intefaz del Kanban

El Kanban es la interfaz central de la aplicación, la que permite la mayoría de las funcionalidades del proyecto.

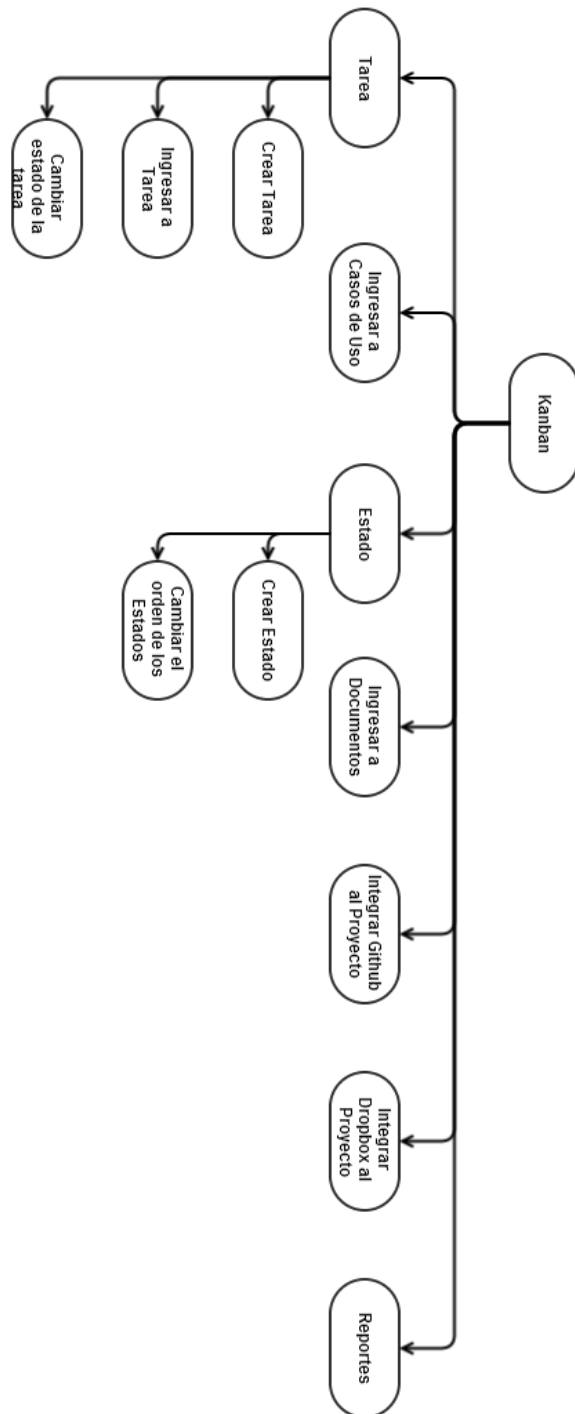
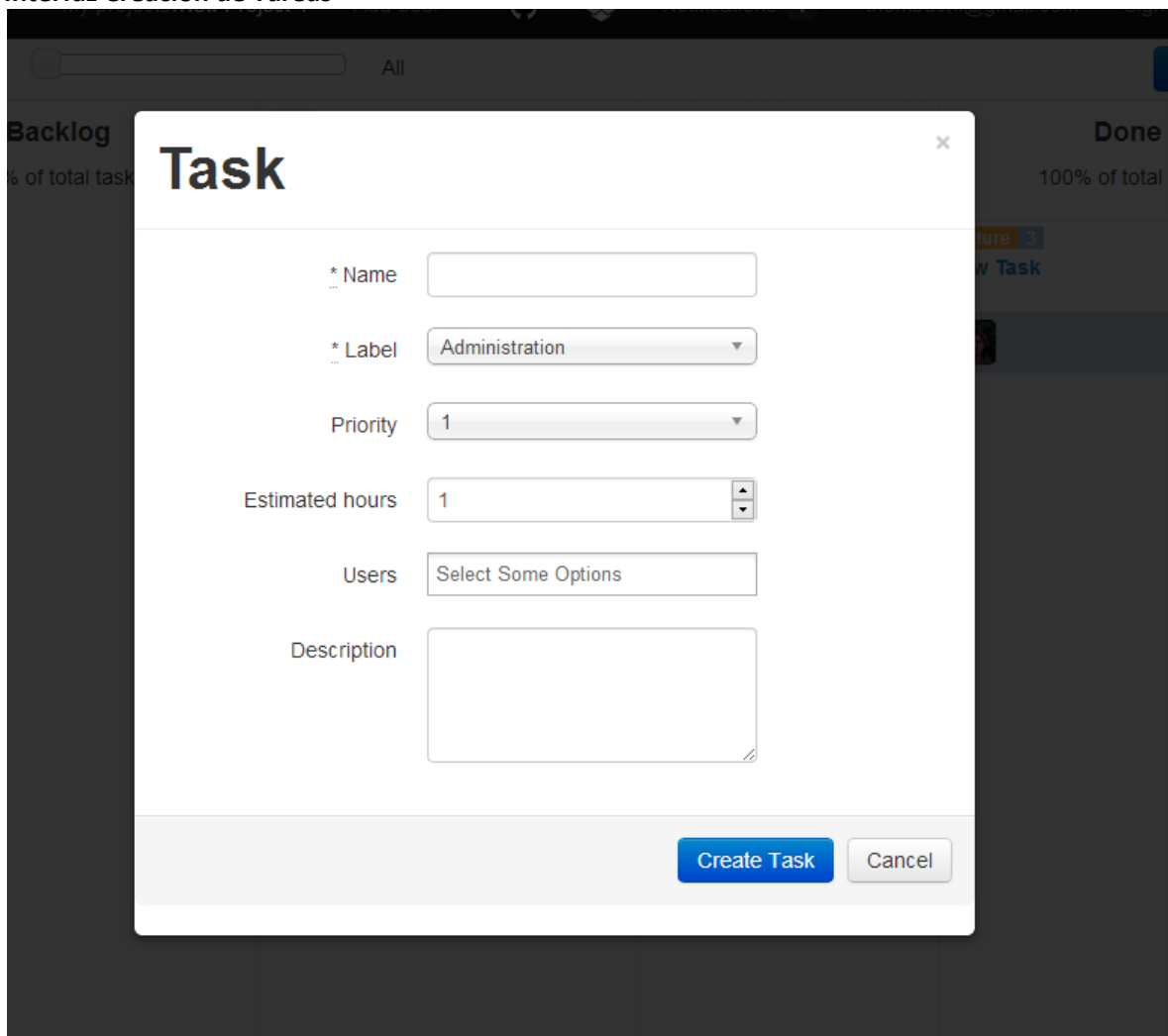


Ilustración 49 Diagrama de flujo de acciones posibles en el kanban

Interfaz Creación de Tareas



The image shows a 'Task' creation modal window overlaid on a blurred background of a project management interface. The modal has a title bar with a close button (X) in the top right corner. The title 'Task' is displayed in a large, bold font. Below the title, there are several input fields and dropdown menus for task configuration:

- * Name:** A text input field.
- * Label:** A dropdown menu with 'Administration' selected.
- Priority:** A dropdown menu with '1' selected.
- Estimated hours:** A text input field with '1' and a small up/down arrow icon to its right.
- Users:** A text input field with the placeholder text 'Select Some Options'.
- Description:** A larger text area for a detailed description.

At the bottom right of the modal, there are two buttons: a blue 'Create Task' button and a grey 'Cancel' button.

