



TRABAJO FIN DE GRADO  
INGENIERÍA EN INFORMÁTICA

# Aplicación kanban para empresa de traducción

---

**Autor**

Jose Manuel Linares Rojas

**Director**

Jose María Guirao Miras



ESCUELA TÉCNICA SUPERIOR DE INGENIERÍAS INFORMÁTICA Y DE  
TELECOMUNICACIÓN

---

Granada, Junio de 2019







# Aplicación kanban para empresa de traducción

---

<https://github.com/ncortex/kanboardmd>

## **Autor**

Jose Manuel Linares Rojas

## **Director**

Jose M<sup>a</sup> Guirao Miras



# Aplicación kanban para empresa de traducción

Jose Manuel Linares Rojas

**Palabras clave:** kanban, traducción, gestión de proyectos, aplicación web

## Resumen

Actualmente existen muchas plataformas genéricas de organización de equipos y reparto de tareas. Estas plataformas permiten a los responsables de las empresas organizar y repartir el trabajo entre los empleados; así como consultar los trabajos asignados a cada empleado, asignar preferencias a las tareas o consultar estadísticas de productividad de cada usuario.

Sin embargo, al ser genéricas, estas plataformas suelen ser demasiado complejas para adecuarse a cualquier tipo de organización y a cualquier forma de trabajar, a la vez que se quedan cortas en funciones más específicas que necesita un tipo de empresa, siendo muy difícil, sino imposible, extenderlas para incluir estas funcionalidades.

Este proyecto pretende desarrollar una plataforma de gestión de tareas específica para una empresa de traducción. Este tipo de empresas suelen tener diferentes grupos de trabajo y empleados con diferentes capacidades (idiomas que pueden hablar, disponibilidad horaria,...) que no son fáciles de manejar con las plataformas tradicionales como Jira o Trello.

La aplicación permitirá a los responsables del grupo de trabajo asignar las tareas pendientes a los distintos traductores y revisores mediante una interfaz clara estilo Kanban que muestre toda la información que puedan necesitar para hacer esta asignación. Los traductores, a su vez, podrán acceder a su tablero para ver las tareas que tienen asignadas, cambiar su estado, consultar detalles,...

Además la aplicación dispondrá de un sistema de métricas y estadísticas que permitirá a los responsables controlar la productividad del grupo de trabajo o de un empleado concreto.

Para que sea multiplataforma y no obligar a los usuarios a utilizar un sistema concreto se va a desarrollar como una aplicación web que se ejecute en un servidor y a la que se acceda mediante un navegador web. Además se publicará con una licencia de software libre.





# Kanban application for translation company

Jose Manuel Linares Rojas

**Keywords:** kanban, translation, project management, web application

## Abstract

Nowadays there are a lot of generic platforms dedicated to organizing teams and scheduling tasks. These platforms allow the managers of a company to organize the work, check the tasks assigned to each employee, change the priority of tasks and see statistics of productivity for each user.

However these generic platforms are usually too complex to adapt for every kind of organization and to every working method; at the same time, they fall short in other specific features needed by a company. And it is too difficult, or even impossible, to extend them to include these features.

In this project I want to develop a task management platform specifically designed to be used in translation companies. This kind of business usually has different work-groups and employees with various capabilities (Spoken languages, time availability, ...) and they are not easy to manage with existing platforms like Trello or Jira.

This new application will allow the company managers to assign the pending tasks to the translators and reviewers through a clear Kanban-style interface which will show all relevant information to make this assignment. Translator and reviewer users can also access the Kanban board to view their assigned tasks, to change status of tasks, check task details,...

In addition, the application will provide statistics and metrics to monitor the productivity of the employees.

The program will be a web application to keep users from using a specific system and allowing them to access it through their web browser. The program will be released as free software.



---

Yo, **Jose Manuel Linares Rojas**, alumno de la titulación GRADO EN INGENIERIA INFORMÁTICA de la **Escuela Técnica Superior de Ingenierías Informática y de Telecomunicación de la Universidad de Granada**, con DNI 76653363K, autorizo la ubicación de la siguiente copia de mi Trabajo Fin de Grado en la biblioteca del centro para que pueda ser consultada por las personas que lo deseen.

Fdo: Jose Manuel Linares Rojas

Granada a 17 de Junio de 2019.



---

D. **José María Guirao Miras**, Profesor del Departamento de Lenguajes y Sistemas Informáticos de la Universidad de Granada.

**Informa:**

Que el presente trabajo, titulado ***Aplicación kanban para empresa de traducción***, ha sido realizado bajo su supervisión por **Jose Manuel Linares Rojas**, y autorizo la defensa de dicho trabajo ante el tribunal que corresponda.

Y para que conste, expide y firma el presente informe en Granada a 17 de Junio de 2019.

**El director:**

**José María Guirao Miras**



# Agradecimientos

A todos los que pensaron, aplicaron y apoyaron el Plan Bolonia, por hacerlo pensando en los intereses de los estudiantes, y no en los suyos propios.









# Capítulo 1

## Introducción

Kanban es un método de gestión de proyectos que busca sobre todo las entregas a tiempo, sin sobrecargar a los encargados de realizarlas. En este enfoque, cada tarea desde que se define por parte del cliente hasta su entrega, se muestra a todos los miembros del equipo para que ellos cojan el trabajo a realizar de una cola.

La parte central de kanban es el tablero. Este se organiza en columnas por las que se van moviendo las tareas según su estado. Al recibir una nueva tarea se coloca en la columna 'New' y cuando se van completando los diferentes pasos necesarios para completar la tarea por parte del equipo, se van moviendo de columna, normalmente de izquierda a derecha. Al llegar a la última columna, la tarea se da por terminada y se puede archivar. De esta forma, de un vistazo al tablero se puede observar las tareas pendientes actualmente y su progreso.

Normalmente las columnas que componen un tablero Kanban son: New, To-Do, Doing y Done; pero este esquema puede variar según el proyecto, por ejemplo con columnas 'To Review' o 'Blocked'. Lo importante es que estas columnas representen los diferentes estados que puede tener una tarea en el flujo de trabajo del equipo y que los miembros del equipo tengan claro que condiciones se tienen que dar para que una tarea pase al siguiente estado.

Cada tarea tiene además una persona asignada. Esta será la responsable de realizar la tarea, al menos hasta el siguiente estado.

Una vez que la tarea se da por completada y se archiva, se pueden sacar métricas y estadísticas de la productividad del equipo estudiando el tiempo que ha pasado la tarea en cada columna según el usuario que la tuviese asignada.

Para organizar un proyecto kanban se suelen utilizar aplicaciones informáticas que automaticen procesos, permitan el acceso a las tareas pendientes desde diferentes puntos a la vez, notifiquen a los usuarios de sus tareas y recopilen estadísticas del proyecto. Para esto existen muchas aplicaciones, con diferentes aproximaciones, pero el uso de una u otra no debería

condicionar la forma de trabajar del equipo a un flujo de trabajo concreto, sino que la herramienta debería ser suficientemente flexible para adaptarse a las particularidades de cada proyecto. Además, a menudo las herramientas pecan de ofrecer demasiadas funciones que nunca o casi nunca se van a usar, sobrecargando la interfaz y haciendo el uso de la aplicación más lento y tedioso.

En el caso concreto de una empresa de traducción, cada tarea debe tener cierta información relativa a la cantidad de palabras a traducir y su dificultad, pasos que debe seguir una tarea antes de darse por completada, idiomas de origen y destino,... y el tablero kanban debe ser capaz de automatizar ciertos procesos relativos a la forma de trabajar de esta empresa.

Por ello, este proyecto pretende desplegar una aplicación estilo kanban adaptada exclusivamente al flujo de trabajo de una empresa de traducción.

## Capítulo 2

# Estado del arte

Actualmente las herramientas existentes para el reparto de tareas entre los miembros de un equipo son genéricas y no he encontrado ninguna que esté enfocada directamente a proyectos de traducción. Aunque estas herramientas se pueden utilizar con este propósito, les faltan características deseables en este tipo de proyectos.

### 2.1. Jira

Jira<sup>1</sup> es una herramienta en línea para la administración de tareas de un proyecto, el seguimiento de errores e incidencias y para la gestión operativa de proyectos. Es software propietario desarrollado y mantenido por Atlassian y, aunque se puede probar de forma gratuita, es de pago. Aunque es bastante flexible, se suele utilizar en proyectos de desarrollo de software, pues muchas de las opciones que tiene están enfocadas a este campo, como la integración con GIT o la posibilidad de gestionar proyectos con metodología scrum.

