TFC – Aplicaciones Web para trabajo colaborativo

MEMORIA

Curso: 2011-2012, segundo semestre
Alumno: José Manuel Villar Fraga
Email: jvillarfr@uoc.edu – manelvf@gmail.com
Consultor: Ferran Prados Carrasco
Fecha: 09/06/2012

Índice de Contenidos

1INTRODUCCIÓN....4
I.Presentación....4
II.Objetivos....4
III.Motivación y campos de aplicación....5
IV.Entorno....5
2Estudio de Viabilidad....6
I.Recursos necesarios....6
a)Materiales....6
b)Humanos....6
II.Presupuesto inicial....7
III.Presupuesto final....7
IV.Viabilidad....8
a)Tecnológica....8
b)Económica....8
3Metodologías....9
I.Metodología básica de trabajo....9
II.Desarrollo guiado por pruebas....10
a)Tests Unitarios....10
b)Tests de Integración....10
4Planificación....11
I.Distribución en el tiempo de tareas....11
II.Hitos....12
5Marco de Trabajo y Conceptos Previos....13
I.Ficheros de traducciones....13
II.Entorno....13
III. Colaboradores....13
6Requisitos del sistema....15
I.Entorno de trabajo de desarrollo....15
a)Hardware....15
b)Software....15
II.Entorno de trabajo de producción....16
a)Hardware 16
b)Software....16
III.Entorno de trabajo de pruebas....17
7Estudios y decisiones....18
I.Consideraciones previas....18
II.Tipo de archivo a importar....18
III.Framework web....19
IV.Librerías adicionales empleadas....19
V.Base de datos....20
VI.Librerías Javascript....21
VII.Librerías de Pruebas Unitarias y de Integración....21
VIII.Imágenes y estilos....21
IX.Documentación....22
X.Sistema de control de versiones....22

XI.Licencia....22
8Análisis y diseño del sistema....23
I.Modelo de datos....23
a)Implementación 23
b)Abstracción del modelo de datos....25
II.Modelo de procesos....25
III.Modelo de ejecución de código en cliente....25
IV.Flujos principales....26
V.Diseño....28
a)Modelo Vista Controlador....28
b)Comunicación Asíncrona entre cliente y servidor....28
c)Subidas de archivos al servidor....28
d)Esquema físico de la aplicación: archivos y directorios....29
e)Esquema interno....29
f)Diseño de la interfaz de usuario....30
9Implementación y pruebas....31
I.Problemas resueltos....31
a)Lenguaje de programación Ruby....31
b)Implementación del Modelo de Datos....31
c)Autentificación....31
d)Implantación....32
e)Realización de Tests....32
II.Pruebas unitarias....33
a)Tests unitarios....33
b)Tests de integración....33
10 Implantación y resultados....34
I.Authentication....34
a)Login/ Initial Screen....34
b)Sign Up....35
II.Core application....36
a)Project List....36
b)Upload a new project....37
c)Collaborate: Translate and Vote proposals....38
d)Aceptar o rechazar colaboradores....39
III.Resultado final....40
a)Descarga del documento final....40
11 Conclusiones....41
12 Trabajo futuro....42
13 Consideraciones adicionales....43
....43
14 Glosario....44
15 Bibliografía....45
Apéndice I. Lista de figuras....46

1 INTRODUCCIÓN

I. Presentación

La presente memoria ha sido realizada por José Manuel Villar Fraga, alumno de Ingeniería Técnica de Informática de Sistemas en la Universitat Oberta de Catalunya para la asignatura de “Trabajo de Fin de Carrera – Aplicaciones web para trabajo colaborativo” en el segundo semestre del curso 2011-2012.

El proyecto consiste en una aplicación web que permita a diferentes grupos de usuarios coloborar en la internacionalización y traducción de aplicaciones de una manera coordinada y eficiente. En su desarrollo se han evaluado diferentes aproximaciones de cara a obtener las mejores soluciones para este problema.

Las aplicaciones web dinámicas ofrecen una excelente plataforma para la realización de tareas comunitarias de manera coordinada, como es el caso de las traducciones, en el que diferentes personas pueden colaborar aportando diferentes puntos de vista para de cara a mejorar la calidad global del trabajo en su conjunto.

Esta propuesta no estaba contemplada en las ofrecidas por el tutor, pero el alumno considera que cumple totalmente los requisitos de la asignatura y ofrece una funcionalidad útil, en primer lugar para el entorno del autor, y potencialmente para otros usuarios.

Durante la elaboración del documento se han intentado seguir fielmente la directrices especificadas por el tutor en sus instrucciones para la elaboración de la memoria, siendo mínimas las desviaciones con respecto al contenido propuesto.

II. Objetivos

El objetivo del proyecto es la creación de una aplicación web que permita la colaboración para la edición de ficheros de traducción a individuos y grupos de personas.

Para ello, por una parte los usuarios podrán subir los archivos de traducción originales a un servidor central y por otra, podrán editar el contenido de esos archivos de manera ágil, además de valorar el trabajo de otros usuarios. Cada uno de esos archivos será una tarea de traducción y esa tarea podrá ser abierta a todos los usuarios o sólo a los que el administrador o administradores de esa tarea decidan.

Durante el proceso de traducción se podrán conocer otros datos, como el porcentaje de completud o el número de usuarios que han colaborado.

El resultado final se obtendrá en forma de archivo descarga ble al que podrán acceder todos los usuarios o sólo los usuarios que el responsable de la traducción decida.

III. Motivación y campos de aplicación

La motivación del proyecto surge de la necesidad que existe de realizar traducciones de textos de aplicaciones. Aunque actualmente la traducción automatizada está en un estado de madurez, los usuarios no suelen estar particularmente satisfechos con ellas. Por una parte son abundantes los casos en que es necesario mejorar el resultado que ofrecen esas traducciones y por otra se requiere en numerosas ocasiones una personalización o localización (como en el caso del español de América y el español europeo) de esos contenidos.

Un aspecto adicional es que las traducciones personalizadas son particularmente importantes en el caso de las lenguas minoradas, en el que la normalización o bien no está aun consolidada o incluso es aun objeto de debate.

Otro campo de aplicación son las aplicaciones de nueva creación, para las que habitualmente es necesario realizar traducciones a diferentes idiomas de manera eficiente.

IV. Entorno

El entorno en el que se desarrolla el proyecto es la web en general y está dirigida potencialmente a cualquier tipo de público, aunque en una primera fase sus principales usuarios serían webmasters o personas relacionadas con el mundo de Internet en general y con ciertos conocimientos informáticos básicos.

Principalmente están contemplados dos tipos básicos de usuarios:

- Los **autores**, que serían los encargados de subir los archivos de traducciones y administrar los proyectos relacionados.
- Los **colaboradores**, personas con los conocimientos necesarios para poder realizar las tareas de traducción y sometidos a la supervisan de los autores.

Adicionalmente, el proyecto contempla la existencia de usuarios con el rol de **administrador**, que llevarán a cabo las tareas de administración y mantenimiento de la aplicación y ajenas al flujo de trabajo de la aplicación.

2 Estudio de Viabilidad

I. Recursos necesarios

a) Materiales

Para la parte del desarrollo de la aplicación, para la elaboración de este proyecto se empleó un equipo de desarrollo convencional:

Procesador: Intel E2220 a 3Gz

Memoria: 4 Gb DDR2

Disco duro: 640 Gb

Fabricante: HP

Para la parte del **entorno de producción** se empleó una máquina virtual ubicada en la nube de Amazon, en la que la aplicación comparte alojamiento con otras aplicaciones y prototipos. Se escogió este tipo de alojamiento por ser más conveniente para el autor, aunque se podría haber empleado otro tipo de servicio de hosting, tal y como un servidor privado virtual o físico.

También se han empleado servicios de terceros, como el servicio de control de versiones Github, además de librerías y código ajeno.

b) Humanos

En el proyecto sólo ha intervenido una persona, el autor, tanto en las labores de programación como de diseño e instalación de la aplicación. En un futuro podrían intervenir otras personas interesadas en colaborar, en función de las necesidades del proyecto.

De cara a futuras versiones, se contará con las aportaciones de terceras personas en la mejora de las funcionalidades de la aplicación.

II. Presupuesto inicial

El presupuesto elaborado inicialmente al comienzo del proyecto es el siguiente:

I. Definición del proyecto – 3h
II. Estudio de las soluciones actualmente existentes . 6h
III. Propuesta de aplicación web. 2h
IV. Selección de plataforma web 8h
V. Diseño del prototipo 8h
VI. Implementación del prototipo 16h
VII. Ajustes, documentación y test unitarios 18h

VIII. Puesta en marcha del servicio 4h
IX. Elaboración de la memoria final 6h
X. Presentación del proyecto 1h

Total. 72 h.

En la entrega del presupuesto al tutor se indicó que estaría sujeto a múltiples variaciones en función de las incidencias surgidas en el trascurso de las etapas de desarrollo.