Ilustración 50 Interfaz de creación de Tareas

Interfaz Ver Tarea

New Task

Id:
#13

Priority:
1

Task type:
Feature

Estimated hours:
3

Requirement
-

Use Case
-

Owner:
thombuchi@gmail.com

Description

Comments

Commits

Documents

Subtasks

Report Hours

Close

Edit

Lock

Delete

Ilustración 51 Interfaz ver Tarea

Interfaz ver Comentarios

New Task

Id:
#13

Priority:
1

Task type:
Feature

Estimated hours:
3

Requirement
-

Use Case
-

Owner:
thombuchi@gmail.com

Description

Comments

Commits

Documents

Subtasks

thombuchi@gmail.com

New comment

X

New Message

Comment

Report Hours

Close

Edit

Lock

Delete

Ilustración 52 Interfaz ver Comentarios

Interfaz Commits

New Task

Id:
#13

Priority:
1

Task type:
Feature

Estimated hours:
3

Requirement
-

Use Case
-

Owner:
thombuchi@gmail.com

Description

Comments

Commits

Documents

Subtasks

Report Hours

Close

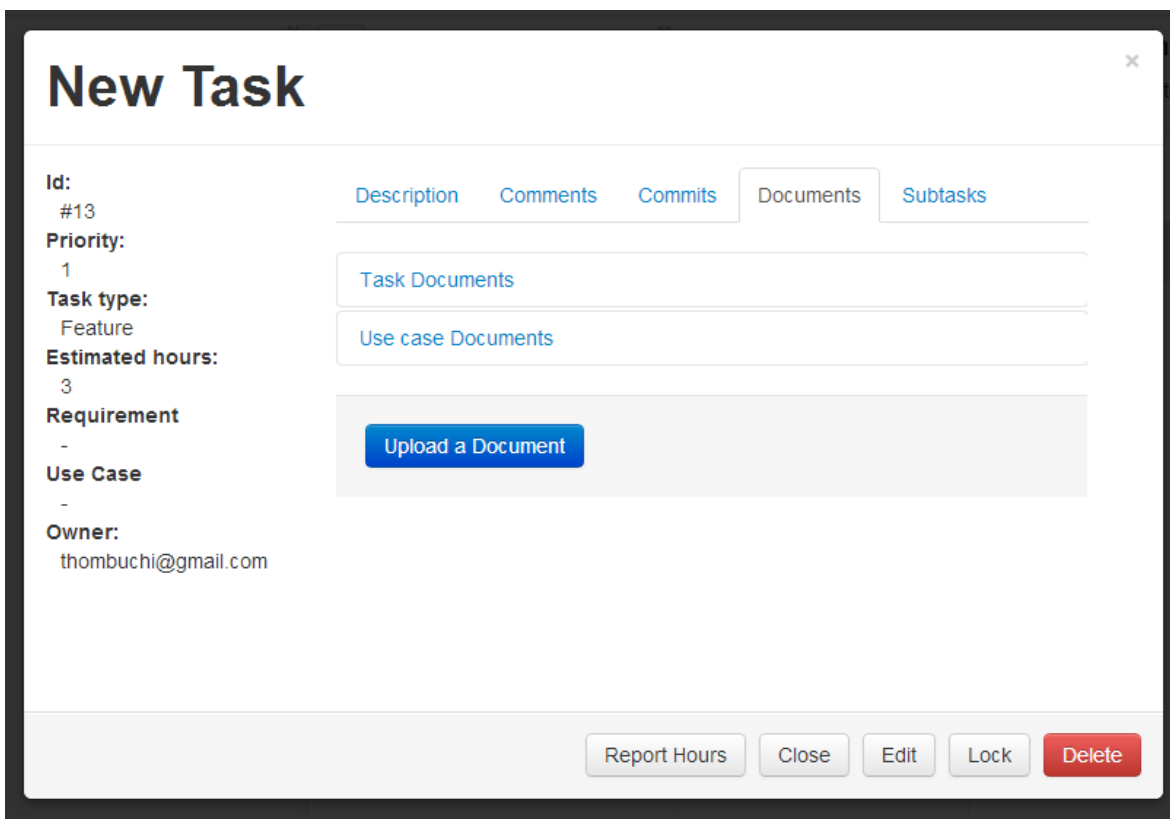
Edit

Lock

Delete

Ilustración 53 Interfaz ver Commits

Interfaz Ver Documentos Asociados



The 'New Task' dialog box features a title bar with a close button. On the left, a sidebar lists task attributes: Id: #13, Priority: 1, Task type: Feature, Estimated hours: 3, Requirement: -, Use Case: -, and Owner: thombuchi@gmail.com. The main area has five tabs: Description, Comments, Commits, Documents (selected), and Subtasks. Under the 'Documents' tab, there are two text input fields containing 'Task Documents' and 'Use case Documents'. Below these is a blue 'Upload a Document' button. At the bottom right, a row of buttons includes 'Report Hours', 'Close', 'Edit', 'Lock', and a red 'Delete' button.

New Task

Id: #13
Priority: 1
Task type: Feature
Estimated hours: 3
Requirement: -
Use Case: -
Owner: thombuchi@gmail.com

Description Comments Commits Documents Subtasks

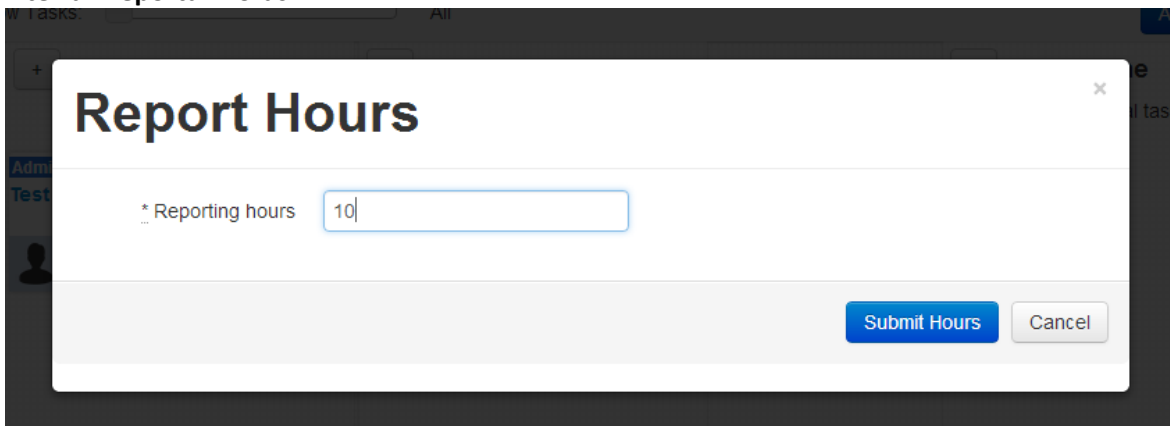
Task Documents
Use case Documents

Upload a Document

Report Hours Close Edit Lock Delete

Ilustración 54 Interfaz Ver Documentos Asociados

Interfaz Reportar Horas



The 'Report Hours' dialog box has a title bar with a close button. It contains a single text input field labeled '* Reporting hours' with the value '10'. At the bottom right, there are two buttons: a blue 'Submit Hours' button and a 'Cancel' button.