---

<sup>1</sup><https://es.atlassian.com/software/jira>

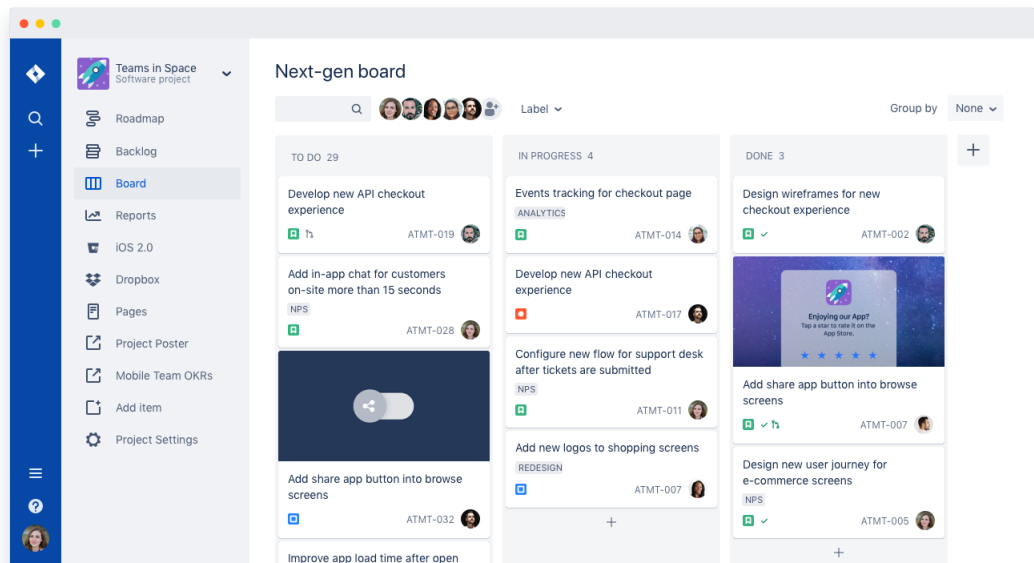


Figura 2.1: Interfaz del tablero de Jira. Fuente: atlassian.com

## 2.2. Trello

Trello<sup>2</sup> es un software de administración de proyectos con interfaz web, cliente para iOS y android para organizar proyectos. También es software propietario aunque la mayoría de las funciones son gratuitas. Tiene más versatilidad que Jira, pues no está enfocado a un tipo de proyecto concreto.

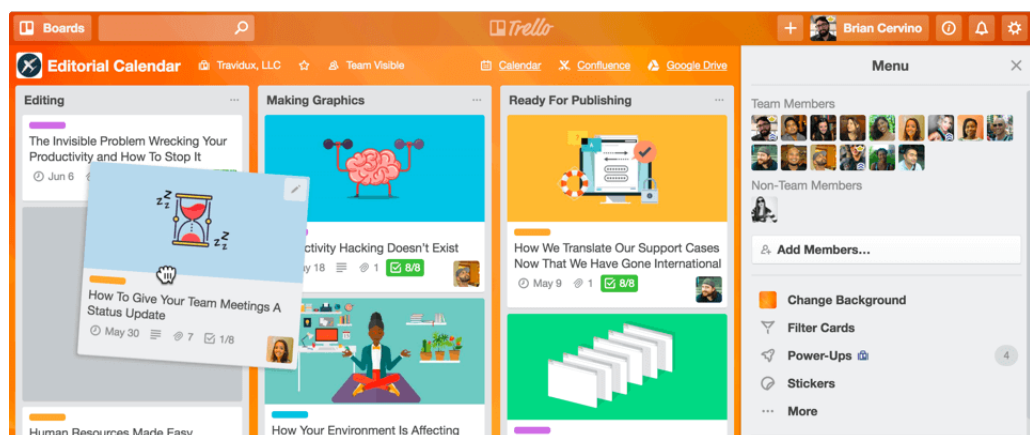


Figura 2.2: Interfaz del tablero Trello. Fuente: trello.com

<sup>2</sup><https://trello.com>

## 2.3. Wekan

Wekan<sup>3</sup> es un proyecto que se está desarrollando activamente como alternativa de software libre a Trello. La interfaz es muy parecida, aunque algunas de las funciones secundarias difieren. Según sus desarrolladores la principal diferencia con trello es la licencia MIT, licencia de software libre que, entre otras cosas, te permite instalarlo en tu propio servidor.

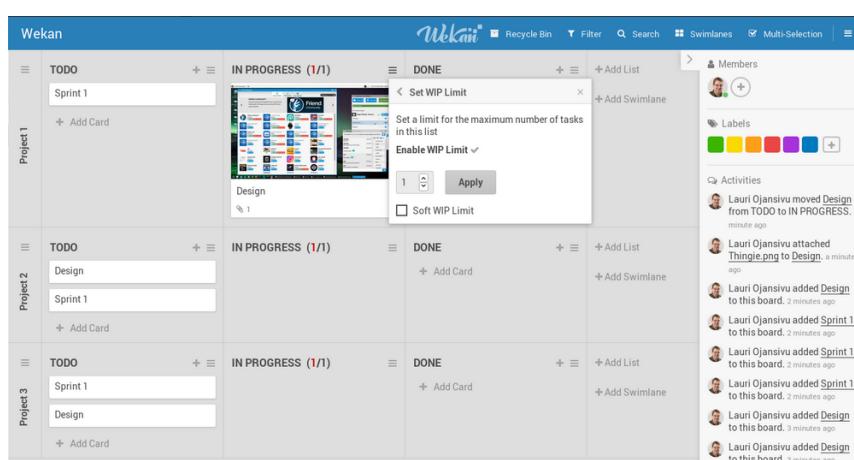


Figura 2.3: Interfaz del tablero Wekan en el que las tarjetas se organizan en 3 proyectos diferentes. Fuente: <https://wekan.github.io>

## 2.4. Restya

Restya<sup>4</sup> también es una plataforma parecida a Trello pero sin intentar emularlo completamente. Tiene algunas funcionalidades extra como la posibilidad de definir grupos de usuarios, chat integrado o la posibilidad de instalar plugins. Aunque es open source, muchos de los plugins son propietarios y de pago.

<sup>3</sup><https://wekan.github.io>

<sup>4</sup><https://restya.com>

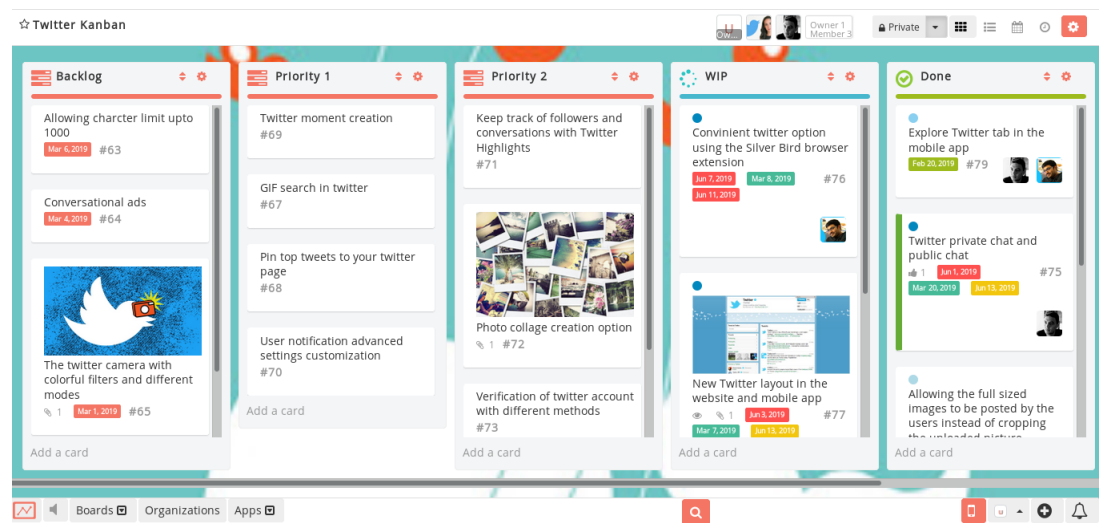


Figura 2.4: Interfaz del tablero de Restya. Fuente: <https://restya.com/board/demo>

## 2.5. Kanboard<sup>5</sup>

Es un proyecto de software libre escrito en PHP que proporciona un tablero kanban básico, así como funcionalidades para filtrar, buscar y listar tareas. Tiene un sistema de plugins que hace fácil ampliar su funcionalidad.