III. Presupuesto final

Para el presupuesto final se tienen en cuenta las siguientes desviaciones:

- El tiempo necesario para realizar la puesta en marcha del servicio fue superior en 2h al estimado.
- El tiempo para la implementación del prototipo, al tratarse de un nuevo entorno para el autor, aumenta en 4h.

Debe tenerse en cuenta que en función de las funcionalidades, ajustes y retoques de diseño que se requieran, el número de horas para la etapa de implementación de los sucesivos prototipos pueden aumentar notablemente.

Para la elaboración de esta primera versión funcional se han implementado únicamente las funcionalidades básicas necesarias.

El número final de horas estimadas sería de 80.

IV. Viabilidad

a) Tecnológica

La viabilidad desde un punto tecnológico está garantizada, al emplear el sistema tecnologías muy consolidadas en el mercado, como el servidor Apache, el lenguaje de programación Ruby y la distribución de Linux Debian. La configuración de estas herramientas es una tarea muy documentada por diversas fuentes.

b) Económica

Desde un punto de vista económico, la rentabilidad del proyecto se basaría en la adición de funcionalidades adicionales de pago a ciertos usuarios, en especial a usuarios corporativos y profesionales. Uno de esos aspectos sería la limitación del número de usuarios que podrían colaborar en un proyecto o el cobro de una cantidad fija para las traducciones de carácter privado y/o una cantidad proporcional al volumen de texto a traducir.

Otro de los aspectos a tener en cuenta es que toda la información incorporada a la web está a disposición de la propia web de cara a elaborar traducciones automatizadas posteriormente. Al no haber una relación directa entre la información suministrada y sus autores, el proceso respetaría la privacidad de los usuarios.

Por último, como ingresos adicionales estaría la posibilidad de incluir anuncios de texto y/o banners de anunciantes que complementen la actividad de la aplicación. Esta opción está cada vez más en desuso, debido a que hace menos atractiva la web para los potenciales usuarios, pero sigue siendo una fuente de ingresos a tener en cuenta.

3 Metodologías

I. Metodología básica de trabajo

Una de las metodologías empleadas consiste en ciclos de trabajo que se inician en la consulta de la documentación relacionada con el objetivo del proyecto, seguida de la aplicación práctica de los datos obtenidos, según lo indicado en las tareas mencionadas en el punto 4 de este Plan de Trabajo. Una vez completado un ciclo o iteración, se vuelve a reiniciar hasta completar las tareas programadas.

Para la realización de las tareas previamente se ha consultado la documentación relacionada con las tecnologías que se van a emplear. En un primer momento, al tener que tomar una decisión acerca de las tecnologías que se van a emplear se realizó un breve estudio acerca de sus ventajas e inconvenientes, parte del cual aparecerá reflejado en los apartados referidos al análisis y diseño de esta memoria.

En las etapas de programación, la metodología empleada siguió los siguientes pasos:

1. Creación de diagramas de flujo mediante UML para determinar los estados de la aplicación.
2. Decisión de las funcionalidades que se van a implementar en una iteración
3. Implementación de los tests correspondientes a las funcionalidades.
4. Codificación de las funcionalidades.
5. Comprobación de los resultados.
6. Documentación del código.
7. Vuelta al punto dos.

II. Desarrollo guiado por pruebas

El desarrollo guiado por pruebas permite automatizar las tareas de comprobación de funcionalidades existentes, así como limitar la aparición de daños colaterales al añadir otras nuevas.

Se han empleado dos tipos de tests:

a) Tests Unitarios

Permiten testear funcionalidades puntuales (atómicas) del código, que modo que se compruebe su funcionamiento independientemente del contexto en que se empleen.

b) Tests de Integración

Permiten comprobar las funcionalidades implementadas desde el punto de vista del usuario, simulando mediante objetos “mock” los requisitos para que las vistas de la aplicación sean invocadas.

4 Planificación

I. Distribución en el tiempo de tareas

Estimación inicial de tiempos prevista:

	Name	Duration	Start	Finish
1	I. Definición del proyecto	37.125...	3/1/12...	3/13/12 5:00 PM
2	II. Estudio de las soluciones actualmente exi	16.125...	3/13/12...	3/18/12 5:00 PM
3	III. Propuesta de aplicación web	19.125...	3/19/12...	3/25/12 5:00 PM
4	IV. Selección de plataforma web	19.25...	3/26/12...	4/1/12 5:00 PM
5	V. Diseño del prototipo	25.25...	4/2/12...	4/10/12 5:00 PM
6	VI. Implementación del prototipo	76.25...	4/11/12...	5/6/12 5:00 PM
7	**VII. Ajustes, documentación y test unitari**	79.375...	**5/7/12...**	**6/2/12 6:00 PM**
8	Ajustes de la aplicación	37.875...	5/7/12...	5/19/12 10:00 PM
9	Documentación	61.25...	5/7/12...	5/27/12 5:00 PM
10	Tests Unitarios	52.375...	5/16/12...	6/2/12 6:00 PM
11	VIII. Puesta en marcha del servicio	1 day?	3/13/12...	3/13/12 4:00 PM
12	IX. Elaboración de la memoria final	19.25...	6/3/12...	6/9/12 5:00 PM
13	X. Presentación del proyecto	1.25 d...	6/10/12...	6/10/12 5:00 PM

Figura 1

Diagrama de Gantt

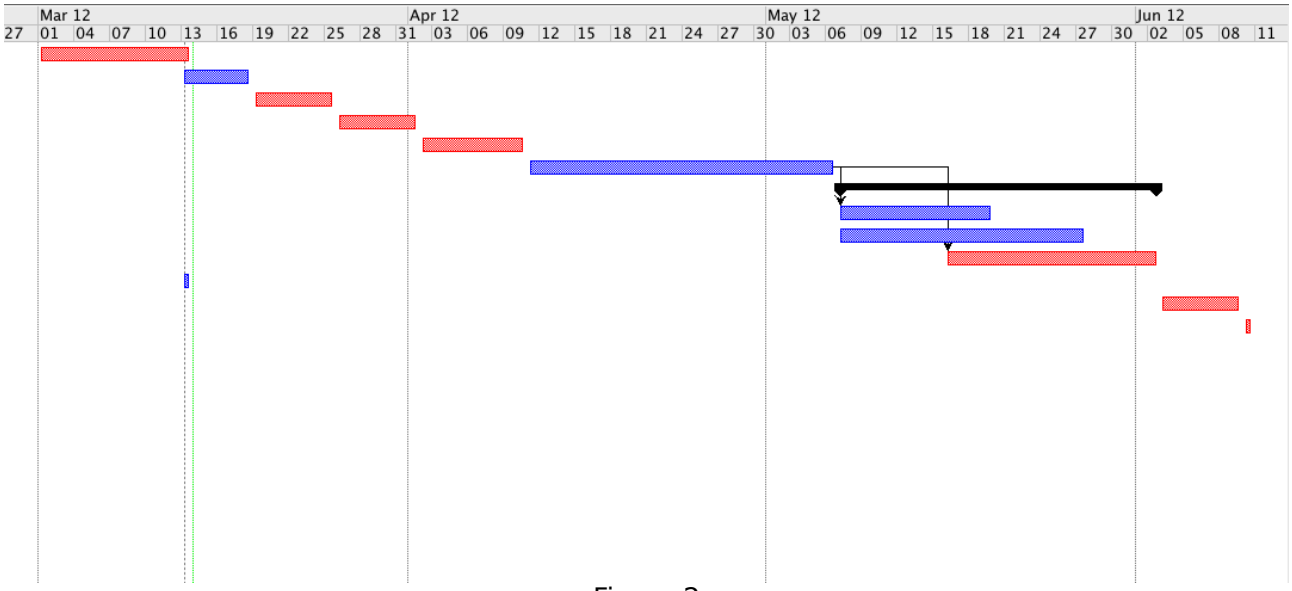


Figura 2

Nota: La entrega del plan de trabajo se retrasó dos días sobre el plan inicial debido a problemas del autor para compaginar su elaboración con su actividad profesional.

II. Hitos

Los tiempos de entrega al tutor de las PACs, o hitos del proyecto son los siguientes:

PAC1 : 13/03/2012 Plan de Trabajo

PAC2 : 25/03/2012 Especificaciones

PAC3 : 10/04/2012 Diseño

PAC4 : 10/06/2012 Entrega final y Presentación

5 Marco de Trabajo y Conceptos Previos

I. Ficheros de traducciones

Para la realización de traducciones (internacionalización) de aplicaciones se emplean diferentes soluciones. Entre las más empleadas destaca la utilización de ficheros de traducción en el que para un determinado idioma o variante idiomática se almacenan cadenas de texto que incluyen tanto los textos usados en el idioma original como los que se emplearán en la futura traducción.

Por ser de uso común en gran cantidad de aplicaciones, el formato escogido ha sido el empleado para la herramienta GNU Gettext.