Report Hours

* Reporting hours 10

Submit Hours Cancel

Ilustración 55 Interfaz Reportar Horas

Interfaz Subtareas

New Task

Id:
#13

Priority:
1

Task type:
Feature

Estimated hours:
3

Requirement
-

Use Case
-

Owner:
thombuchi@gmail.com

DescriptionCommentsCommitsDocumentsSubtasks

Not Ready

New SubTask:
New

X

Create SubTask

Report Hours

Close

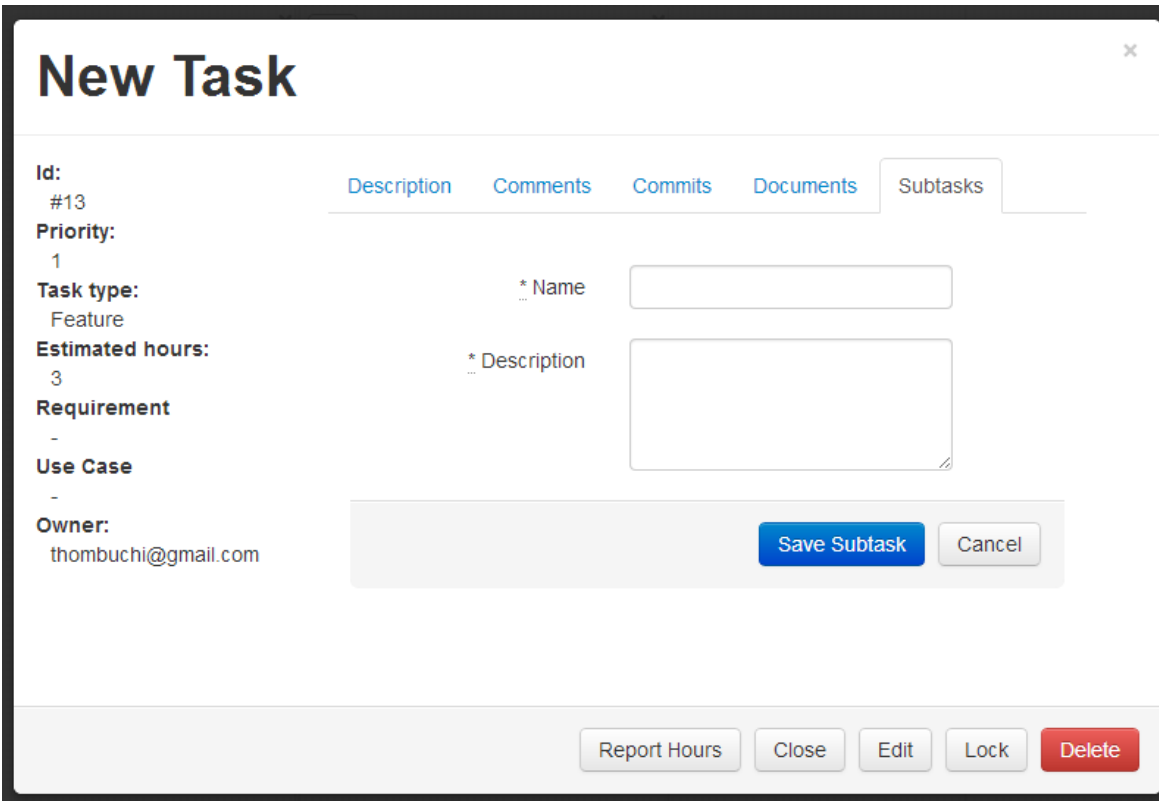
Edit

Lock

Delete

Ilustración 56 Interfaz ver Subtareas

Interfaz Crear Subtarea



The 'New Task' dialog box features a title bar with a close button. It has a sidebar on the left with fields for 'Id: #13', 'Priority: 1', 'Task type: Feature', 'Estimated hours: 3', 'Requirement: -', 'Use Case: -', and 'Owner: thombuchi@gmail.com'. The main area has tabs for 'Description', 'Comments', 'Commits', 'Documents', and 'Subtasks'. The 'Subtasks' tab is active, showing a form with a '* Name' field and a '* Description' text area. At the bottom right of the main area are 'Save Subtask' and 'Cancel' buttons. A footer bar contains 'Report Hours', 'Close', 'Edit', 'Lock', and 'Delete' buttons.

New Task

Id: #13

Priority: 1

Task type: Feature

Estimated hours: 3

Requirement: -

Use Case: -

Owner: thombuchi@gmail.com

Description Comments Commits Documents Subtasks

* Name

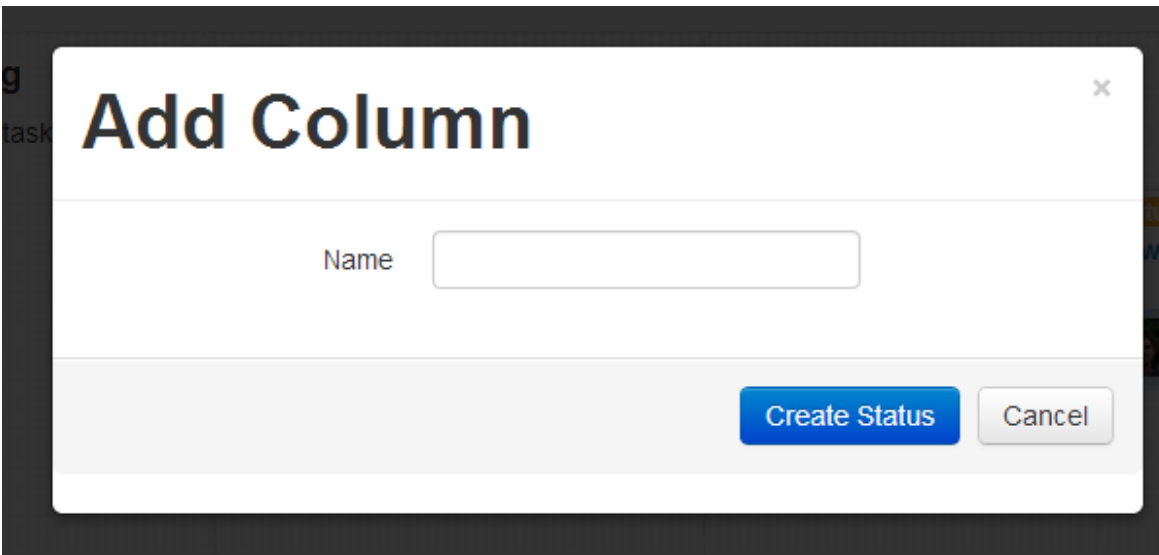
* Description

Save Subtask Cancel

Report Hours Close Edit Lock Delete

Ilustración 57 Interfaz Crear Subtarea

Interfaz Crear nuevo estado



The 'Add Column' dialog box has a title bar with a close button. It contains a single 'Name' text input field. At the bottom right are 'Create Status' and 'Cancel' buttons.

Add Column

Name

Create Status Cancel

Ilustración 58 Interfaz Agregar Columna

Interfaz Documentos del Proyecto

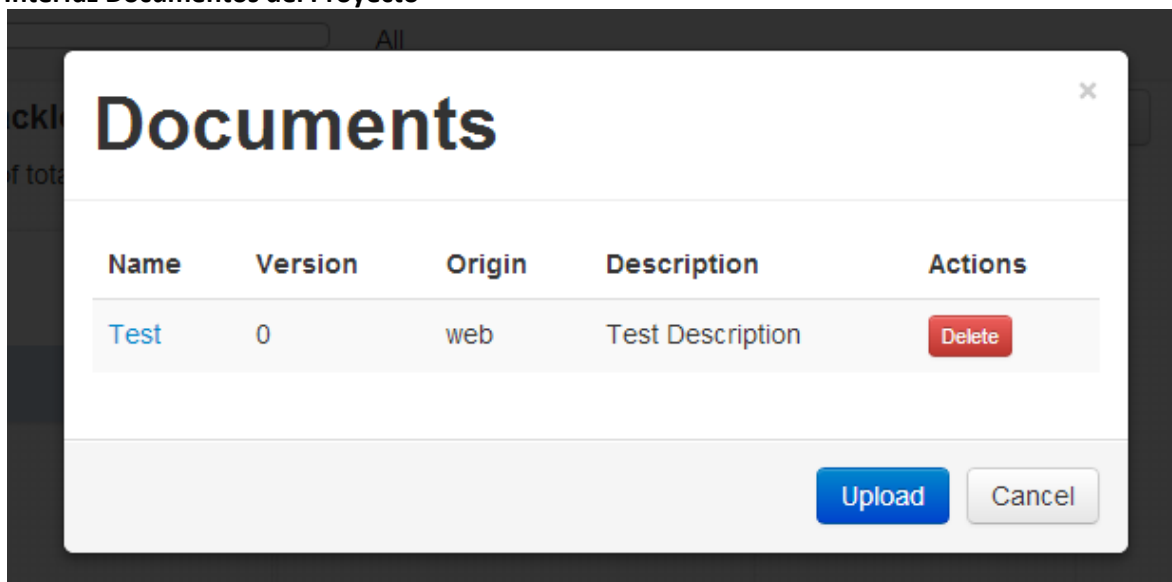


Ilustración 59 Interfaz ver Documentos del Proyecto

Interfaz Reportes

The image shows a web interface titled "Reports". At the top, there are two date selection fields: "Initial Date" with the value "11/20/2013" and "Final Date" with the value "11/27/2013". Below these are four navigation tabs: "User Hours", "User Tasks", "Label Tickets", and "Use Case Structure". The "User Tasks" tab is currently selected. The main content area is a large, empty rectangular box. On the left side of this box, the email address "mbuchi@gmail.com" is visible.