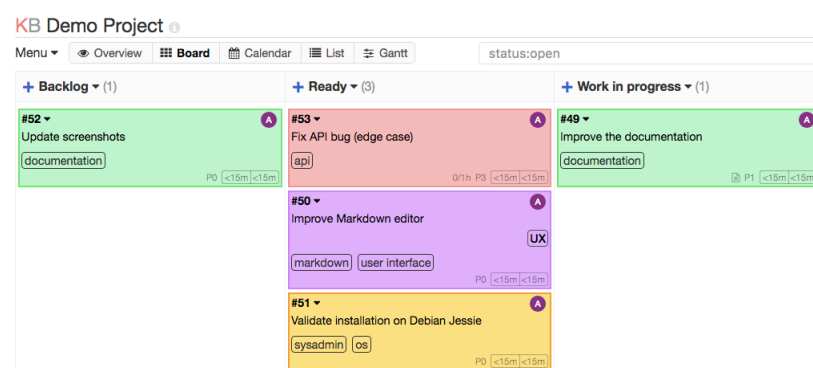


Figura 2.5: Interfaz del tablero de Kanboard con algunos plugins activados. Fuente: <https://kanboard.org>

<sup>5</sup><https://kanboard.org>



## 2.6. Solo.

Solo<sup>6</sup> es un tablero kanban mucho mas sencillo que los anteriores y no tiene demasiadas funciones extra, solo un tablero kanban con un diseño muy sencillo. Las tareas solo tienen un título, una descripción una fecha de vencimiento y una persona asignada; sin la posibilidad de añadir ningún campo adicional. La licencia permite el uso y modificación del programa, pero no permite la redistribución.

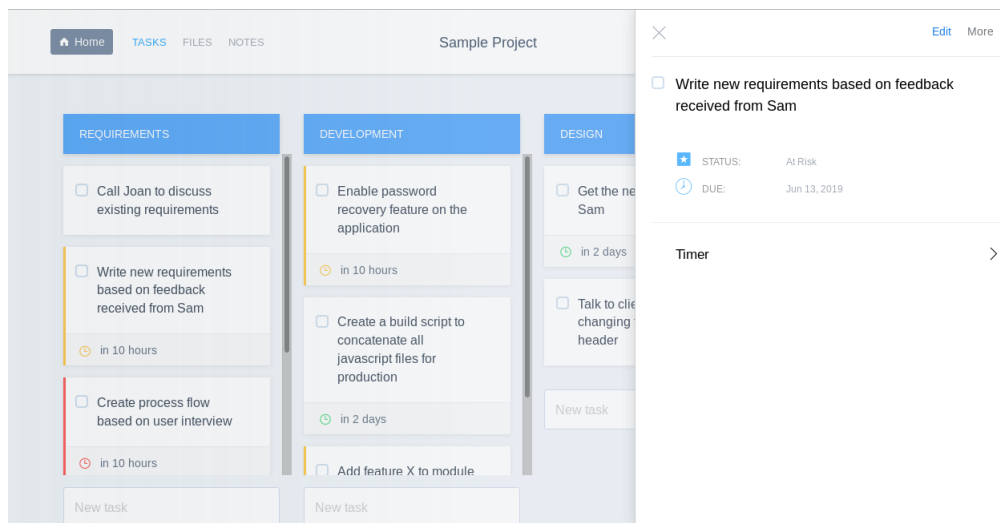


Figura 2.6: Interfaz del tablero Solo. Fuente: <http://getsoloapp.com/demo>

---

<sup>6</sup><http://getsoloapp.com>



## Capítulo 3

# Objetivos

El objetivo final del proyecto es desarrollar una plataforma Kanban que cubra las necesidades concretas de esta empresa de traducción tal y como están definidas en los Requisitos. Para no reinventar la rueda, se va a hacer uso de las herramientas de software libre disponibles en vez de empezar a desarrollar de cero.

Para definir los requisitos de la plataforma se acordó una reunión con la dirección y los empleados de la empresa. En esta reunión se acordaron las funcionalidades que debe incorporar para aceptarla y empezar a usarla como plataforma de gestión en su día a día.

Los objetivos mas importantes que busca la empresa en la plataforma es que sea simple y fácil de usar, es decir, que no tenga funciones extra; que sea rápida, y que se adapte a la forma de trabajar de la empresa. Además, buscan que la organización se pueda llevar a cabo por varios usuarios a la vez.

Su objetivo último es ganar productividad al ahorrar tiempo de organización de sus empleados, pues actualmente están llevando a cabo esta organización de forma manual, ayudándose únicamente de una hoja Excel. Además, quieren que la aplicación sea mantenible, por lo que la organización del código debe ser clara.

### 3.1. Especificación de Requisitos

#### 3.1.1. Requisitos Funcionales

- La aplicación debe mostrar un tablero Kanban en el que se puedan ver las tareas organizadas por columnas según el estado en el que se encuentren (Por hacer, a traducir, a revisar, terminadas o en espera/-bloqueadas)
- La aplicación debe ser capaz de autenticar a los usuarios.

- La aplicación debe manejar distintos roles que podrán tener los usuarios (Resource Manager, gestor y traductor)
- El usuario debe poder consultar todas las tareas, así como filtrarlas.
- Los usuarios con privilegios de Resource Manager podrán crear tareas y asignarlas a un gestor.
- Los usuarios asignados como gestores de una tarea pueden asignarle un usuario traductor y un usuario revisor.
- Los usuarios asignados como traductores de una tarea solo pueden marcarla como traducida.
- Los usuarios asignados como revisores de una tarea solo pueden marcarla como revisada.
- Una vez que una tarea pasa por todos los pasos y se marca como completada, aparecerá en la columna de terminadas y podrá ser archivada por el Resource Manager.
- Cada tarea deberá incluir los siguientes detalles:
  - Cliente
  - Product
  - Subproduct
  - Project number
  - Package number
  - PO code
  - Amount PO
  - Cantidad de palabras (MT/New/Fuzzy/100 % o Ponderadas)
  - Tipo de tarea: Translation, Review, LQA, LSO, QA y Rebranding
  - Tipo de contenido: HC, UI, Marketing, Legal, Subtitles, API-Technical, Glossary y Other.
  - Locale (idiomas de origen y destino)
  - Fecha de recepción de la tarea
  - Fecha y hora límite de la tarea
  - Herramienta (campo de texto)
  - Link
  - PM Cliente (campo de texto)
- Las tareas cuya hora límite esté entre las 8:00 y las 17:00 deberán resaltarse en otro color.

- El título de la tarea debe definirse a partir de los valores de Subproducto, numero de proyecto y numero de paquete de la siguiente manera: "Subproducto - Project number (Package number)"
- La aplicación debe mandar notificaciones a los usuarios por email. Estas notificaciones serán configurables.
- La aplicación debe tener además un sistema de notificaciones interno.
- El usuario deberá poder filtrar las tareas según criterios personalizados.
- La aplicación debe controlar el tiempo que pasa cada tarea en cada fase de la cadena.

### 3.1.2. Requisitos No Funcionales

- La aplicación debe ser accesible desde el navegador.
- La aplicación debe instalarse facilmente, ofreciendo un contenedor con todas las dependencias necesarias.
- El codigo de la aplicacion debe estar suficientemente documentado para facilitar su mantenimiento por terceras personas.
- La aplicación debe ser simple y facil de usar, ofreciendo solo las funcionalidades necesarias.



## Capítulo 4

# Planificación

El primer paso será reunirse con la empresa para tener una primera idea de la funcionalidad que buscan en la plataforma, que partes son mas importantes y cuales mas secundarias. Esta reunión también servirá para ver que forma de trabajar tienen actualmente para poder desarrollar un sistema que encaje con esta metodología.

La información que saquemos de esta reunión la usaré para definir los requisitos que serán puestos en común con el cliente y que serán la base a partir de la cual empezar el diseño de nuestra plataforma.

Luego se hará un análisis de las herramientas existentes que puedan ser de utilidad en este proyecto o que puedan servir de punto de partida. También se estudiarán las plataformas que están siendo utilizadas para proyectos similares para ver sus debilidades y sus puntos fuertes.