II. Entorno

Para el sistema de elaboración de las traducciones colaborativas se usa el criterio de escoger la opción más votada por los colaboradores de un proyecto concreto. El usuario “autor” de la traducción tendrá la opción de vetar aquellas que no considere apropiadas.

Existe la posibilidad de desarrollar y ampliar este criterio en futuras versiones, así como de elaborar criterios nuevos, como por ejemplo, un ranking de méritos, un sistema de puntuación o una medida de entropía que ayude a mejorar la calidad de los trabajos que se realicen.

III. Colaboradores

Además del autor del proyecto, el tutor ha realizado una importante aportación a este trabajo. En primer lugar, ha aceptado la realización de este proyecto que, aunque cumple los objetivos propuestos por la asignatura, no estaba entre los propuestos inicialmente. Además, su ayuda ha sido importante particularmente importante en las primeras fases del proyecto, en las que se han perfilado sus líneas generales más importantes.

Para la realización de este proyecto se contó con la opinión a nivel informativo de los miembros del proyecto Trasno, grupo de voluntarios para la creación y mantenimiento de traducciones de software libre a lengua gallega. Además de esta colaboración, diferentes usuarios de comunidades online han dado su opinión sobre la necesidad de contar con una herramienta de este estilo a la hora de realizar traducciones para todo tipo de aplicaciones.

6 Requisitos del sistema

I. Entorno de trabajo de desarrollo

a) Hardware

Equipos utilizados:

- Apple iMac early 2009 3Ghz. 4 Gb de RAM. 640 Gb disco duro. Sistema operativo: OSX Lion
- HP Intel E2220 2Gz. 4 Gb de RAM. 500 Gb disco duro. Sistema operativo: Ubuntu Linux 11.10

También se han empleado máquinas virtuales emuladas con el software VMWare Fusion con el fin de testear puntualmente los navegadores de la familia Internet Explorer, que sólo pueden ejecutarse fielmente en el sistema operativo Microsoft Windows.

b) Software

Las aplicaciones que se han utilizado han sido, entre otras:

- La suite ofimática LibreOffice para crear los documentos necesarios (PACs, memoria final y presentación).
- Para la realización de diagramas mostrados en las PACS se ha empleado el editor ArgoUML.
- Para las tareas de codificación se ha empleado el editor de textos Vim. Se han añadido las funcionalidades de diferentes plugins: pathogen, ctags, NerdExplorer..
- Para la ejecución de la aplicación web en el entorno de desarrollo se ha empleado el servidor Webrick, incluido en el framework Sinatra.
- El intérprete de Ruby utilizado ha sido la versión 1.9.3 de la edición MRI.
- Para el manejo de las versiones de los paquetes Ruby se han empleado el gestor de paquetes 'gem' y el gestor de versiones 'rvm'.
- La base de datos utilizada tanto en desarrollo como en producción ha sido MongoDB. Las razones para su elección se detallan el apartado de análisis de la aplicación.

II. Entorno de trabajo de producción

a) Hardware

Para el entorno de producción se ha escogido una distribución del sistema operativo Linux Debian “Lenny” ejecutándose sobre una instancia de la arquitectura Amazon Cloud con las siguientes características:

Amazon Small Instance:

1.7 GB memory
1 EC2 Compute Unit (1 virtual core with 1 EC2 Compute Unit)
160 GB instance storage
32-bit or 64-bit platform
I/O Performance: Moderate
API name: m1.small

Los motivos de haber escogido este entorno son la flexibilidad al no tener que depender de un equipo físico para administrar y el hecho de ser una plataforma fácilmente escalable, que hipotéticamente admitiría una ampliación de los recursos con los que cuenta actualmente la instancia, de forma transparente. Se trata de una solución pensada para solucionar los problemas de escalabilidad.

b) Software

En el entorno de producción se han utilizado la siguientes aplicaciones (production stack):

- Sistema operativo: Debian Lenny. La razón de usar este sistema es la gran cantidad de software y documentación disponibles y el hecho de que sea uno de los sistemas operativos más empleados en todo el mundo para instalar servidores web.
- Servidor web: Apache. Las razones para su empleo son similares a las del punto anterior. Aunque actualmente existen servidores más potentes, como Nginx o Cherokee Server, el hecho de disponer previamente de gran cantidad de documentación ha hecho que el autor de este proyecto se decante por el.
- Intéprete Ruby: MRI. Aunque existen otros intérpretes de Ruby que ofrecen mejor rendimiento e incluso una implementación en la máquina virtual de Java (JVM), la gran cantidad de paquetes disponibles hacen que sea la opción más recomendable para este tipo de proyectos.

III. Entorno de trabajo de pruebas

De cara a disponer de un entorno para la realización de pruebas y comprobación de funcionalidades, el autor valora la posibilidad de emplear una imagen de la máquina virtual de producción que refleje fielmente ese entorno antes de realizar la implantación en la máquina de producción.

En ese entorno se realizarían principalmente las pruebas de tests unitarios y de integración antes de la subida final a producción.

7 Estudios y decisiones

I. Consideraciones previas

Para la realización de este proyecto se requería un entorno ligero, que permitiese un manejo ágil de la información y un prototipado rápido, aunque tuviese una eficiencia máxima.

Por ese motivo se desecharon opciones que ofrecían un mejor rendimiento que las enumeradas en este apartado, en favor de las que ofrecían mejores ventajas al programador en la etapa de desarrollo.

Otra cuestión importante a la hora de decidir el software a emplear ha sido el que esté respaldado por una licencia de código abierto, de cara a contar con una serie de garantías a la hora de ser usado.

II. Tipo de archivo a importar

Después de realizar un estudio de los diferentes tipos de archivo empleados para realizar las tareas de internacionalización, el autor tomó la decisión de utilizar el formato de archivos **".po"** perteneciente al conjunto de utilidades Gettext, debido a su amplia difusión y a la simplicidad de su manejo.

El formato ".po" está formado por cadenas de texto consecutivas en las que figura por una parte la cadena en la lengua original y a continuación la cadena de traducción en la lengua destino. Admite también el uso de comentarios que no se contemplan en el producto final.

De cara a futuras mejoras, se realizaría la ampliación de la aplicación para que soportase diferentes tipos de archivos, ampliando el modelos de datos si fuese necesario para proveer la funcionalidad necesaria.

III. Framework web

Después de valorar diferentes soluciones, se entendió que la más óptima era la de emplear un framework web de los ya existentes, por incluir ya implementadas gran cantidad de tareas necesarias para elaborar una aplicación web.

Para la tarea de escoger el framework a utilizar, el autor se decidió por emplear Sinatra, en el lenguaje de programación Ruby, debido a considerarlo perfectamente apropiado para la tarea. Entre los motivos:

- Incorpora una funcionalidad mínima, que permite utilizar los aspectos básicos de una aplicación web, como los tipos de request (PUT, GET, POST) o las rutas.
- No está atado a un sistema de plantillas en particular, por lo que se podrían emplear incluso diferentes sistemas según las necesidades de la aplicación.
- No utiliza un sistema de acceso a base de datos u ORM por defecto, por lo que los programadores que lo usan tienen libertad para escoger el tipo de acceso que necesiten.
- Al ser un framework ligero, permite servir un gran volumen de datos con un consumo mínimo de recursos.
- Tanto el framework Sinatra como el lenguaje de programación Ruby permite desarrollar aplicaciones de forma rápida y fácilmente extensible.

Después de utilizarlo para codificar, éstas son algunas de mis conclusiones:

- El utilizar un framework ligero permite un mayor control de las posibilidades de la aplicación.
- El tiempo para tener un prototipo básico funcionando es menor que cuando se utiliza un framework completo.
- Para un equipo de desarrollo mayor (>3 desarrolladores), puede ser interesante emplear un framework con mayores opciones.

IV. Librerías adicionales empleadas

Se han empleado las siguientes librerías adicionales en la parte del servidor:

- 'gettext', '2.2.0' . Soporte para internacionalización
- 'fast_gettext', '0.6.7' . Soporte para internacionalización en C puro.
- 'haml', '3.1.5' . Módulo de plantillas HAML para codificación rápida de HTML
- 'sinatra-authentication', '0.4.1' . Módulo de autentificación de usuarios.
- 'sinatra-reloader', '1.0' . Usado en desarrollo. Permite recargar automáticamente el servidor de desarrollo al modificar un archivo del

código fuente.

- 'sinatra-flash', '0.3.0' . Soporte para mensajes 'flash', enviados entre peticiones al servidor.
- 'mongo', '1.6.2' . Acceso a la base de datos MongoDB.
- gem 'mongo_mapper', '0.11.1' . Capa de abstracción sobre el acceso a la base de datos.
- 'bson', '1.6.2' . Soporte binario requerido por la librería de acceso a mongodb.

V. Base de datos

Cuando se estudió el caso del modelo de base de datos a emplear, se valoraron dos opciones principales.