Ilustración 60 Interfaz Reportes

Interfaz Casos de Uso

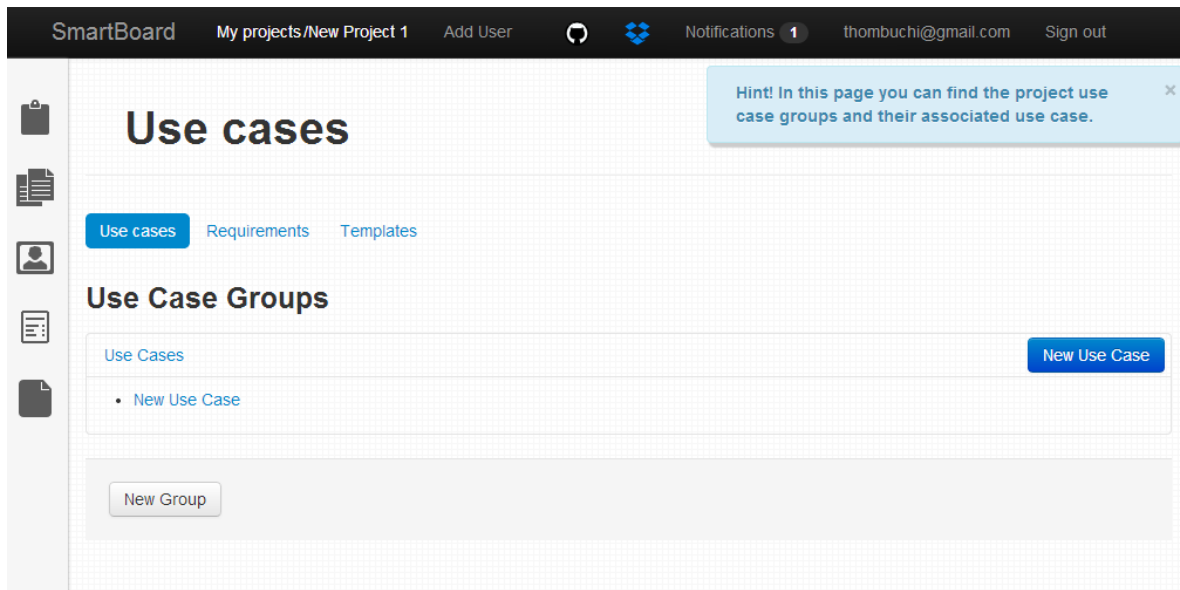


Ilustración 61 Interfaz de Casos de Uso

Diagrama de la interfaz de caso de uso:

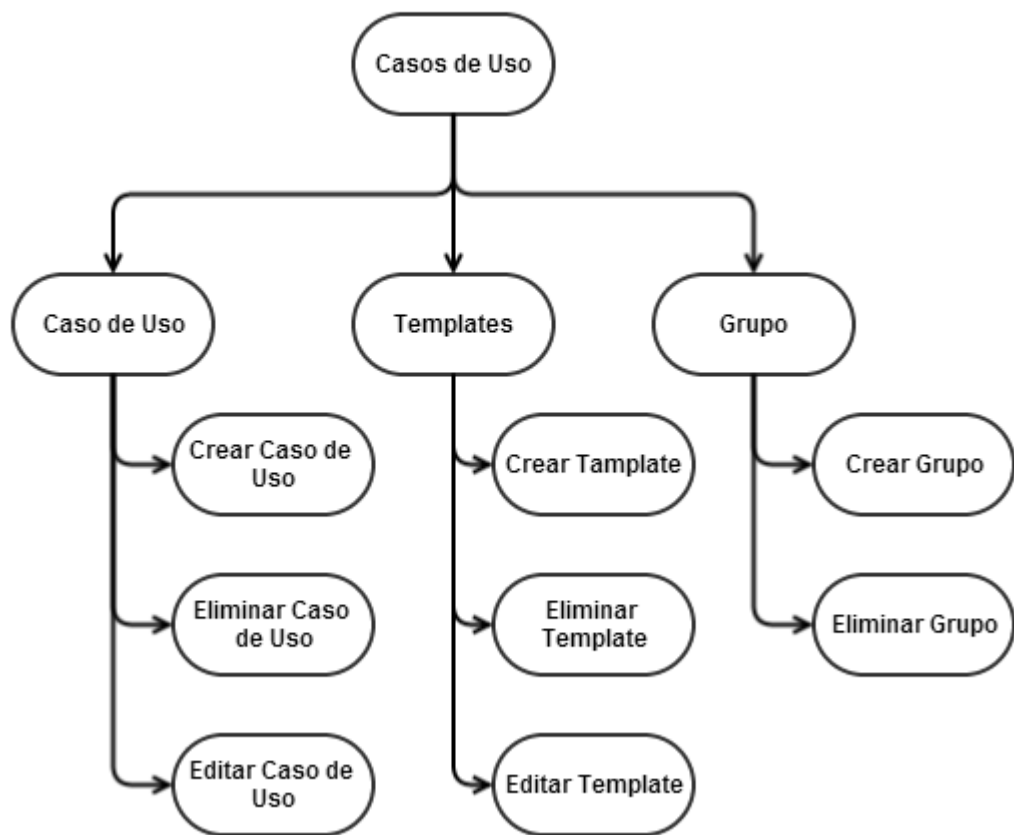


Ilustración 62 Diagrama del Flujo posible de en Casos de Uso

Interfaz Crear Caso de Uso

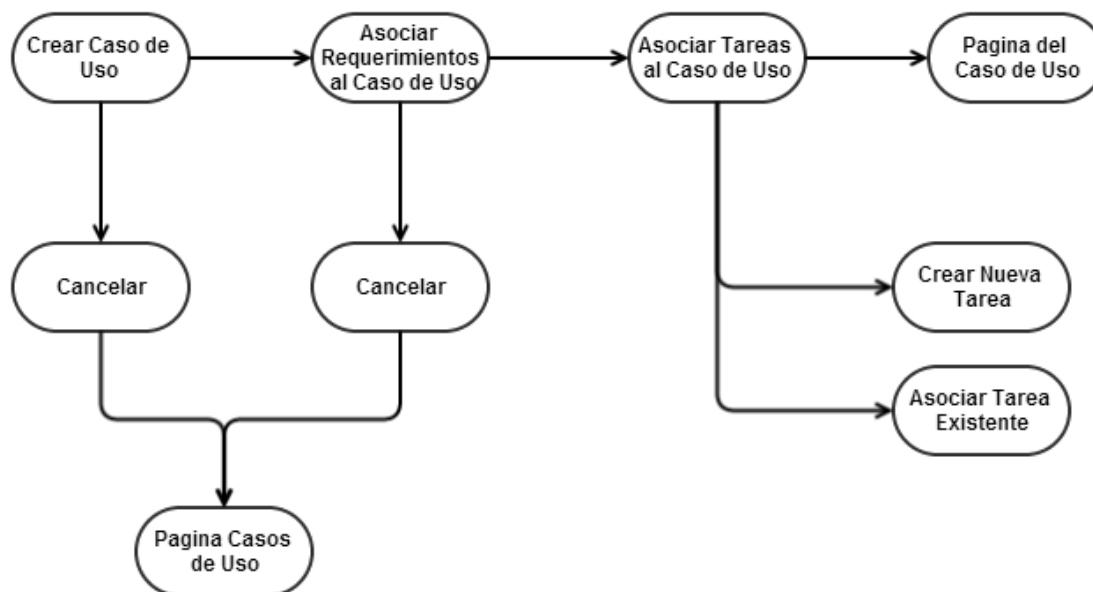


Ilustración 63 Flujo Crear Caso de Uso

Interfaz Crear Caso de Uso

SmartBoard My projects / New Project 1 Add User Notifications 1 thombuchi@gmail.com Sign out

New Use Case

1 Use Case 2 Requirements 3 Tasks

* Name

* Use case template

Primary Actor

Basic Flow

Alternate Flow 1

Alternate Flow 2

Alternate Flow 3

Ilustración 64 Interfaz Crear Caso de Uso

Interfaz Agregar Requerimientos al Caso de Uso

The screenshot shows the 'Add Requirements to Use Case' interface in the SmartBoard application. The top navigation bar includes 'SmartBoard', 'My projects / New Project 1', 'Add User', a refresh icon, a settings icon, 'Notifications 1', 'thombuchi@gmail.com', and 'Sign out'. The main title is 'Add Requirements to Use Case New Use Case'. Below the title are three tabs: '1 Use Case', '2 Requirements' (which is active), and '3 Tasks'. A 'New Requirement' button is on the left, and a 'Delete' button is on the right. Below these are two buttons: 'Add Existing Requirement' and 'New Requirement'. At the bottom are three buttons: 'Back', 'Continue' (highlighted in blue), and 'Cancel'.