Una vez hecho esto se va a realizar un prototipo no funcional para mostrar la interfaz de usuario y asegurar que la imagen del proyecto que tiene el cliente se va a corresponder con el programa final. Este prototipo mostrará como se va a utilizar cada una de las funcionalidades definida en los requisitos. Este prototipo será puramente visual y servirá únicamente para acordar entre las partes el aspecto y distribución que tendrán los elementos dentro de la aplicación.

Luego tendrá lugar otra reunión con el cliente en la que se acordará que el prototipo está bien y contempla todas las funcionalidades requeridas, y se empezará con la implementación propiamente dicha. Durante el desarrollo se implementarán también, en la medida de lo posible, test que comprueben la funcionalidad de las opciones más importantes de forma automática.





## Capítulo 5

# Análisis

Para la implementación de la plataforma se va a tomar Kanboard como base y a partir de ahí se va a desarrollar el resto de la aplicación. He elegido Kanboard porque es una aplicación de software libre cuya licencia (MIT) me permite tanto la modificación como la redistribución del código sin tener que pedir permiso. Además, la herramienta resuelve los problemas que se podrían plantear a la hora de desarrollar un tablero kanban y me proporciona un tablero básico para modificarlo y extenderlo. Además, es una aplicación de navegador, lo cual proporciona la ventaja de poder utilizarlo desde cualquier dispositivo.

Según su propia documentación, se centra en la simplicidad, por lo que el número de funcionalidades se ha limitado a propósito para conseguir una herramienta sencilla que haga ni mas ni menos de lo estrictamente necesario. Permite ver el tablero kanban personalizado para cada usuario, mover tareas entre columnas, añadir información a las tareas, así como buscar y filtrar tareas. Otros servicios con los que cuenta kanboard que se pueden reutilizar son el sistema de usuarios y privilegios y el sistema de traducción y localización de la aplicación a distintos idiomas.

Además, la estructura del código es bastante intuitiva y provee un sistema de plugins para ampliar la funcionalidad de forma natural, sin modificar el código base. Esto nos permitirá exportar las nuevas funcionalidades que implementemos para que sean instalables en otras instancias de Kanboard. Además, hará mucho más sencilla la actualización del código base, pues este proyecto está actualmente en desarrollo activo y actualizarlo significaría perder las modificaciones que se le hubieran hecho al núcleo.

### 5.1. Estructura del programa

El código de Kanboard intenta seguir una arquitectura MVC(modelo vista controlador) en la que se separa claramente los datos y su declaración, la lógica del programa y las entradas y salidas en 3 componentes distintos.

En ciertas partes no se cumplen algunas de las máximas de esta arquitectura, como es conseguir bajo acoplamiento entre los componentes, pero en general, el proyecto es mantenible y extensible.

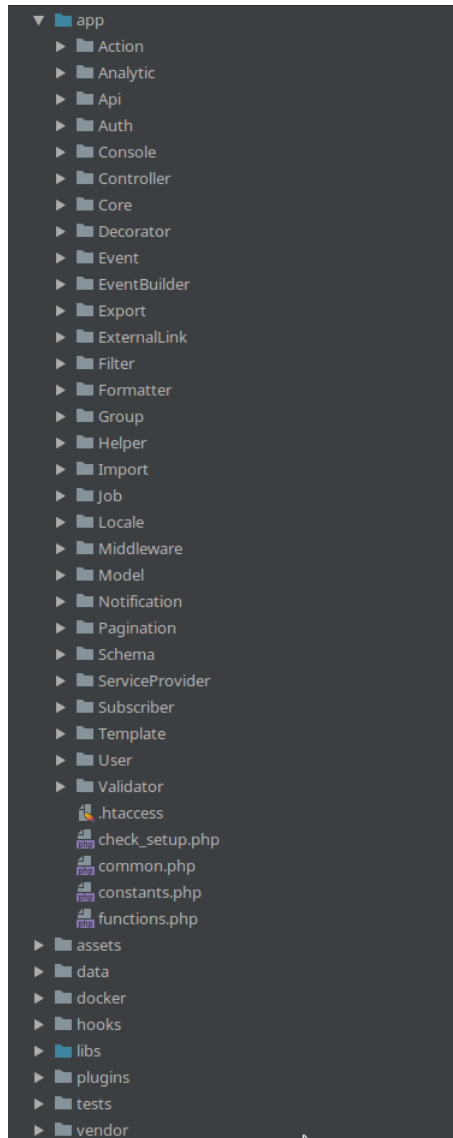


Figura 5.1: Podemos hacernos una idea de como está organizado el código viendo las carpetas en las que se distribuye.

#### 5.1.1. Sistema de plugins

El sistema de plugins de Kanboard es uno de los motivos principales por los que he elegido este proyecto como base, pues permite extender la funcionalidad de forma facil y modular, sin necesidad de modificar el código

del núcleo. Funciona de la siguiente forma:

- Kanboard lee todos los subdirectorios de la carpeta plugins buscando archivos de declaración de plugins. Los plugins son clases que heredan de una clase BasePlugin.
- Comprueba que el plugin sea compatible con la versión actual de kanboard y, si lo es, llama a la función initialize de la clase.
- En esta función se registran en la aplicación los nuevos elementos que necesita el plugin. Estos elementos pueden ser acciones, plantillas de vistas y componentes, rutas, modelos, filtros,... El plugin además puede modificar la estructura de la base de datos con migraciones. Estas migraciones permiten hacer modificaciones incrementales y reversibles del esquema de la base de datos.

```
35
36 //Project
37 $this->template->hook->attach( hook: 'template:project:sidebar', template: 'metaMagik:project/sidebar');
38
39 //Task
40 $this->template->hook->attach( hook: 'template:task:sidebar:information', template: 'metaMagik:task/sidebar');
41 $this->template->hook->attach( hook: 'template:board:task:icons', template: 'metaMagik:task/footer_icon');
42 $this->template->hook->attach( hook: 'template:task:form:second-column', template: 'metaMagik:task/rendermeta');
43 $this->template->hook->attach( hook: 'template:task:details:bottom', template: 'metaMagik:task/metasummary');
44
45 $this->template->setTemplateOverride( original_template: 'project_header/dropdown', new_template: 'metaMagik:project_header/d');
46 $this->template->setTemplateOverride( original_template: 'export/tasks', new_template: 'metaMagik:export/tasks');
47
48 //Routes
49 $this->route->addRoute( path: 'export/metatasks/:project_id', controller: 'NewExportController', action: 'tasks');
50
```

Figura 5.2: Ejemplo de plugin haciendo uso de hooks para integrar nuevos elementos a las plantillas, sobrescribir plantillas y declarar nuevas rutas

### 5.1.2. Inyección de dependencias

La inyección de dependencias es un patrón de diseño orientado a objetos en el que, en lugar de ser las propias clases las que creen objetos, se suministran objetos a las clases por parte de otra clase contenedora. Esos objetos cumplen contratos que necesitan nuestras clases para poder funcionar (de ahí el concepto de dependencia).

Aunque los objetos a inyectar se pueden referenciar mediante sus clases, esto se considera una mala práctica, pues provoca un fuerte acoplamiento entre las clases. Por eso, en la inyección de dependencias, se recomienda usar interfaces.

Como inyector de dependencias, el Kanboard utiliza Pimple <sup>1</sup>. Este componente permite definir servicios y usarlos en el proyecto donde vayan haciendo falta. El hecho de que el proyecto cuente con un inyector de dependencias

<sup>1</sup><https://pimple.symfony.com>

hace que sea mas facil la modularización y refactorización del código, lo que seguramente ha ayudado a que la estructura del programa sea tan clara.