La primera opción fue la de emplear una base de datos relacional, tal y como ha venido empleándose en el ámbito de las aplicaciones web en los últimos quince años, especialmente desde la consolidación de las herramientas LAMP (Linux + Apache + MySQL + PHP). Debido a que el principal tipo de datos a tratar era un documento (el fichero de traducción), se valoró una segunda posibilidad, que era la posibilidad una base de datos orientada a documentos.

Las bases de datos orientadas a documentos forman parte de un grupo mayor de bases de datos llamado NoSQL y permiten tratar los datos como si se tratase de un documento en vez del tratamiento que se les da en las relacionales, basado en tablas y campos incluidos en éstas.

Una vez corroborado que el sistema se adaptaba potencialmente al objetivo propuesto, se enumeraron las diferentes bases de datos orientadas a documento existentes que cumpliesen los requisitos principales. La elección se restringió a dos posibilidades: CouchDB y MongoDB.

En último término, la mayor versatilidad y simplicidad de MongoDB determinó la elección. En cualquier caso se contempla la posibilidad de emplear otra base de datos en el futuro con el fin de obtener algún tipo de mejora sobre el sistema actual, como podría ser un sistema de control de versiones de los documentos.

VI. Librerías JavaScript

Las librerías de JavaScript para el uso en el navegador de los usuarios (cliente) son las siguientes:

- jQuery: se utiliza para realizar manipulaciones del DOM (Document Object Model), realización de peticiones Ajax y otras tareas menores, como la extensión de objetos js.
- Mustache: se utiliza para la elaboración y uso de plantillas javascript.

VII. Librerías de Pruebas Unitarias y de Integración

Para la realización tanto de pruebas unitarias como de integración se usa en el lado del servidor las librerías:

- Test/Unit . Se encarga de proporcionar las funciones propias de una suite de Unit Testing básica, por lo que es independiente del framework.
- Rack/Test . Es la librería que da soporte para poder realizar pruebas sobre los componentes del framework. Es interdependiente con el framework web empleado.

Existe la posibilidad de realizar Pruebas orientadas al comportamiento (BDD – Behavior Driven Development), pero en esta etapa de desarrollo del proyecto no se ha entendido que fuese oportuno realizarlos.

VIII. Imágenes y estilos

Para el estilo visual de la aplicación, se ha usado el conjunto de hojas de estilo y librerías javascript “Twitter Bootstrap”, de cara a dotar de una funcionalidad mínima y estilo homogéneo que permitan una mejor experiencia de usuario. Estos estilos se han complementado con modificaciones puntuales.

En cuanto a las imágenes empleadas, se ha usado un conjunto de iconos de uso público, SweetiePlus y disponibles gratuitamente.

IX. Documentación

Para la generación de la documentación de la aplicación se ha empleado la herramienta RDoc, perteneciente al conjunto de herramientas incluidas en el paquete de instalación del lenguaje Ruby en Debian.

Esta herramienta genera unos archivos html que pueden ser posteriormente leídos normalmente por un navegador web, con la información sobre clases, métodos y módulos incluidos en el proyecto.

X. Sistema de control de versiones

El sistema de control de versiones empleado ha sido 'Git'. Se trata de un sistema de control de versiones distribuido que permite que diferentes usuarios trabajen con diferentes copias del mismo código sin necesidad de emplear un repositorio central.

El servicio escogido ha sido GitHub, que permite repositorios públicos para los proyectos de código abierto. Esta elección permite la incorporación de aportaciones a partir de la realización de 'pull requests' por parte de futuros colaboradores.

XI. Licencia

La licencia escogida para el código del proyecto ha sido la del MIT (Massachussets Institute of Technology), por ser una licencia que ampara el uso del código por parte de terceros y permite su distribución al ser de código abierto.

Las aplicaciones con las que se ha elaborado el proyecto también cuentan con licencias de código abierto, sean la misma de la aplicación (MIT) o variantes de la GNU.

8 Análisis y diseño del sistema

I. Modelo de datos

a) Implementación

Para el modelo de datos se ha escogido utilizar una variación del modelo Entidad – Relación, adaptado al esquema de una base de datos orientada a documentos. La principal variación consiste en que las diferentes entidades se incluyen asociativamente en una entidad principal, al no tener sentido su existencia fuera de ésta.

En este caso, las dos entidades principales son la entidad Documento y la entidad Usuario. La relación entre las dos se realiza mediante la inclusión del identificador de usuario como un campo de la entidad documento.

Dentro de la entidad Documento se incluye el modelo colaboración y el modelo línea, para cada línea de traducción, dentro de cada modelo la propuesta y a su vez dentro de estas los votos de cada usuario.

Modelo de datos:

```
class Document
  include MongoMapper::Document

  key :name, String, :required => true
  key :language, String, :required => true
  key :variation, String
  key :description, String
  key :authorId, String, :required => true
  key :status, String, :required => true
  key :is_private, Boolean
  key :collaboratorIds, Array

  many :lines

  timestamps!
end


class Line
  include MongoMapper::EmbeddedDocument

  key :msgid, String
  key :msgstr, String

  many :proposals
  embedded_in :Document
end

class Proposal
  include MongoMapper::EmbeddedDocument

  key :msgstr, String
  key :authorId, String

  many :votes
  embedded_in :Line

  timestamps!
end

class Vote
  include MongoMapper::EmbeddedDocument

  key :authorId, String

  embedded_in :Proposal

  timestamps!
end
```

```
class Collaboration
  include MongoMapper::EmbeddedDocument

  key :authorId, String
  key :status, String

  embedded_in :Document

  timestamps!
end
```

Figuras 3 y 4

b) Abstracción del modelo de datos

De cara a facilitar la manipulación del modelo de datos, se ha empleado la librería MongoMapper, que añade soporte a operaciones más complejas de asociación, consulta y empotrado de documentos.

II. Modelo de procesos

El modelo de procesos empleado se basa en peticiones (requests), que son realizadas por los clientes que se conectan al servidor atendidas por éste. Cada petición crea un nuevo hilo que finaliza al acabar la ejecución de las operaciones las operaciones que le corresponden.

III. Modelo de ejecución de código en cliente

El modelo de ejecución de código en el cliente corresponde a una arquitectura orientada a eventos, en la que se asocian funciones a eventos que suceden en el sistema, sean producto del propio sistema (carga del DOM) o realizados por el usuario (pulsar un botón, enviar un formulario).

IV. Flujos principales

Los flujos principales de la aplicación son los siguientes:

- Flujo principal de actividad:

Figura 5

- Flujo de opciones:

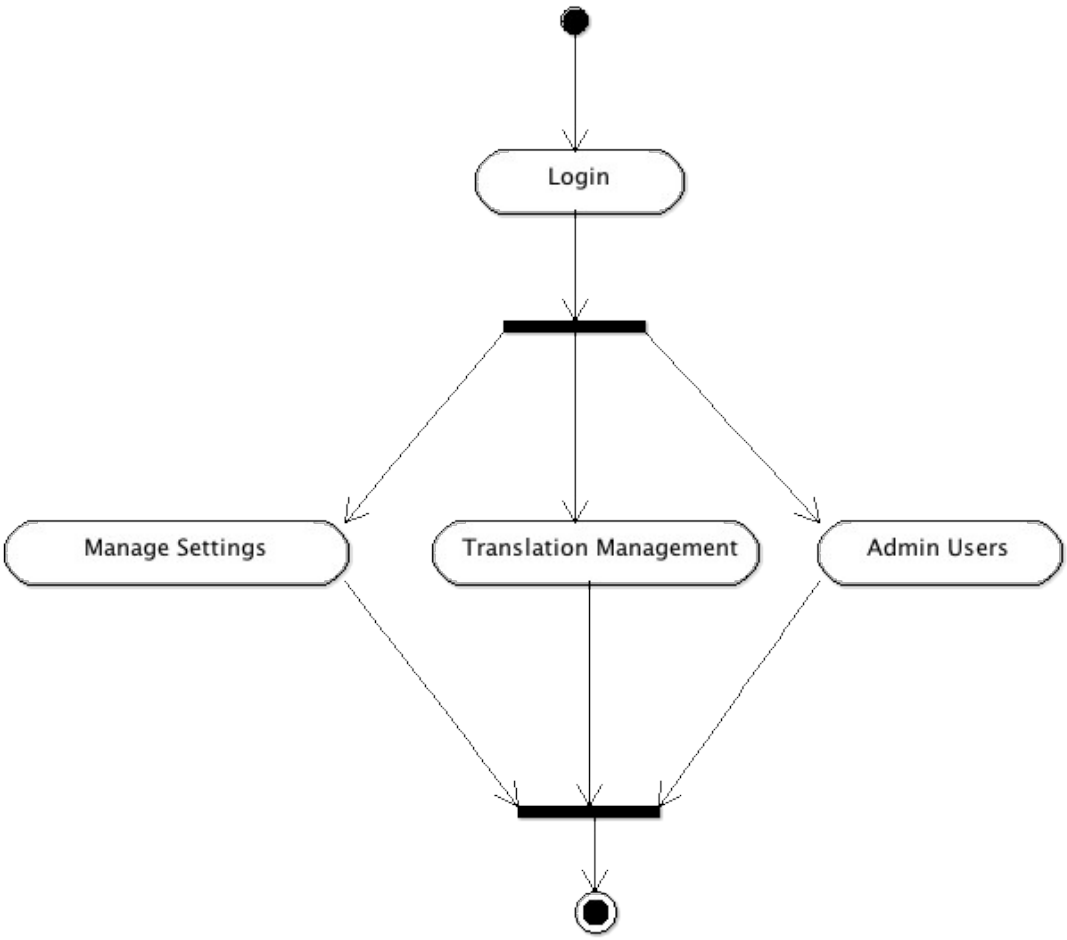


Figura 6

V. Diseño

a) Modelo Vista Controlador

En cuanto al diseño del esquema de la aplicación, se ha empleado el patrón Modelo Vista Controlador, con el objetivo de separar lo datos su representación y de las acciones que se realicen. En ocasiones diversos autores han argumentado que el modelo debería ser denominado (Modelo-Vista-Plantilla), y el controlador ser considerada la aplicación en su conjunto. Ese punto de vista también se aplica al modelo de aplicación utilizado.