Ilustración 65 Interfaz Asociar Requerimientos al caso de Uso

Interfaz Crear Requerimiento

The screenshot shows a modal form titled 'Requirement' with a close button (X) in the top right corner. The form has two input fields: '* Name' and 'Description'. The 'Description' field is a larger text area. At the bottom of the form are two buttons: 'Create Requirement' (highlighted in blue) and 'Cancel'.

Ilustración 66 Interfaz Crear Nuevo Requerimiento

Interfaz Agregar Tareas al Caso de Uso

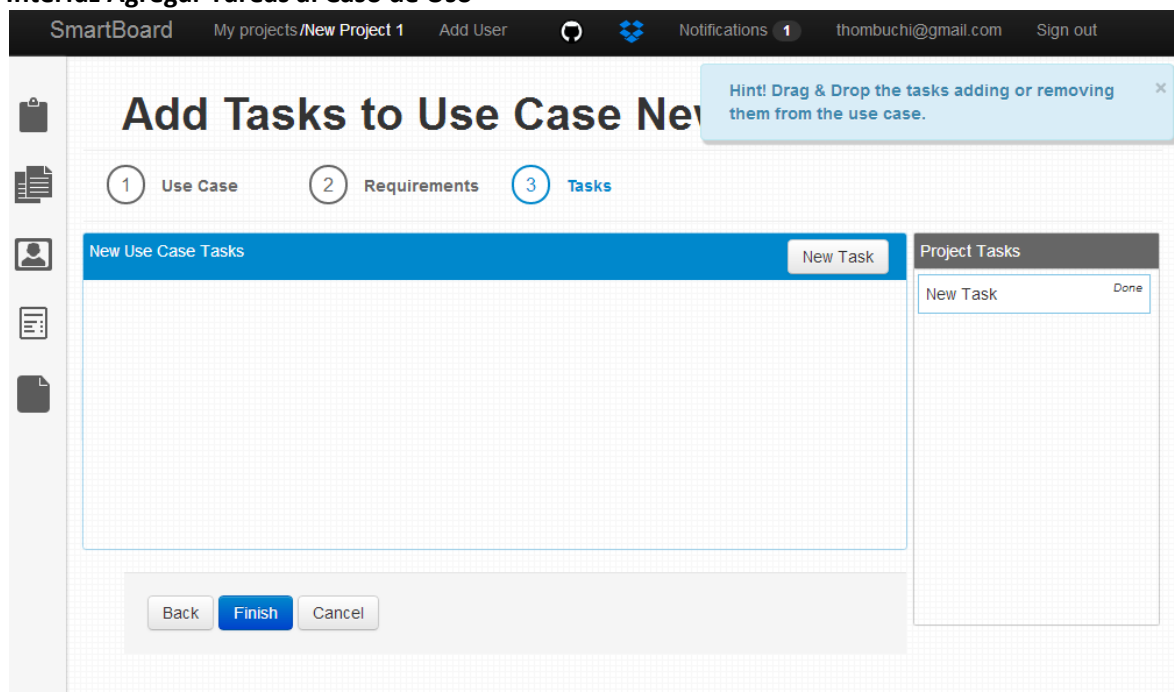


Ilustración 67 Interfaz Asociar Tareas al Caso de Uso

Interfaz Crear Template

SmartBoard

My projects / New Project 1

Add User

Notifications 1

thombuchi@gmail.com

Sign out

New

Use casesRequirementsTemplates

Instructions: Select the data type and click the '+' button to add a field to the Template.

Name

Name

Input Type

input

Data Type

string

Input Values

Add Atributte +

submit

Cancel

Ilustración 68 Interfaz Crear Template

Interfaz crear nuevo Grupo de Casos de Uso

New

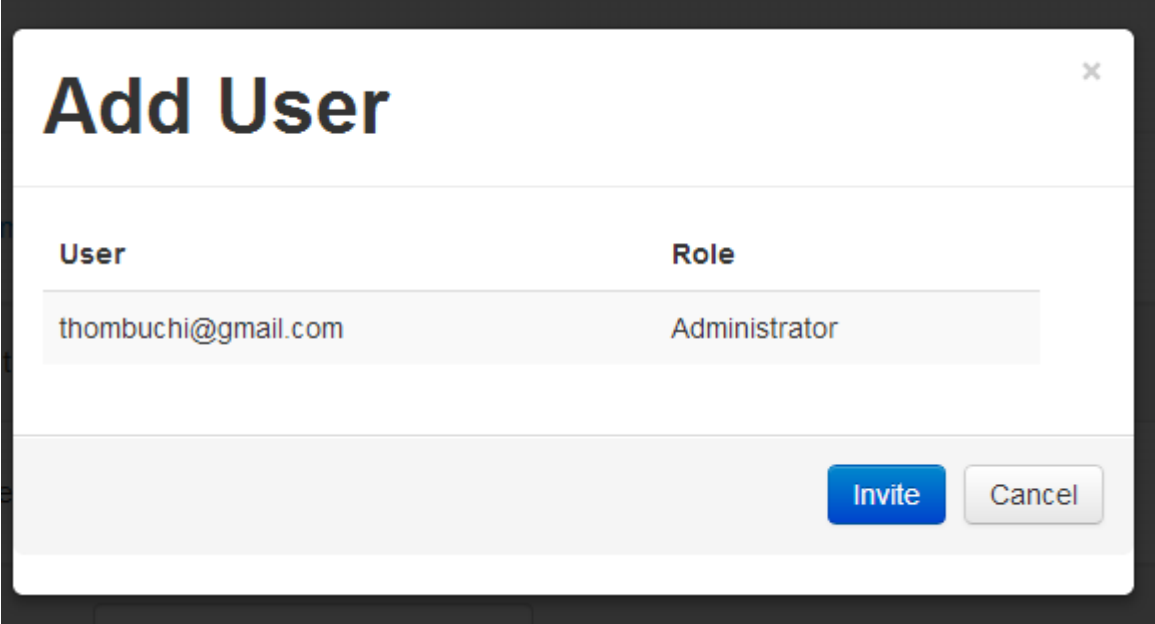
* Name

Create Use case group

Cancel

Ilustración 69 Interfaz crear nuevo Grupo de Casos de Uso

Interfaz Agregar Usuario

A dialog box titled "Add User" with a close button (X) in the top right corner. It contains a table with two columns: "User" and "Role". The "User" column contains the email address "thombuchi@gmail.com" and the "Role" column contains the text "Administrator". At the bottom right of the dialog, there are two buttons: "Invite" (blue) and "Cancel" (gray).

User	Role
thombuchi@gmail.com	Administrator

Invite Cancel

Ilustración 70 Interfaz Agregar Usuario a Proyecto

2.1.3.2 Servicios de Sistema

Ruby On Rails mantiene un sistema rígido MVC por lo que su base es mantener estrictamente el uso de Vistas, Controladores y el Modelo en sí. Por lo que el Modelo es la capa en la cual se tiene la estructura lógica de la aplicación, las vistas son la presentación de cada interfaz de usuario y cada controlador se encarga de hacer la conexión entre el modelo y las vistas para el usuario. Por lo que cada conexión con el servidor se hace vía un servicio web estándar.