### 5.1.3. Sistema de rutas

El proyecto cuenta también con un enrutador que maneja las rutas de la aplicación, las vincula con funciones concretas de los controladores y se ocupa de manejar los parámetros GET que puedan formar parte de la ruta. También se ocupa de realizar 'URL Rewriting', es decir, transformar las URLs de la aplicación para hacerlas mas vistosas a la vez que añade una capa de abstracción entre los archivos usados para generar la página y los enlaces que se muestran al usuario. Esto permite que los enlaces de la aplicación sean del estilo:

```
example.com/project/1/taskedit/1
```

Estos enlaces son más cortos y fáciles de entender que los enlaces tradicionales:

```
example.com/?controller=TaskModificationController&action=edit&task_id=1&project_id=1
```

Para definir una ruta en la aplicación desde un plugin, se usa la función `AddRoute` del helper `route`:

```
$this->route->addRoute('/mi/ruta/:mi_variable', 'miController',  
'miAction', 'miplugin');
```

Luego, desde las plantillas se renderizan los enlaces usando el helper `url`:

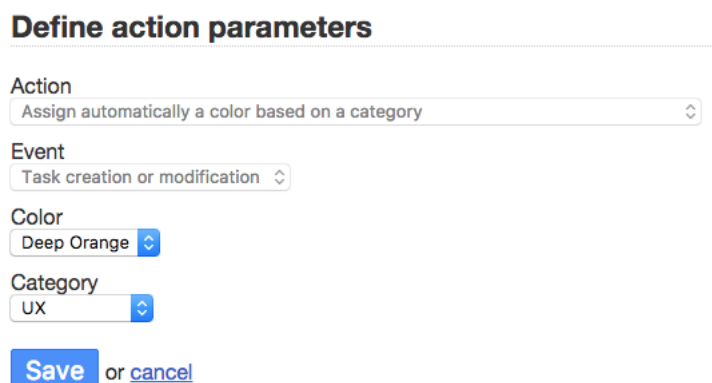
```
$this->url->link('Mi link', 'miController', 'miAction',  
array('plugin' => 'miplugin', 'mi_variable' => 'valor'));
```

Lo que generaría un enlace con el texto 'Mi link' que enlaza a `http://example.com/mi/ruta/valor`

### 5.1.4. Sistema de eventos

Kanboard también dispone de un sistema de Triggers o acciones automáticas que se disparan cuando ocurre un evento concreto. Internamente, Kanboard usa el componente `EventDispatcher` de Symfony ( <https://symfony.com/doc/2.3/con>) para manejar los eventos. Es facil definir nuevos eventos desde los plugins, así como registrar nuevos callbacks que se ejecuten cuando suceda un evento:

```
$this->eventManager->register('my.event.name', 'Descripcion');  
  
$this->on('my.event.name', function($container) {  
    // Código que se lanzará cuando ocurra el evento  
});
```



**Define action parameters**

Action  
Assign automatically a color based on a category

Event  
Task creation or modification

Color  
Deep Orange

Category  
UX

**Save** or [cancel](#)

Figura 5.3: Interfaz para definir nuevas acciones automáticas asociadas a un tipo de evento. Fuente: <http://kanboard.org>

#### 5.1.5. Control de usuarios y privilegios

La plataforma tiene un sistema de control de usuarios y de privilegios implementado que permite la autenticación de usuarios tanto de forma local, con usuario y contraseña, como de formas externas como autenticación con Google o Facebook. Se pueden definir también nuevos proveedores (Authentication Providers) para hacer login con otras plataformas de forma fácil.

Todo lo relativo al registro de usuarios y posterior autenticación, así como funciones extra como el cambio de contraseña en caso de olvidarla están ya implementados en Kanboard.

Cada usuario tiene un rol dentro del proyecto que le da ciertos privilegios. Los roles y privilegios asociados se configuran a nivel de proyecto y se pueden definir nuevos.

Aunque mi aplicación no va a hacer uso de grupos, Kanboard permite también la creación de grupos de usuarios para asignar privilegios y restricciones en bloque.

#### 5.1.6. Traducciones

Kanboard utiliza un sistema de traducción de cadenas para poder cambiar el idioma de la aplicación. Este sistema se utiliza también en los plugins y su funcionamiento es muy sencillo. Se basa en diccionarios en los que la clave es la cadena original y el valor es la cadena traducida a un idioma concreto. Estos diccionarios se encuentran en el directorio `/locale/xx_XX/translations.php` o en `/plugins/plugin/Locale/xx_XX/translations.php`; donde `xx_XX` es el idioma en cuestión. Son archivos PHP que devuelven un array de parejas clave-valor.

Para imprimir una cadena traducida desde las vistas, se usa la función `t()` :

```
<?= t('Remove Metadata'); ?>
```

Si no se encuentra ninguna coincidencia para una cadena concreta en el idioma que tiene configurado el usuario, se busca una traducción en el idioma por defecto. Si tampoco se encontrase ahí, se imprime la cadena tal cual sin traducir.

```
1 <?php
2
3 return [
4     'Metadata'           => 'Metadatos',
5     'Key'                 => 'Clave',
6     'Value'              => 'Valor',
7     'No metadata'        => 'No hay metadatos',
8     'Remove Metadata'    => 'Eliminar metadatos',
9     'Do you really want to remove this metadata?' => 'Seguro que quieres eliminar estos metadatos?',
10    'Add Metadata'        => 'Añadir metadatos',
11 ];
```

Figura 5.4: Ejemplo de un archivo de traducción a español de un plugin.

#### 5.1.7. Notificaciones

Kanboard incluye un sistema de notificaciones en el que se informa a los usuarios de los últimos cambios que les afectan. Estas notificaciones se manejan internamente en un panel de notificaciones y, además, se envían fuera de la plataforma mediante correo electrónico, mensajería instantánea u otros métodos. Existe la posibilidad de registrar nuevos Handlers de notificaciones para notificar al usuario mediante otros canales.

#### 5.1.8. Proyectos, columnas y tareas

La aplicación permite tener varios proyectos, cada uno con su propio tablero. El tablero es el componente principal donde se organizan las tareas. Permite arrastrar y soltar tareas entre las columnas del tablero, así como cambiar sus atributos. Cada tarea tiene campos genéricos como título, descripción, responsable, creador, categoría, fecha de creación,... pero para añadir campos extra hay que usar plugins.

En este proyecto se limitará la capacidad de los usuarios para arrastrar y soltar tareas entre columnas, permitiéndoles solamente mover las tareas que tengan asignadas actualmente a la siguiente columna.

#### 5.1.9. Filtro y motor de búsqueda de tareas

Uno de los puntos fuertes de Kanboard es su motor de búsqueda de tareas, que permite configurar que tareas exactamente queremos que se muestren en nuestro tablero. Además de filtrar por los campos básicos de la tarea

(Título, persona asignada,...) permite acumular los filtros y guardarlos con un nombre identificativo para poder usarlos más adelante.

Uno de los requisitos de la aplicación era que se pudiese filtrar de forma fácil para ver solamente las tareas que nos interesen, así que este es un motivo más para elegir Kanboard como punto de partida.

#### 5.1.10. API

Además de utilizar la aplicación mediante la interfaz web, también se puede interactuar con la aplicación mediante una API. Esta API utiliza el protocolo JSON-RPC (Json Remote procedure Call) y una autenticación mediante token o combinación usuario/contraseña. Puesto que la aplicación web hace uso de esta API, todo lo que se puede hacer en la web se puede hacer con la API.

Kanboard permite añadir nuevos métodos a la API de forma fácil:

```
$this->api->getProcedureHandler()->withCallback('mi_metodo', function(){  
    // Cuerpo del método  
});
```

#### 5.1.11. Sistema de migraciones

Kanboard incluye un sistema de migraciones del esquema de la base de datos. Este sistema maneja los cambios que se van produciendo en el esquema de la base de datos y permite revertirlos cuando sea necesario. Con este sistema resulta muy fácil introducir cambios en la base de datos desde los plugins para añadir los campos que nos hagan falta, permitiendo también revertir estos cambios si el plugin se desinstala.

```
//Ejemplo de migración de un plugin que inserta 4 columnas nuevas en la tabla tasks  
namespace Kanboard\Plugin\Fordchain\Schema;  
  
use PDO;  
const VERSION = 1;  
  
function version_1(PDO $pdo)  
{  
    $pdo->exec('ALTER TABLE ONLY "tasks" ADD COLUMN "translator_id" integer default 0,  
        ADD COLUMN "gestor_id" integer default 0,  
        ADD COLUMN "fordchainStep" integer default 0,  
        ADD COLUMN "reviewer_id" integer default 0;  
    ');  
}
```

### 5.1.12. Hooks

Los hooks (o ganchos) están pensados para cambiar, reemplazar o extender el comportamiento por defecto de Kanboard. Cada hook tiene un nombre único identificativo, y se les puede asignar listeners.

Por ejemplo, en las plantillas que renderizan las distintas páginas de la aplicación hay hooks que permiten añadir información a estas páginas desde los plugins sin tocar el código de la plantilla original.

