

Proyecto de final de carrera de Ingeniería Informática Superior

the **T I M E B I R D**



de  **atenea
tech**

Alumno: César Díez Sánchez

cesar@ateneatech.com

Versión: 1.0

Director: Siddharta Navarro Castellar

Secretario (ponente): Jose Ramon Herrero Zaragoza

Presidente: Jordi García Almiñana

Vocal: Jordi Marco Gómez

DATOS DEL PROYECTO

Título del proyecto: *The Time Bird*

Autor: César Díez Sánchez

Titulación: Ingeniería Informática Superior

Créditos: 37,5

Director: Siddharta Navarro Castellar

Co-director: Luis Ortiz Ramos

Empresa: Atenea tech

MIEMBROS DEL TRIBUNAL

Presidente: Jordi García Almiñana

Vocal: Jordi Marco Gómez

Secretario: Jose Ramon Herrero Zaragoza

CALIFICACIÓN

Calificación numérica:

Calificación descriptiva:

Fecha:

Agradecimientos

En la última DrupalCamp celebrada en Barcelona en Febrero de 2010, tuve el placer de encontrar a dos emprendedores de la FIB que desarrollan webs con el gestor de contenidos Drupal. Posteriormente me ofrecieron un convenio de prácticas, para poder realizar *The Time Bird* como proyecto de final de carrera, una gran idea que consideré muy apropiado hacer realidad.

Gracias al soporte recibido por estos dos grandes emprendedores y a la vez desarrolladores, he podido diseñar y plasmar ***The Time Bird***, un producto que se espera que sea utilizado y mejorado en el tiempo.

Tanto a Luis Ortiz como a Siddharta Navarro les quiero agradecer también el buen ambiente de trabajado creado, ya que me ha permitido vivir buenas experiencias, una mayor colaboración y responsabilidad por la empresa, y la oportunidad tan fantástica de trabajar con proyectos y clientes reales, así como poder ver con más claridad, el día a día que vive un desarrollador web.

También quiero agradecer a mis padres y a Sílvia Bergadà, el apoyo incondicional recibido en todos los momentos duros, así como los más felices durante toda la carrera y la elaboración del proyecto. Gracias por estar ahí.

Índice de contenido

1.	Introducción.....	5
1.1.	Organización de la documentación.....	5
1.2.	Situación.....	6
1.3.	Motivación.....	7
1.4.	Alcance	8
1.5.	Objetivos	9
1.5.1.	Objetivos desglosados	9
1.6.	Metodología de desarrollo.....	10
1.6.1.	Metodología escogida.....	13
1.7.	Productos similares.....	14
1.7.1.	<i>Tick</i>	15
1.7.2.	<i>Timr</i>	16
1.7.3.	<i>Task Coach</i>	17
1.7.4.	<i>Officetime</i>	19
1.7.5.	<i>FreshBooks</i>	20
1.7.6.	<i>Toggl</i>	22
1.7.7.	<i>Klok</i>	24
1.7.8.	<i>Cashboard</i>	25
1.7.9.	<i>Harvest</i>	26
1.7.10.	<i>Paymo</i>	28
1.7.11.	<i>Freckle</i>	29
1.7.12.	<i>The Time Bird</i>	31
2.	Análisis inicial.....	33
2.1.	Análisis de Requerimientos	33
2.1.1.	Requerimientos funcionales	33
2.1.2.	Requerimientos no funcionales.....	34
2.2.	Planificación	35