Los subsistemas que se mostrarán a continuación será uno general, que engloba el funcionamiento general de los controladores de la plataforma, que es repetitivo a lo largo de las distintas entidades del modelo.

Subsistema General

Al hacer una petición http de tipo get o post, se tiene por default las peticiones de crear un nuevo elemento a un modelo, editarlo, mostrarlo, mostrar todos, y borrar. Estos son direccionados al controlador específico de un modelo, y él se encarga de proporcionar las vistas y manejar los datos pedidos.

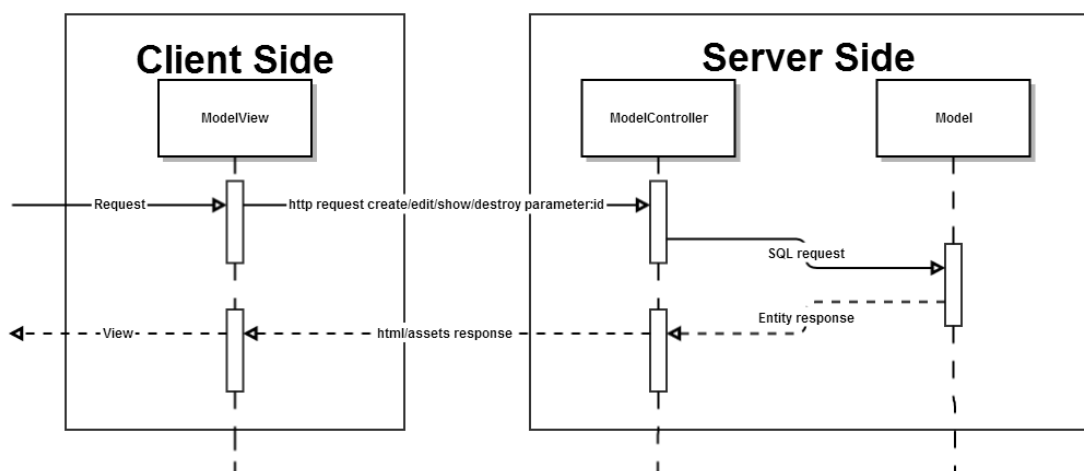


Ilustración 71 Diagrama de flujo general de la aplicación

3. Vista de Implementación

3.1.1 Estructura de la Aplicación

La aplicación web fue desarrollada usando ruby on rails, en donde se programa utilizando código Ruby, Javascript, CSS y HTML. El framework obliga a mantener una estructura muy rígida MVC la cual se mantuvo en este proyecto.

Se mantiene una estructura de archivos de vistas (vista.html.erb también conocidos como Action View), controladores (model_controller.rb), helpers (model_helper.rb) y modelos (model.rb conocido como Active Record). Además se tienen todos los assets que incluyen los archivos de estilos (estilo.css), javascript (javascript.js) y los de imágenes (.png, .jpg, .gif, etc...).

3.1.2 Arquitectura de Implementación

Se presenta a continuación el diagrama de arquitectura de una aplicación Ruby on Rails, que es exactamente la que fue usada en este proyecto

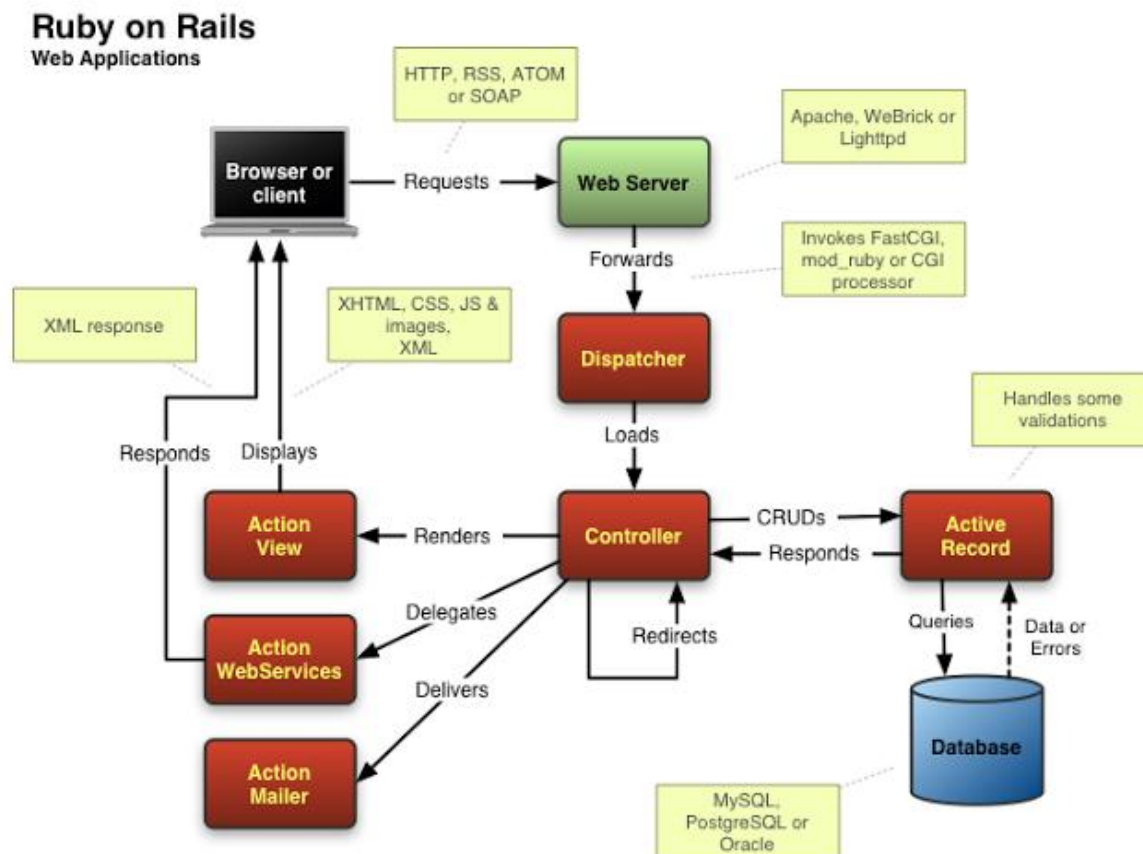


Ilustración 72 Arquitectura Ruby on Rails (Fuente: <http://www.sentex.net/~pkomisar/Ruby/Rails2.png>)

Para la aplicación se usaron distintas gemas algunas que son sólo para el ambiente de desarrollo y otras para la producción del software. A continuación se listará las gemas utilizadas y cuál es su uso:

Gemas	Descripción
Coffee-rails	Gema que compila lenguaje coffee a lenguaje javascript. https://github.com/rails/coffee-rails
Bootstrap-sass	Compilador de código sass para la versión de bootstrap. https://github.com/thomas-mcdonald/bootstrap-sass
uglifyer	https://github.com/lautis/uglifyer
Bootstrap-modal-rails	Gema para implementar los modals de bootstrap. https://github.com/vicentereig/bootstrap-modal-rails
Jquery-rails	Instala jquery en la aplicación. https://github.com/indirect/jquery-rails
Jquery-ui-rails	Instala jquery ui. https://github.com/joliss/jquery-ui-rails
Bootstrap_helper	Ayuda al manejo de bootstrap https://github.com/xdite/bootstrap-helper
devise	Gema para manejar autorización y sesiones de usuarios. https://github.com/plataformatec/devise
devise_invitable	
Omniauth-google-oauth2	Gema para ingresar usando una cuenta de google. https://github.com/zquestz/omniauth-google-oauth2
Omniauth-facebook	Gema para ingresar usando una cuenta de facebook. https://github.com/mkdynamic/omniauth-facebook
Cancan	Gema para permisos de usuarios. https://github.com/ryanb/cancan
Sass-rails	Gema para compilar idioma sass a css. https://github.com/rails/sass-rails
compass	
Compass-rails	https://github.com/Compass/compass-rails
Jquery-turbolinks	https://github.com/kosnocomp/jquery.turbolinks
turbolinks	Mejora el cargado de páginas, al no recargar los assets cargados anteriormente. https://github.com/rails/turbolinks
Twitter-bootstrap-rails	Agrega css de bootstrap. https://github.com/seyhunak/twitter-bootstrap-rails

	rails
Simple_form	Gema que ayuda a manejar los componentes para crear forms. https://github.com/plataformatec/simple_form
Flash_render	https://github.com/adamhunter/flash_render
Dropbox-sdk	Gema con el sdk de dropbox que permite integrar dropbox y subir o bajar archivos. https://www.dropbox.com/developers/core
Github_api	Gema para la integración de github. http://developer.github.com/v3/libraries/
Gretel	Gema para el fácil manejo de breadcrumbs con la ubicación actual. https://github.com/lassebunk/gretel
Chosen-rails	Gema para usar select visualmente mejores. https://github.com/tsechingho/chosen-rails
Best_in_place	https://github.com/bernat/best_in_place