El modelo de datos está incluido en el archivo **model.rb**, las peticiones (controlador) se despachan a través de **transboard.rb** . Existen una serie de archivos en forma de librerías que dan soporte a este a este último (upload.rb, etc...). Por último, en la carpeta views figuran las plantillas que se utilizarán para mostrar visualmente la aplicación.

b) Comunicación Asíncrona entre cliente y servidor

De cara a facilitar la comunicación entre cliente y servidor sin tener que refrescar íntegramente la página en el cliente, el sistema emplea peticiones Ajax, implementadas mediante peticiones GET ó POST.

Las peticiones están asociadas a un evento, que lanza la consulta Ajax y la respuesta se recibe asíncronamente, ejecutando el cliente la acción correspondiente al resultado de la llamada.

c) Subidas de archivos al servidor

Los archivos subidos por los usuarios al servidor se almacenan temporalmente en una carpeta temporal, '/tmp'. Su contenido es analizado por las funciones destinadas a esa tarea y almacenado consecuentemente en la base de datos.

d) Esquema físico de la aplicación: archivos y directorios

- Raíz
  - transboard.rb – Cuerpo principal de la aplicación. Define cómo se responderá a las peticiones servidas por el servidor Apache.
  - model.rb – Archivo donde se define el modelo de datos y ciertas operaciones relacionadas con ellos.
  - views <Directorio>
    - Plantillas HAML
  - static <Directorio>
    - Todo tipo de archivos estáticos (imágenes, archivos javascript, hojas de estilo)
  - doc <Directorio> - Documentación generada por RDOC
  - test <Directorio> : Carpeta que contiene tanto los tests unitarios como de integración.
  - fixtures <Directorio>: Pre-carga de datos tanto para la aplicación en producción como para los tests.

e) Esquema interno

En cuanto al funcionamiento el interno, el framework Sinatra proporciona un lenguaje específico de dominio (DSL), que permite definir cómo se atienden las peticiones que se reciben a través del servidor. También proporciona métodos para trabajar con sesiones y despachar peticiones diferentes del html habitual.

El manejo de rutas está también incluido en el framework Sinatra, así como la evaluación de parámetros. Es por medio de esas rutas que se define el comportamiento de la aplicación.

Las librerías incluidas en forma de gemas de ruby dan soporte a temas como la autentificación, acceso a bases de datos o plantillas. De cara a la posterior instalación se emplea la gema “bundler”, que permite llevar un

seguimiento de las versiones de las librerías que se van a usar.

f) Diseño de la interfaz de usuario

Para la elaboración de la interfaz gráfica se toman en cuenta las consideraciones del W3C en cuanto a estándares y se toma como referencia el navegador Mozilla Firefox en su versión 12 por ser objetivamente de entre los disponibles uno de los más fieles a esos estándares.

9 Implementación y pruebas

I. Problemas resueltos

a) Lenguaje de programación Ruby

Al carecer el autor de experiencia previa en este lenguaje de programación, debió de adquirirlos en el transcurso de las etapas de desarrollo. Al tratarse de un lenguaje muy documentado, esa limitación fue desapareciendo paulatinamente.

b) Implementación del Modelo de Datos

De cara a la implementación del modelo de datos, para el autor fue complicado usar el modelos de datos orientado a documentos debido a estar acostumbrado a usar un esquema relacional.

También fue un problema la decisión de si era necesario emplear una capa de abstracción por encima del acceso básico a la base de datos y en caso afirmativo, cual de las disponibles actualmente en la red era más óptima.

Después de comprobar que el acceso a través de la librería básica no era lo realmente completo que se necesitaba, se probó la librería MongoId. Al encontrar dificultades con las consultas complejas, el autor decidió emplear la librería MongoMapper, que resultó ser más adecuada, aunque incompleta en ciertos aspectos.

El resultado final fue un empleo de la librería MongoMapper con accesos puntuales la base de datos empleando la librería de acceso básico. El autor estudia la posibilidad de emplear un patrón adaptador para realizar un cambio de librería de abstracción de acceso a datos en un futuro.

c) Autentificación

Para la autentificación de usuarios se empleó una gema ya configurada que realiza un acceso por defecto a través de Email. A los campos por defecto se añadió el campo “UserName” de cara a preservar la privacidad de los correos de los usuarios.

d) Implantación

En la etapa de la implementación en el servidor de producción se utilizó la herramienta Git para realizar un volcado del código fuente a un directorio del servidor.

De cara a la configuración del servidor, se empleó un archivo config.ru no presente en el código fuente. Para la configuración de apache se empleó el siguiente código:

```
<VirtualHost *:80>
        ServerName transboard.manelvf.com

        DocumentRoot /var/www/Transboard/static/
    <Directory /var/www/Transboard/static/>
      # This relaxes Apache security settings.
      AllowOverride all
      # MultiViews must be turned off.
      Options -MultiViews
    </Directory>
</VirtualHost>
```

e) Realización de Tests

En el entorno de ejecución de tests, hubo problemas para el uso de funciones de prueba de peticiones web, además de la parte de configuración (setUp) de los propios tests. El uso de fixtures y una ampliación del conocimiento relativo a objetos Mock y su uso en tests de integración resolvieron en parte esos problemas.

II. Pruebas unitarias

a) Tests unitarios

Los tests unitarios creados se realizaron para comprobar funciones concretas del flujo de la aplicación, como es el caso de comprobación de que las cadenas de texto incluidas en los archivos subidos al servidor cumplan determinadas reglas.

b) Tests de integración

Los tests de integración que se realizaron permiten comprobar el correcto funcionamiento de las principales peticiones realizadas, como es el caso del login/logout y de la subida de archivos.

Estas pruebas permiten dar la seguridad de que la el contenido actual de la web responderá ante futuros cambios en el código.