### 5.1.13. Webhooks

Además de los Hooks, también se pueden definir Webhooks, que sirven para realizar acciones con aplicaciones externas. Básicamente consisten en enviar información a una url externa cada vez que se dispare un evento concreto.

Los webhooks se declaran en Opciones ¿Webhooks y cada vez que se dispara el evento asociado, se codifica la información en JSON y se envía por POST a la URL especificada.

### 5.1.14. Sistema de estadísticas

Kanboard incluye un sistema de estadísticas que guarda información acerca del uso del sistema y los muestra gráficamente; tanto a nivel de tarea como a nivel de proyecto. Mantiene estadísticas acerca de cantidad de tareas en cada columna, tiempo promedio que pasa cada tarea en cada columna, distribución de tareas por usuario,...

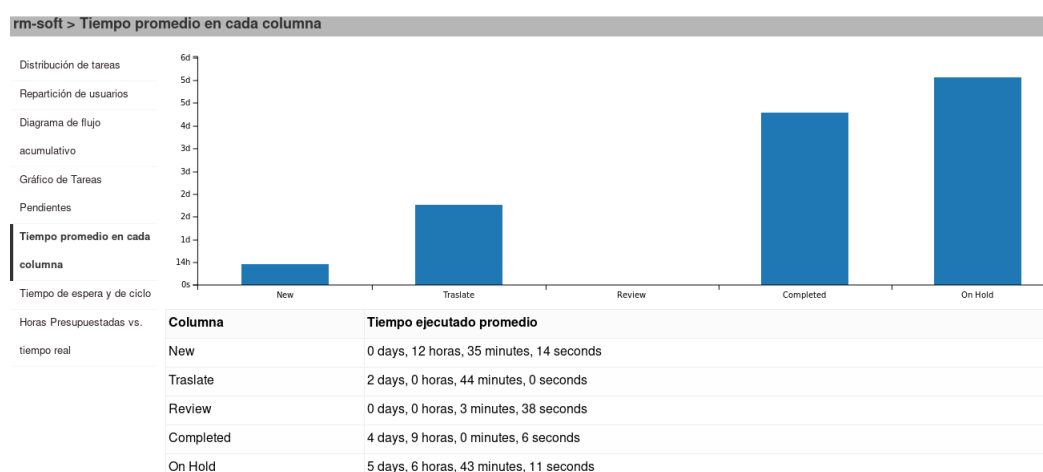


Figura 5.5: Gráfico que representa el tiempo promedio que pasa una tarea en cada una de las columnas del proyecto





# Diseño

### 6.1. Diagrama entidad/relacion

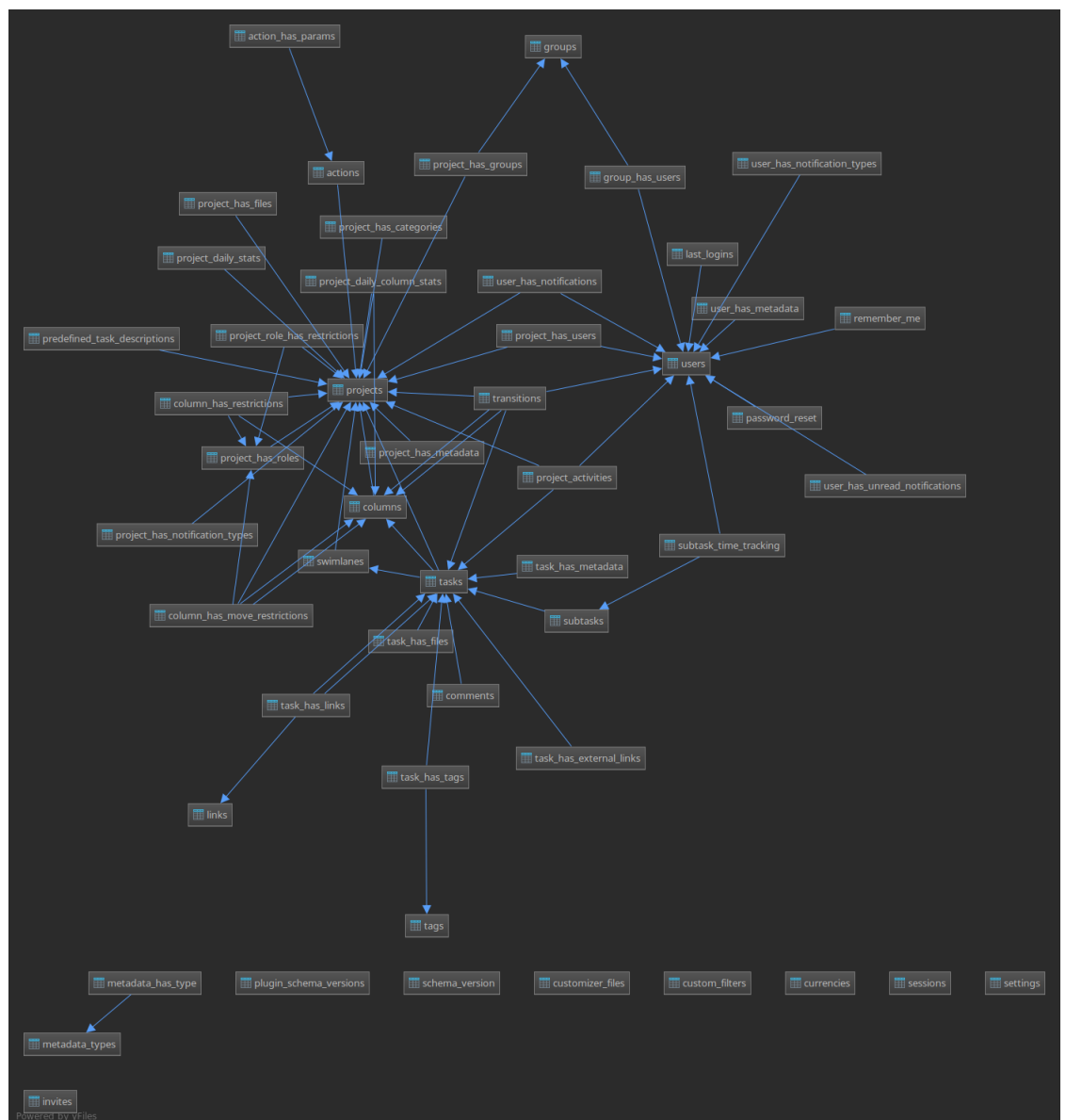


Figura 6.1: Vista global de las relaciones entre las entidades de la base de datos