Tabla 1 Gemas Generales

Gemas para desarrollo y testeo	Descripción
Sqlite3	Base de datos.
Rspec-rails	Gema para testeo. https://github.com/rspec/rspec-rails
Capybara	Gema de testeo que simula un usuario. https://github.com/jnicklas/capybara
Better_errors	Gema para mostrar de una forma más amigable los errores al desarrollar. https://github.com/davjand/better_forms
Binding_of_caller	https://github.com/banister/binding_of_caller
Rails-end	
railsroad	
railroad	http://railroad.rubyforge.org/

Tabla 2 Gemas para desarrollo y Testeo

Gemas para producción	
Pg	Gema para el uso de la base de datos PostgreSQL. https://rubygems.org/gems/pg

Tabla 3 Gemas para Producción

También respecto al back end del sistema, se programó manteniendo la buena práctica de Fat-Model And Skinny Controller, manteniendo la lógica de negocios dentro del modelo, y dejando los controladores sólo como un nexo de la vistas con los modelos. Eso si se mantuvo la lógica de re direccionamiento entre páginas según los casos posibles.

Front-End

El siguiente diagrama presenta los detalles de estas dependencias.

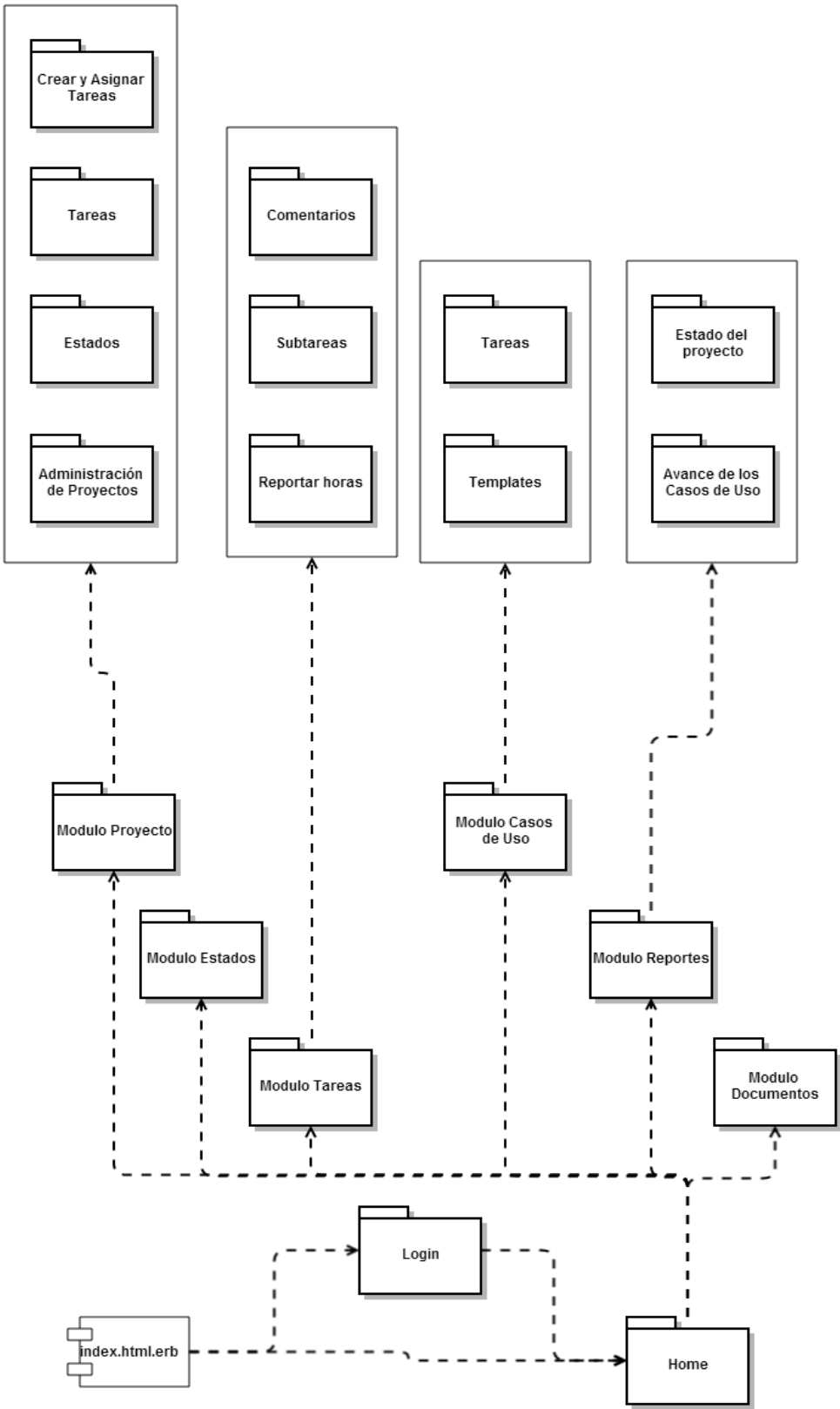
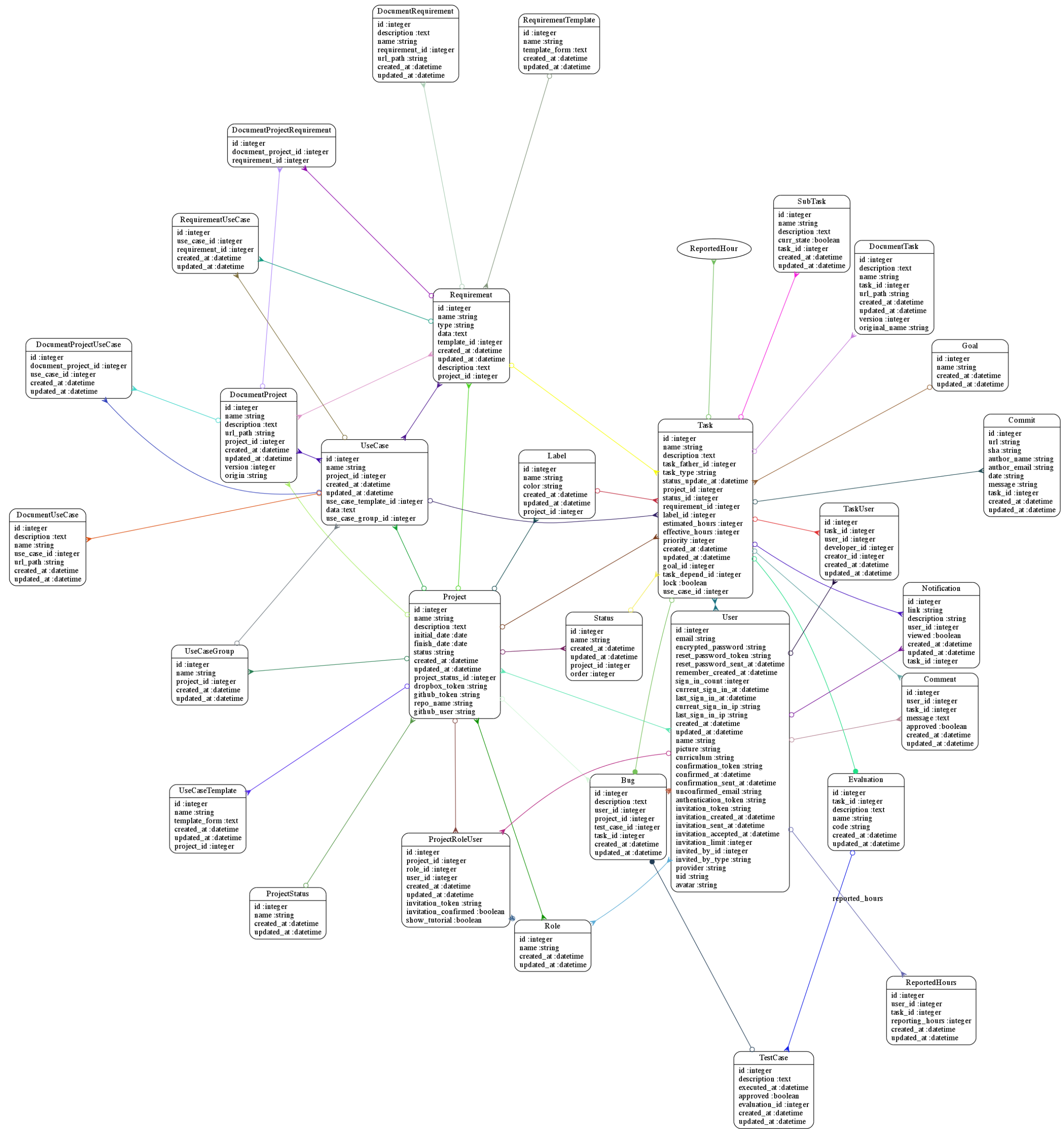