10 Implantación y resultados

I. Authentication

a) Login/ Initial Screen

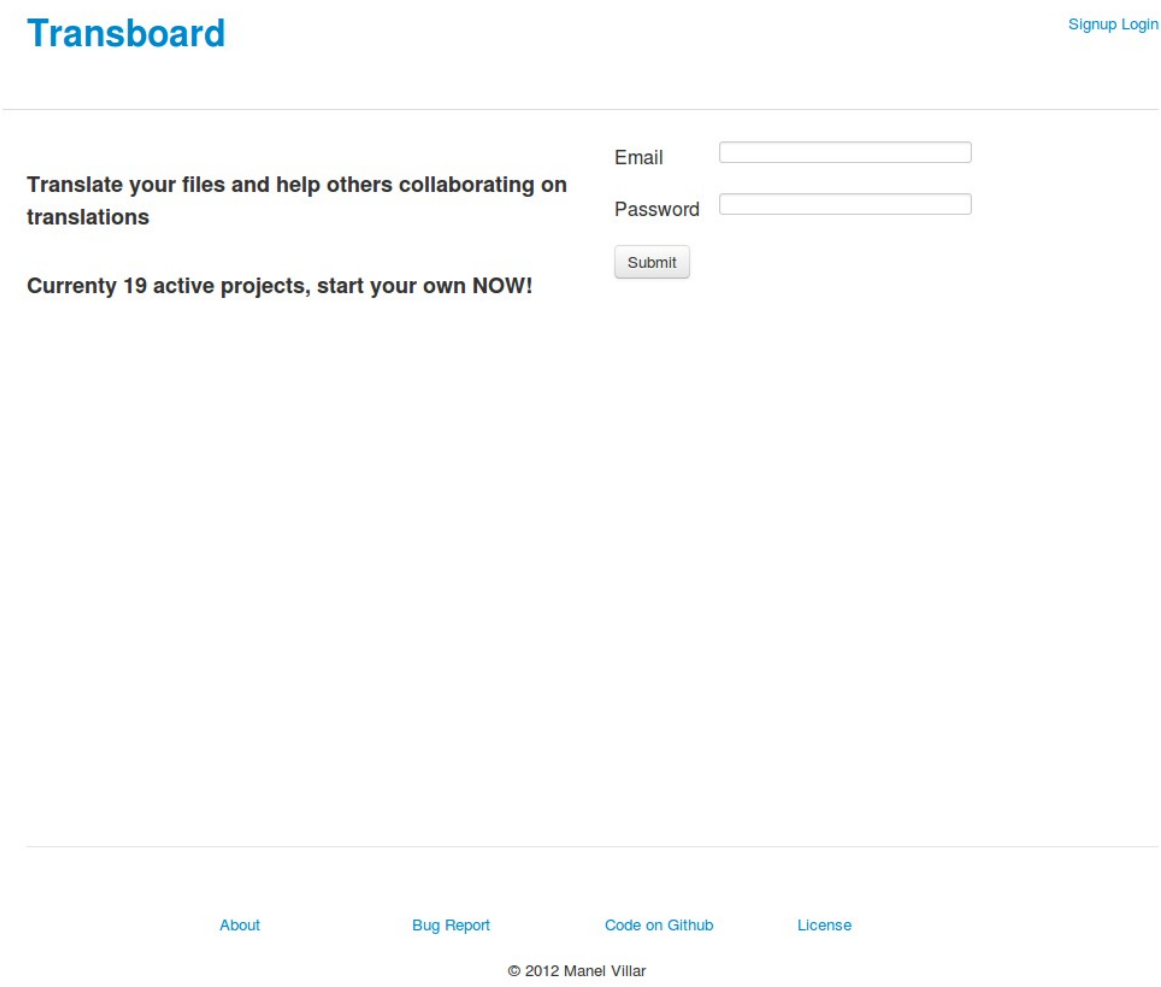


Figura 7

b) Sign Up

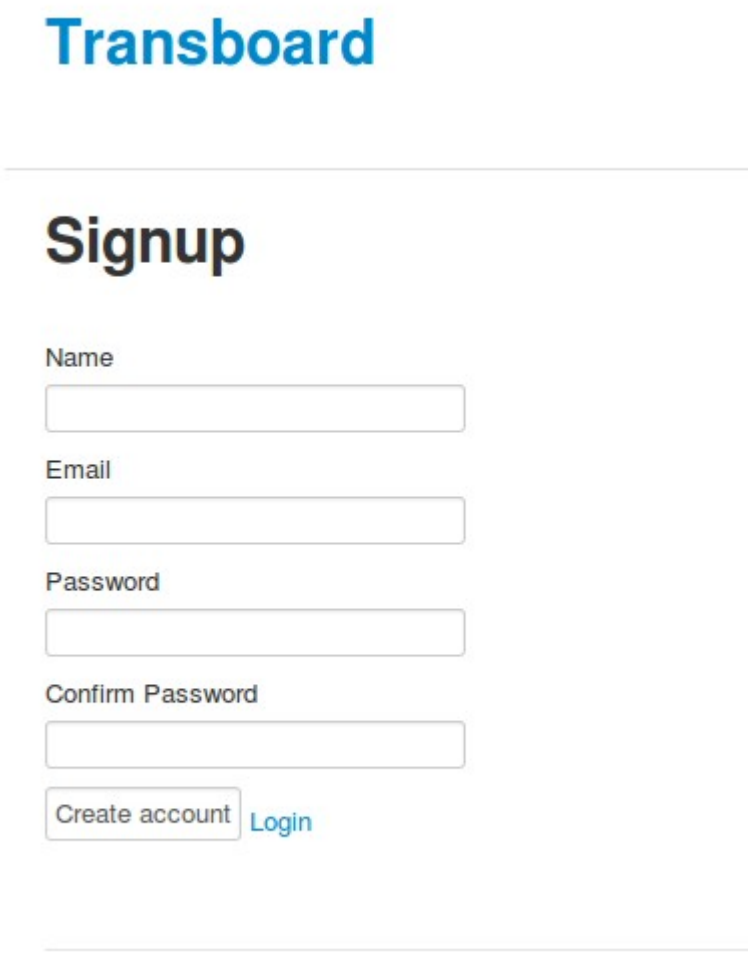


Figura 8

II. Core application

a) Project List

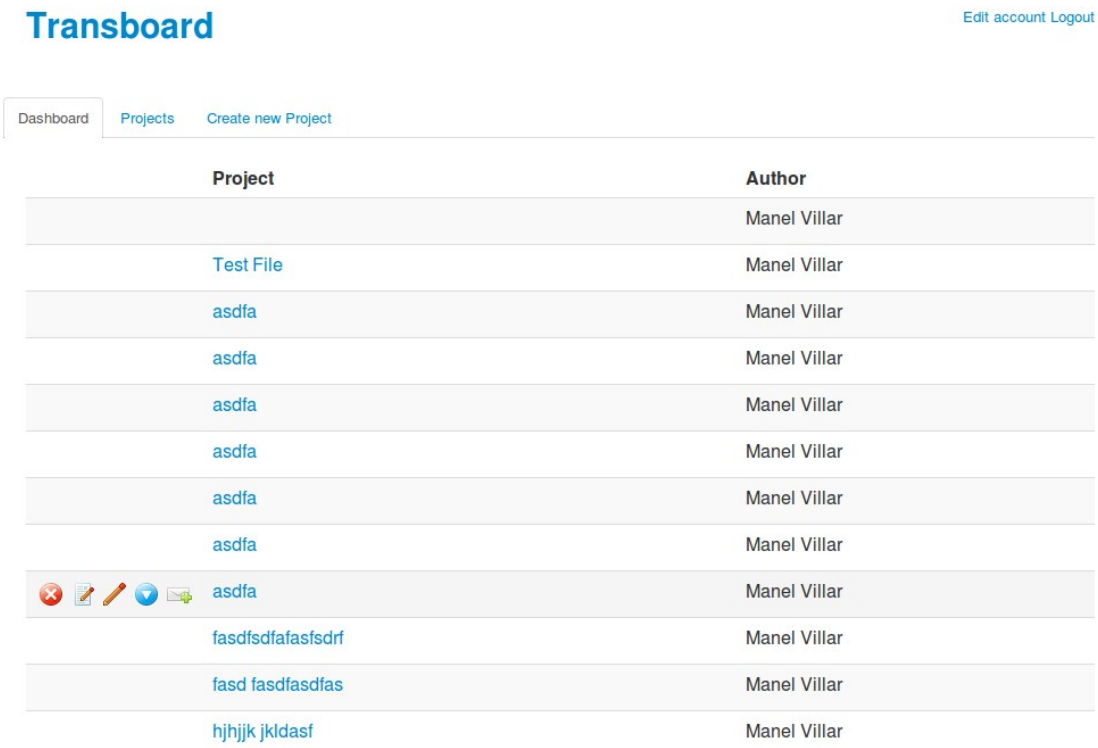


Figura 9

Los iconos aparecen sólo cuando el usuario pasa el cursor por encima de una línea de detalle.

Las acciones incluidas por cada icono son:

- Borrar documento: Sólo para el autor del archivo subido
- Editar opciones: Sólo para el autor del archivo subido. Permite editar las opciones del proyecto.
- Editar archivo: Permite editar el archivo para proponer y votar propuestas de traducción.
- Descargar archivo: Permite descargar el archivo, obteniendo la versión con las propuestas de traducción más votadas.
- Pedir permiso para colaborar: Pide permiso a un autor para coloborar en una traducción.