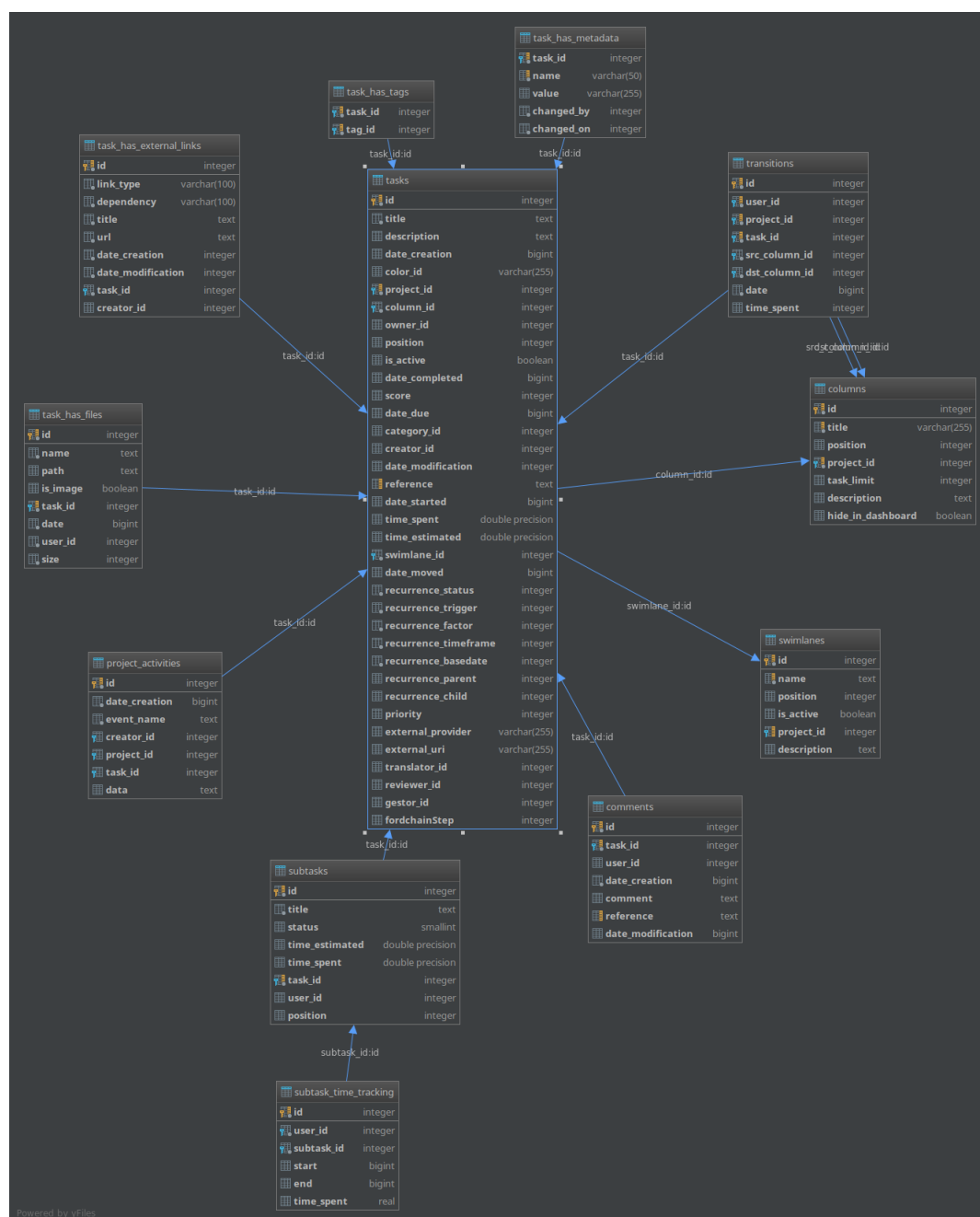


Figura 6.2: Diagrama completo de la entidad tarea y de sus relaciones

EL código de Kanboard sigue una arquitectura MCV, por lo que mis aportaciones seguirán también esta arquitectura, separando bien entre la visualización de los datos, su manejo y su definición.



## Capítulo 7

# Implementación

Gran parte del proceso de implementación va a consistir en ocultar funcionalidades que ofrece Kanboard pero que no son necesarias en la plataforma. Por ejemplo, Kanboard permite la creación de varios proyectos, cada uno con su propio tablero kanban. En este caso se precisa solo de un tablero, por lo que se ocultarán las opciones relativas a crear nuevos proyectos.

Además, aunque la herramienta estuviese pensada en un principio para ser flexible, la verdad es que tiene muchas opciones pensadas para ser usado en proyectos de desarrollo de software. Estas opciones también serán eliminadas y sustituidas por otras orientadas al flujo de trabajo de la empresa del cliente.

Como se ha explicado antes, se va a desarrollar haciendo uso del sistema de plugins que incluye kanboard. Esto, además de hacer mas facil la actualizacion del codigo base, hará que las funcionalidades implementadas sean reutilizables en otros proyectos, pues los plugins serán distribuidos con una licencia libre.

Todos los plugins que voy a usar para cumplir los requisitos están desarrollados desde cero, salvo Metamagik, que es un plugin que ya estaba en desarrollo y al que he añadido alguna funcionalidad.

Para empezar el desarrollo de un nuevo plugin, se recomienda usar el programa cookiecutter. Esta herramienta crea proyectos nuevos a partir de un archivo de plantilla. Ejecutando el siguiente comando:

```
cookiecutter gh:kanboard/cookiecutter-plugin
```

crea toda la estructura de directorios y todos los archivos que definen el nuevo plugin:

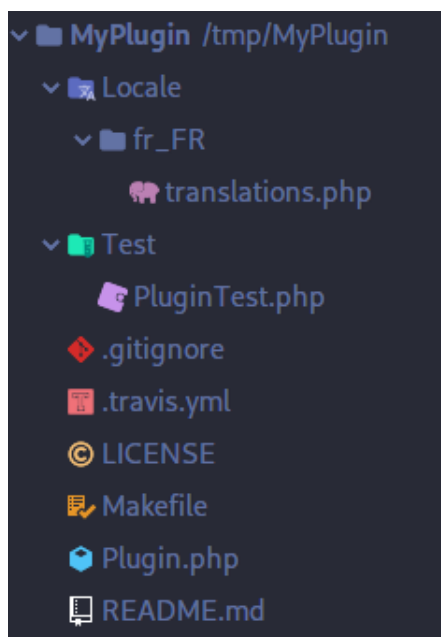


Figura 7.1: Arbol de directorios por defecto de un plugin de Kanboard creado con cookiecutter

La parte central del plugin es la funcion initialize, que es la que será llamada cuando el plugin esté activado. En esta función se definen las personalizaciones de código que llevará a cabo el plugin. Para hacerlo, se utilizan 'hooks' o ganchos.

## 7.1. Metamagik

Metamagik es un plugin que está actualmente en desarrollo activo que permite añadir campos extra a cada tarea. Es muy útil, pues elimina la restricción de que a cada tarea solo le puedes añadir información genérica y te permite añadirle metadatos. Como aún está en una fase temprana del desarrollo, he contribuido con varias opciones que no incluía. Por ejemplo, al principio solo aceptaba campos de tipo texto, dropdown, radio o checkbox y he añadido el tipo número, usuario y select, cuyos valores posibles se leen desde una tabla de la base de datos.

Este plugin me ha permitido añadir los campos mas basicos a la tarea sin necesidad de modificar el esquema de la base de datos. Los datos que se solicitan con este plugin son los de tipo texto, número o desplegable:

- numero de palabras del trabajo
- idioma de origen y destino
- PM(Client)

- Amount(PO)
- Tool
- Link

## 7.2. Just one board

Este plugin se encarga de ocultar la existencia de otros proyectos. Solo permite mostrar el primer proyecto. Además oculta las opciones relativas a crear y administrar proyectos, así como las distintas formas de cambiar de proyecto o de mover tareas entre proyectos.

Para hacerlo, modifica las plantillas de las vistas y los sustituye por otros en los que se han eliminado estas opciones. Internamente Kanboard asocia las tareas a proyectos, y a columnas dentro de esos proyectos, por lo que en vez de eliminar el concepto de proyecto, he creado uno y lo he definido como principal.

Después de hacer login, se redirige al proyecto principal, saltándose la página de gestión de proyectos.

## 7.3. DateFilter

Este plugin añade filtros de búsqueda que permiten filtrar las tareas de acuerdo a su fecha de vencimiento; por ejemplo:

```
due: "tomorrow"  
due: ">2019-06-29"  
due: "+1 month"
```

## 7.4. Minimalistic Theme

Este plugin recoge toda la funcionalidad relativa a la ocultación de funciones. Sustituye las plantillas que generan las distintas partes de la aplicación por otras con menos elementos. También es el plugin responsable de mostrar la información con el formato y diseño requeridos. Además, carga un archivo de estilos CSS para modificar la apariencia de la aplicación.

**New task**

Title

Description

Tags

☐ Create another task

**Save** or [cancel](#)

Color

Assignee

Column

Priority

Complexity

Reference

Original estimate  hours

Time spent  hours

Start Date

Due Date

Figura 7.2: Formulario de crear nueva tarea en Kanboard estándar.

**rm-soft > Nueva tarea**

Locale

Client

Product

Subproduct

Project number

Package number

Extra number

Categoría

Etiquetas

Gerente de proyecto

Traductor

Revisor

MT

New

Fuzzy

100%

Weighed

Fecha de Inicio

Fecha Límite

Referencia

Color

PM(Client)

Amount(PO)

Tool

Link

**Guardar**

Figura 7.3: Formulario de crear nueva tarea en Kanboard con el plugin activado.

## 7.5. Fordchain

Este se podría considerar el plugin principal, pues maneja la logica perteneciente exclusivamente a la forma de trabajar que tiene la empresa. Los requisitos principales de la aplicación eran que se pudiesen crear tareas y que estas fuesen cambiando de responsable y de columna hasta que se den



por finalizadas. En el caso concreto de esta empresa de traducción, cada trabajo pasa por las etapas: por asignar, traduciendo, revisando y finalizado. Cada una de estas etapas se realiza por un usuario concreto. Puesto que en otros proyectos las etapas situadas entre la asignación de cada paso a un miembro del equipo y la finalización de la tarea pueden ser diferentes, se ha desarrollado pensando en poder expandir los pasos necesarios hasta que la tarea se da por finalizada.