Ilustración 73 Front End Módulos

4. Vista de Datos

4.1.1 Modelo de Datos

Se utiliza una única base de datos relacional corriendo sobre el motor de base de datos PostgreSQL. La siguiente figura presenta el modelo de datos.



4.1.2 Servicios de Persistencia

Para la persistencia de datos, Ruby On Rails implementa en su librería principal Active Record, un modelo ORM (Object-Relational Mapping). La interfaz ofrece un servicio de ejecución de comandos, `execute`, para ejecución de las sentencias SQL UPDATE, INSERT y DELETE. El otro servicio, `query`, destinado a sentencias SQL basadas en SELECT. El tipo de retorno de este último es un DataSet, clase miembro de ADO.NET.

4.1.3 Servicios de Transaccionalidad

El poder transaccional utilizado es el brindado por la gema de postgresSQL que fue instalada. En una sesión de usuario corre, por vez, un módulo de la interfaz de usuario. Cada ejecución de un caso de uso completo, dirigida por uno de estos módulos, está enmarcada en una transacción. Al iniciar un caso de uso da comienzo una transacción, y al terminar el mismo, se realiza el commit. Cuando el caso de uso falla se realiza un rollback.

El enmarcado de las transacciones es realizado en la biblioteca Manager ubicado en la Interfaz de Usuario. Al comienzo de un caso de uso, el mismo solicita el comienzo de una transacción, y al final de este se realiza un commit o rollback según corresponda. La propagación del contexto transaccional se realiza junto al hilo de ejecución. No es necesario ningún mecanismo de propagación de contexto transaccional especial dado que toda la aplicación corre en forma local dentro de un único servidor.

5. Vista de Deployment

La vista de deployment presenta la infraestructura necesaria para instalar la aplicación de administración de proyectos SmartBoard.

5.1.1 Arquitectura Técnica

Se puede identificar dos nodos desde el punto de vista de procesos, el cliente y el servidor. El nodo cliente representa los dispositivos por el cual los usuarios ingresan a la página, estos siendo computadores, smartphones, tablets, entre otros. Y el nodo servidor se refiere en donde la aplicación se encuentra instalada y recibe las consultas del nodo cliente.

El nodo servidor requiere un servicio de hosting que soporte aplicaciones en Ruby on Rails y que esté configurada en forma de servicios. Actualmente se tiene hecho usando la plataforma Heroku que utiliza los servicios de Amazon.

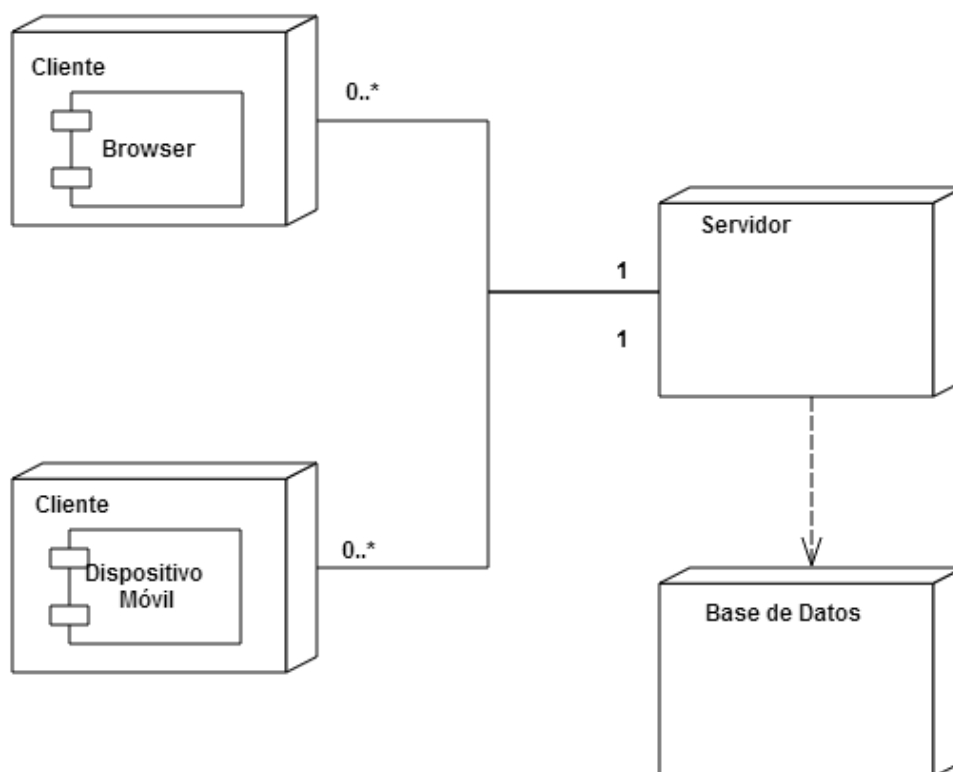


Ilustración 74 Arquitectura técnica

5.1.2 Tecnología requerida

Las estaciones de trabajo de los clientes (Cliente) deben usar preferentemente un browser con Google Chrome, Mozilla, Safari o Internet Explorer con javascript habilitado, con soporte a las últimas tecnologías web, CSS3 y HTML5. Y el servidor donde estará almacenada la aplicación tiene que soportar la plataforma de Ruby on Rails y una base de datos PostgreSQL.

5.1.3 Deployment

Para instalar esta aplicación es necesario usar un servidor con una instalación de Rails 3.2 como mínimo. La Base de datos instalada tiene que ser PostgreSQL y se tienen que instalar todas las gemas correspondientes en el GemFile. Se recomienda un servidor con 512MB de RAM, 100MB de Disco y un procesador de al menos 2.00GHz. Se puede usar por Intranet si se desea.

Para la instalación se necesita:

- Servidor web que soporte Ruby y aplicaciones Rails como:
 - Apache con la extensión de Passenger , la cual permite ejecutar la aplicación rails.
 - Lighttpd con FastCGI o SCGI.
 - Nginx
 - Algún servidor compatible con FastCGI o SCGI.
- Alternativamente se puede ejecutar en una plataforma como Heroku.