b) Upload a new project

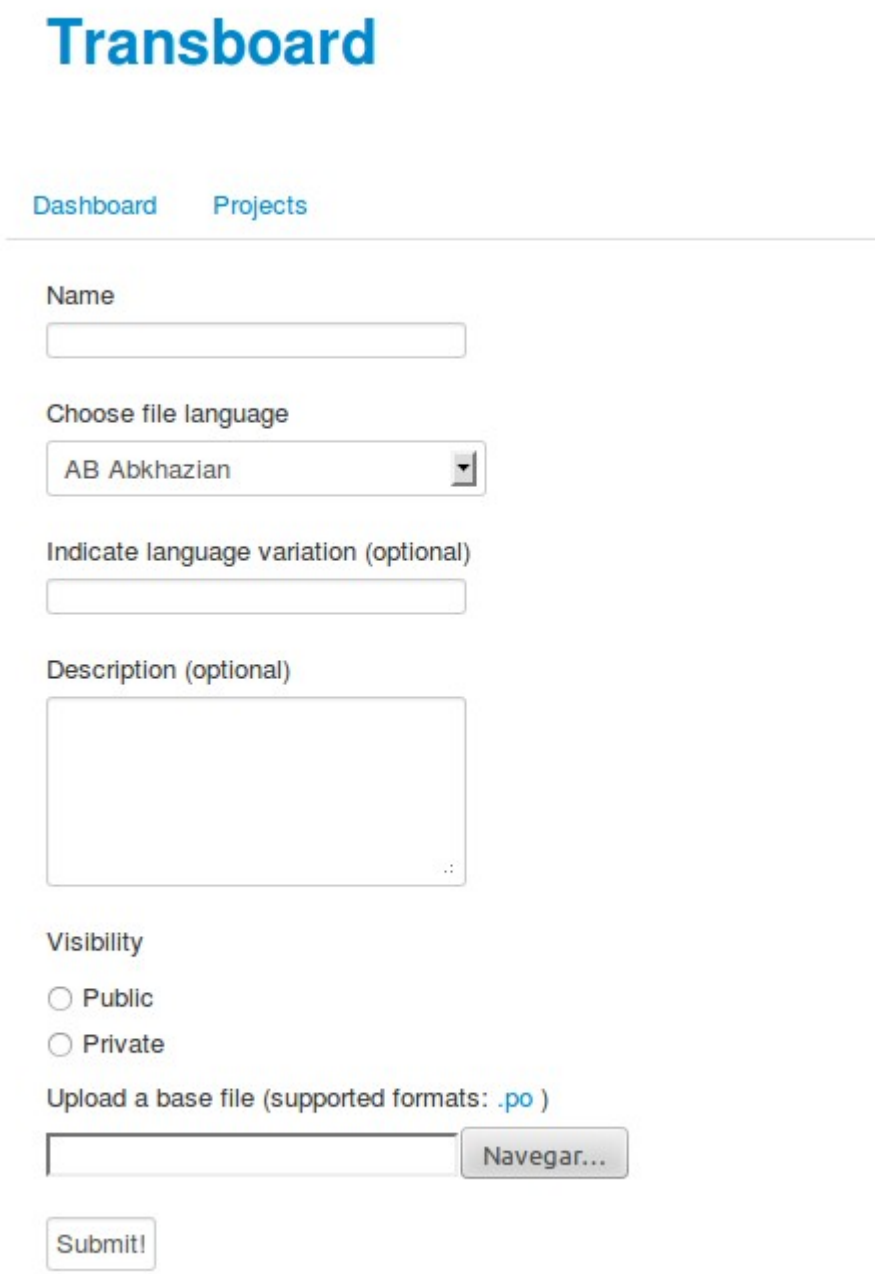
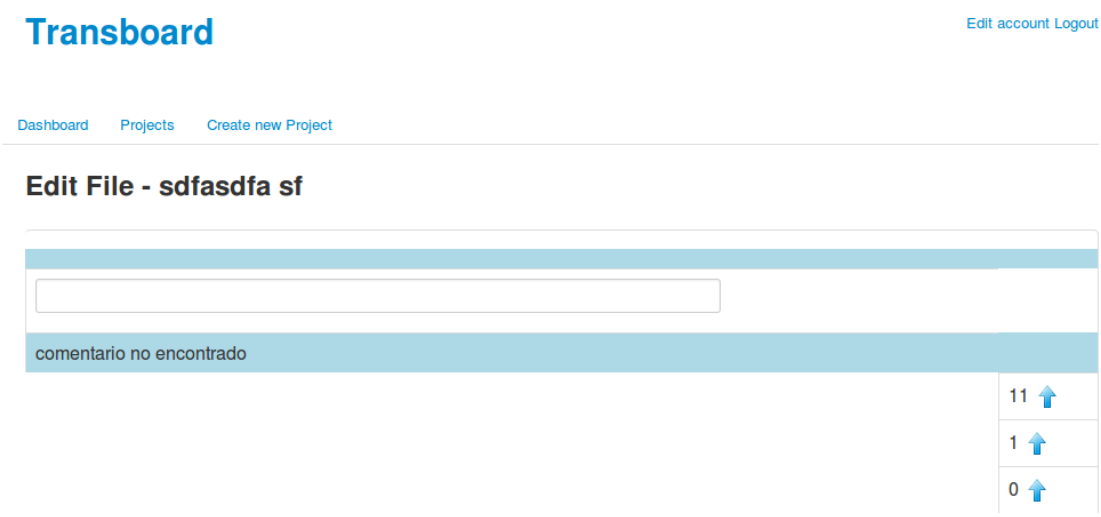
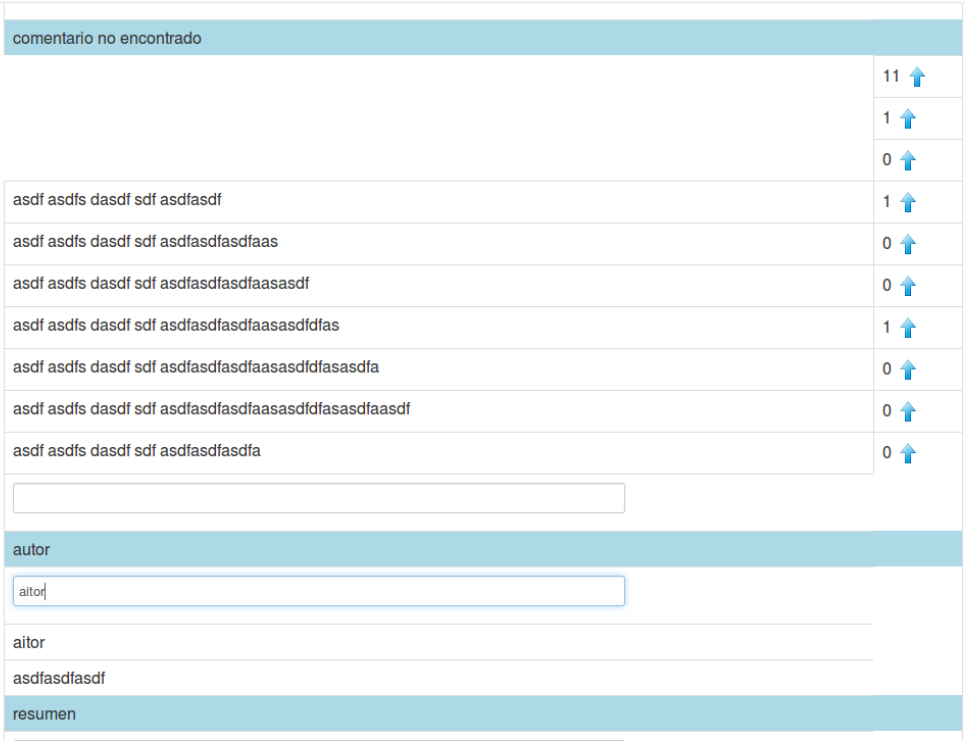


Figura 10

c) *Collaborate: Translate and Vote proposals*





Figuras 11 y 12

d) Aceptar o rechazar colaboradores

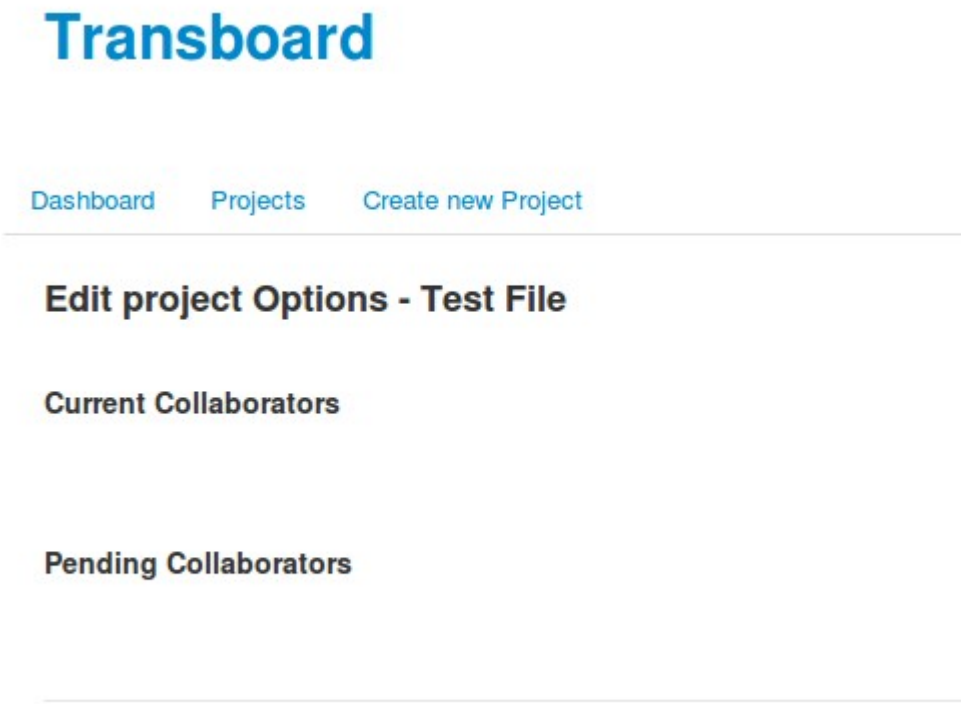


Figura 13

En caso de que haya peticiones de usuarios pendientes de aceptar, aparecerán en esta vista, junto con los colaboradores que ya se hayan aceptado.