He utilizado el sistema de acciones automatizadas (o triggers) de Kanboard que se activan cuando se produce un evento concreto. He extendido este sistema con 2 triggers nuevos que se irán disparando conforme el progreso de la tarea vaya avanzando:

- **Iniciar cadena:** Esta acción se dispara cuando se crea una tarea nueva. Inicializa las variables internas necesarias y asigna la tarea al usuario marcado como gestor. En caso de que estén establecidos el traductor y el revisor, se asignará al traductor y se colocará en la columna de traducción.
- **Siguiente paso:** Esta acción asignará la tarea al siguiente usuario que corresponda y la moverá a la siguiente columna. En caso de que la tarea ya se encuentre en el último paso, en vez de moverse de columna, se cerrará y se archivará.

He creado además un evento "Paso finalizado" que se dispara cuando el usuario responsable del paso actual de la cadena hace click en el enlace correspondiente en la tarea. Este enlace lanza la acción de un controlador que lanza el evento y las acciones automatizadas configuradas en el proyecto harán que la tarea se mueva a la columna correcta y el usuario asignado cambie al responsable del siguiente paso.

Para que el plugin no rompa el resto de funciones de Kanboard lo que hace es ocultar la posibilidad de asignar manualmente una tarea a un usuario concreto. En vez de eso se tienen que asignar usuarios a cada una de las fases por las que tiene que pasar la tarea y es el plugin el que cambia el usuario asignado actualmente a la tarea. Esto hace el sistema de notificaciones y de acciones automáticas funcione sin problemas.

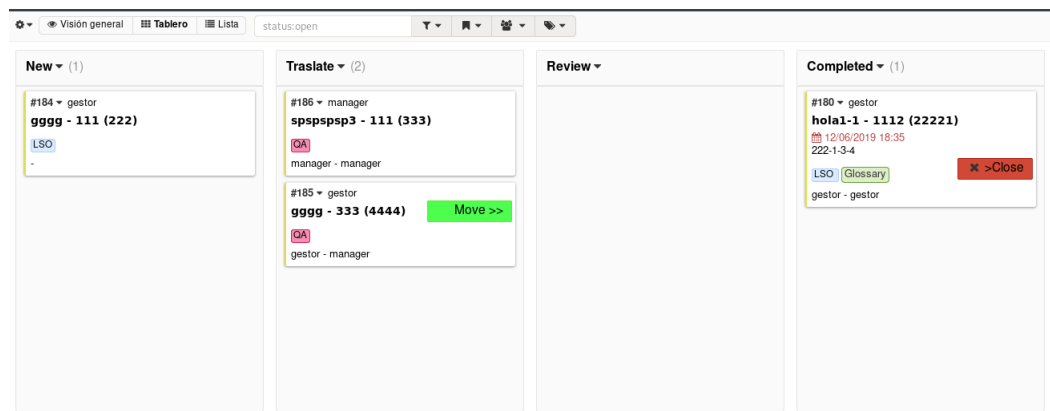


Figura 7.4: Captura de pantalla del tablero del usuario 'gestor' en el que se ven los botones que permiten marcar la tarea como lista para pasar al siguiente paso. Cuando se le asigne un traductor y un revisor a la tarea 184 se moverá a la segunda columna automáticamente.

## 7.6. TitleComposer

En Kanboard, cada tarea debe tener obligatoriamente un título que será el que la identifique. En el caso del cliente, no querían establecer un título para cada tarea sino que quieren que este título se componga automáticamente a partir de los valores introducidos al crear la tarea. Concretamente el formato que piden que tenga el título es: "Subproducto - Project number (Package number)"

Además, el subproducto se debe seleccionar en un desplegable junto al producto y al cliente que pide la tarea, por lo que he implementado una interfaz en la que se pueda generar esta jerarquía Cliente;Producto;Subproducto a la que se accede desde las opciones de la aplicación. Luego, en el formulario de creación de tareas los desplegables de producto y subproducto se actualizan mediante peticiones ajax.

Este plugin se ocupa de coger la información de la nueva tarea y componer el título que tendrá.

Client	Product	Subproduct
cliente1 ▾	Producto1 ▾	Subproducto1 ▾
Project number	Package number	Extra number
1234	789	

Figura 7.5: Desplegables de selección de clientes, productos y subproductos dependientes entre ellos. Con esa selección, el título final de la tarea será: 'Subproducto1 - 1234 (789)'

**Configurar clientes**

Nuevo cliente

Create

Client Name	Acción
✕ c1	Editar productos
✕ c2	Editar productos

Figura 7.6: Interfaz de inserción y edición de clientes y productos.

## 7.7. Core

Aunque en principio se planeaba la extensión de kanboard sin modificar el núcleo, durante la implementación he encontrado cosas que se podían mejorar del programa principal para hacer más sencillo el desarrollo de nuevos plugins. Esto incluye desde errores de traducción hasta hacer más flexibles las funciones de renderizado de formularios. Estos cambios no son relativos a la inclusión de funciones nuevas sino a la mejora de las existentes. Se ha solicitado la inclusión de estas mejoras en siguientes versiones de la versión oficial de kanboard.

## 7.8. Configuración

Gracias a las funcionalidades que trae Kanboard, algunos de los requisitos se pueden cumplir sin necesidad de programar componentes nuevos sino únicamente configurando correctamente las opciones del proyecto.

En esta configuración se definen que columnas componen el tablero, las restricciones de los distintos roles, que rol tendrá cada usuario, las etiquetas y las categorías a las que puede pertenecer una tarea.

## Capítulo 8

# Conclusiones y trabajo futuro

Mientras realizaba el trabajo he llegado a la conclusión de que la mayor parte del esfuerzo a la hora de desarrollar una aplicación se emplea en cosas que ya están resueltas y puestas a nuestro alcance en forma de software libre. Si en vez de utilizar herramientas ya probadas y refinadas hubiese desarrollado todo desde cero, tendría que haber dedicado mucho mas tiempo para llegar al mismo resultado.

Esto me ha hecho apreciar mas aun el software libre, pues nos permite centrarnos en las necesidades del proyecto en vez de reinventar la rueda una y otra vez.

El código del proyecto se puede consultar en el siguiente repositorio de Github: <https://github.com/ncortex/kanboarmd>. En ese repositorio se encuentran también los archivos Dockerfile y docker-compose.yml que permiten la creación de contenedores para ejecutar la aplicación.

Como trabajo futuro, quizás se podría plantear una integración de los sistemas de traducción para que el propio proceso de traducción se realice dentro de la plataforma. De esta forma se podría tener una interfaz completamente personalizada con las funciones de ayuda que necesiten los empleados (sugerencias de traducción, diccionario,...)



## Capítulo 9

# Glosario y acrónimos

- Licencia MIT: es una licencia de software que fue creada en el Instituto Tecnológico de Massachusetts y es posible usarla tanto para licenciar software libre como software no libre (Es una licencia sin copyleft, lo que permite que la creación de trabajos derivados a partir de ella pudieran ser no libres). Se encuentra entre las licencias compatibles con GNU-GPL y es parecida a la Licencia BSD.
- Kanban: es un método para gestionar el trabajo intelectual (como el desarrollo de software o la traducción de textos) con énfasis en la entrega justo a tiempo, y sin sobrecargar a los miembros del equipo. En este enfoque, todo el proceso desde la definición de una tarea hasta su entrega al cliente, se muestra para que los miembros del equipo lo vean y tomen el trabajo de una cola.
- Hooking: el término hooking abarca una gama de técnicas utilizadas para alterar o aumentar el comportamiento de un programa o aplicación interceptando mensajes o eventos pasados entre sus componentes. El código que se ejecuta al interceptar estos mensajes se denomina hook.
- Webhook: Un webhook es un método de alteración del funcionamiento de una página o aplicación web con callbacks personalizados. El término viene de hook.





# Bibliografía

- Documentación oficial de Kanboard: <https://docs.kanboard.org>
- Documentación oficial de Docker: <https://docs.docker.com/>
- <https://kanbantool.com/es/metodologia-kanban>
- <https://www.w3schools.com>
- <https://github.com/creecros/MetaMagik>