III. Resultado final

a) Descarga del documento final

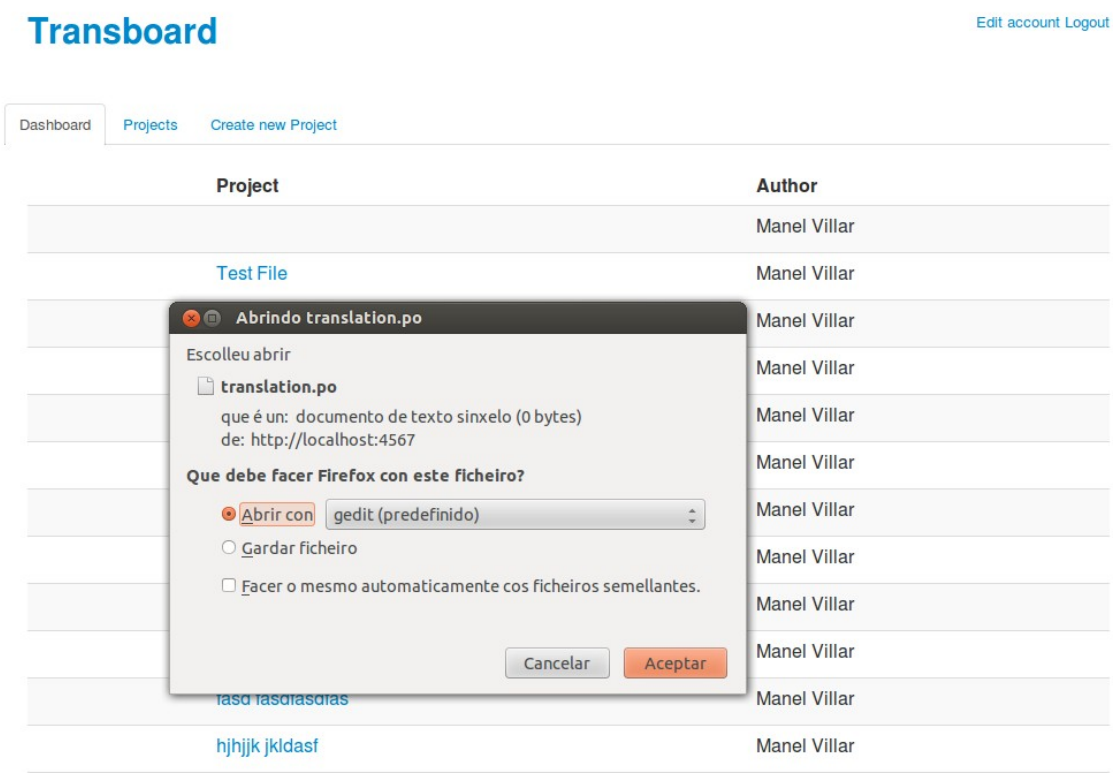


Figura 14

11 Conclusiones

Como conclusión principal el autor destaca que es técnicamente viable el desarrollo de una aplicación web que permita el trabajo colaborativo en la realización de traducciones mediante archivos destinados a ese uso.

El uso de un modelos de base de datos orientado a documentos se encuentra óptimo para este tipo de proyecto, ya que permite una integración total de los datos que pertenecen aun mismo documento. El principal problema para el uso de este tipo de base de datos fue para el autor el acostumbrarse al cambio de paradigma, al basar sus experiencias anteriores en aplicaciones web en modelos de bases de datos relacionales (MySQL, PostgreSQL). Una vez adaptado al nuevo modelo, se muestra muy eficiente a la hora de reflejar las relaciones entre las diferentes entidades.

El uso de las herramientas de Ruby, como rdoc para documentación o Gemfile, rvm y bundle para manejar las diferentes versiones, permite llevar un control de las dependencias del proyecto, permitiendo que pueda ser portado ágilmente a otras instalaciones. El uso de una herramienta de control de versiones como 'Git' se muestra práctico para llevar un control del código, incluso en un entorno en el que sólo hay aportaciones de un desarrollador.

En cuanto a la apariencia gráfica, el uso de un sistema de hojas de estilos que proporcionan un estilo visual predefinido permite dar un acabado de una calidad aceptable que permite al desarrollador centrarse en los aspectos prácticos y funcionales de la aplicación.

De cara a las tareas colaborativas, el empleo de tecnología Ajax permite hacer que la experiencia sea mucho más fluida que en una aplicación tradicional en la que cada operación implique un refresco completo de la página web cargada en el navegador.

Como conclusión final, el autor comenta que el uso que tenga la aplicación dependerá de la acogida que los usuarios realicen de la herramienta y de la utilidad que encuentren en ella. Es voluntad del autor seguir realizando mejoras en el código hasta obtener la mejor experiencia de usuario posible.

12 Trabajo futuro

Tareas pendientes de realizar que complementan la funcionalidad existente:

- Añadir soporte de internacionalización a la propia aplicación
- Añadir notificaciones por Email
- Modificar el estilo visual para lograr un producto diferenciado
- Incrementar el número de formatos a traducir, empleando otros de los existentes en el mercado
- Reforzar la seguridad, implementando sistemas adecuados al tipo de base de datos empleado
- Crear un sistema de ranking para los usuarios, basado en la entropía con la propia comunidad (similitud de sus votos con los resultados finales)
- Mostrar porcentajes de completud

13 Consideraciones adicionales

Ubicación del código: https://github.com/manelvf/Transboard.git

Ubicación de la aplicación: http://transboard.manelvf.com

14 Glosario

Términos relevantes del presente estudio:

- **AJAX**: Tecnología empleada para comunicar asíncronamente con un servidor web.

- **Cliente web**: Cualquier tipo de navegador que permita renderizar páginas a través del protocolo HTTP.

- **Gettext**. Aplicación informática empleada en aplicaciones de código abierto para realizar las tareas de internacionalización.

- **Internacionalización**. Proceso que permite mostrar los textos de una aplicación en diferentes idiomas, según las preferencias del usuario.

- **Licencia MIT**. Licencia de uso para el código informático que establece los límites y condiciones de su uso.

- **Modelo Vista Controlador**. Patrón de desarrollo de aplicaciones web que permite separar los contenidos de su representación.

- **Ruby**. Lenguaje de programación elaborado por Yukihiro Matz basado en características del lenguaje LISP.

- **Servidor Web**. Aplicación informática que permite suministrar los contenidos almacenados en un equipo informático a los clientes que se conecten a éste.

- **Sinatra**. Framework web que permite agilizar el desarrollo de aplicaciones web.

15 Bibliografía

Elementos bibliográficos de los que el autor se ha servido durante la elaboración de este trabajo:

[1] **Lenguaje Ruby**. Sitio oficial del lenguaje. http://www.ruby-lang.org/en/

[2] **Framework Sinatra**. Sitio oficial del framework web. http://www.sinatrarb.com/

[3] **MongoDB**. Sitio oficial de la base de datos MongoDB. http://www.mongodb.org/

[4] Understanding Model View Controller. Explicación del patrón Modelo Vista Controlador. http://www.codinghorror.com/blog/2008/05/understanding-model-view-controller.html

[5] **Gettext**. Página oficial del proyecto Gettext. http://www.gnu.org/software/gettext/

[6] **Rack**. Página oficial del proyecto Rack, que proporciona una arquitectura modular para las aplicaciones web desarrolladas en Ruby https://github.com/rack/rack

[7] **NoSQL**. Explicación de las bases de datos no basadas en SQL, su motivación y desarrollo. http://en.wikipedia.org/wiki/NoSQL

[8] **How to translate with Gettext .po and .pot files. Explicación de cómo traducir los archivos de internacionalización .po .pot** http://www.icanlocalize.com/site/tutorials/how-to-translate-with-gettext-po-and-pot-files/

[9] **Phusion Passenger**. Módulo que permite interconectar un servidor Apache con un servicio Ruby. http://www.modrails.com/

[10] **Apache**. Servidor web empleado para servir las aplicación web. http://httpd.apache.org/

Apéndice I. Lista de figuras

Ordenadas por capítulo y orden de aparición:

- **Figura 4.1**. Lista de Tareas
- **Figura 4.2.** Diagrama GANTT
- **Figura 8.3**. Código del modelo de datos principal
- **Figura 8.4**. Código del modelo de datos de colaboración
- **Figura 8.5**. Figura del flujo de actividad principal
- **Figura 8.6**. Figura del flujo de actividad de opciones
- **Figura 10.7**. Pantalla de identificación (login)
- **Figura 10.8**. Pantalla de registro (signup)
- **Figura 10.9**. Pantalla de lista de proyectos
- **Figura 10.10**. Pantalla de subida de nuevo proyecto
- **Figuras 10.11 y 10.12**. Pantalla de edición de traducciones
- **Figura 10.13**. Pantalla de administración de colaboradores
- **Figura 10.14**. Pantalla de descarga del archivo final