2.3.	Análisis económico	37
2.4.	Análisis tecnológico	40
2.4.1.	Plataforma móvil	40
2.4.2.	Plataforma web.....	47
3.	Especificación	55
3.1.	Casos de uso	55
3.1.1.	Aplicación móvil	56
3.1.2.	Aplicación web	59
3.2.	Modelo conceptual.....	65
3.2.1.	Aplicación móvil	65
3.2.2.	Aplicación Web	66
3.3.	Modelo del comportamiento.....	68
3.3.1.	Aplicación móvil	68
3.3.2.	Aplicación Web	72
4.	Diseño.....	79
4.1.	Arquitectura física	80
4.2.	Tecnologías utilizadas	81
4.2.1.	Drupal	81
4.2.2.	Android.....	87
4.2.3.	Comunicación entre la aplicación móvil y web.....	93
4.3.	Arquitectura lógica	95
4.3.1.	Interfaz gráfica	99
4.3.2.	Lógica de negocio.....	105
4.3.3.	Gestión de datos	110
5.	Implementación.....	117
5.1.	Configuración inicial de la aplicación web.....	117
5.1.1.	Creación de contenido	118
5.1.2.	Permisos de usuario	118
5.1.3.	Instalación y configuración del módulo <i>Services</i>	119

5.2.	Desarrollo librerías de comunicación	120
5.2.1.	Android.....	120
5.3.	Desarrollo de funcionalidades principales.....	123
5.3.1.	Inicio de sesión	123
5.3.2.	Listar etiquetas.....	124
5.3.3.	Crear etiqueta	126
5.3.4.	Activar etiqueta	128
5.3.5.	Desactivar etiqueta	130
5.3.6.	Deshabilitar etiqueta	132
5.3.7.	Proceso de análisis mediante Cron.....	134
5.3.8.	Estadísticas	136
5.4.	Diseño gráfico.....	145
5.4.1.	Aplicación móvil	145
5.4.2.	Aplicación web	146
		149
6.	Pruebas	150
6.1.	Aplicación móvil	150
6.1.1.	Inicio de sesión	151
6.1.2.	Cierre de sesión	151
6.1.3.	Creación de etiquetas	151
6.1.4.	Edición de etiquetas	151
6.1.5.	Activación y desactivación de etiquetas.....	151
6.1.6.	Deshabilitación de etiquetas	152
6.1.7.	Compatibilidad.....	152
6.2.	Aplicación web	152
6.2.1.	Estadísticas	152
6.2.2.	Sugerencias.....	152
6.2.3.	Interfaz.....	153
6.2.4.	Seguridad y permisos de usuario.....	153

6.2.5.	Compatibilidad.....	153
6.2.6.	Rendimiento	153
7.	Conclusiones, ampliaciones y mejoras	154
7.1.	Revisión de los objetivos	154
7.2.	Desviaciones de la planificación inicial y presupuesto	156
7.2.1.	Problemas encontrados.....	156
7.2.2.	Planificación final	157
7.2.3.	Presupuesto final	159
7.3.	Valoración	161
7.4.	Ampliaciones y mejoras	161
7.4.1.	Ampliaciones	162
7.4.2.	Mejoras.....	162
8.	Glosario.....	164
9.	Bibliografía.....	168
10.	Anexos	170
10.1.	Lista de elementos que aparecen en la memoria	170
10.2.	Instalación y configuración de un sitio web con Drupal	177
10.2.1.	Configuración del servidor e instalación de Drupal	177
10.2.2.	Configuración inicial de la aplicación web	181
10.3.	Instalación y configuración de un entorno Android.....	183
10.4.	Manual de usuario	185
10.4.1.	Aplicación móvil	185
10.4.2.	Aplicación web	192

1. Introducción

Este documento pretende materializar el trabajo realizado durante en el desarrollo del Proyecto de Final de Carrera ***The Time Bird***. Se ha realizado la documentación conforme el modelo más realista posible, así como la descripción adecuada de cada una de las fases por las que generalmente pasa un proyecto de estas características (desarrollo de software).

1.1. Organización de la documentación

La documentación se ha estructurado de la siguiente forma:

En el capítulo de introducción se presenta el proyecto, y se definen los conceptos importantes que envuelven su desarrollo, ya sea las principales motivaciones y objetivos, así como la metodología a seguir durante todo el periodo de desarrollo y una muestra de los productos existentes que se cree que puedan ser competencia directa para ***The Time Bird***.

Seguidamente se realiza un análisis inicial de todos los elementos importantes a tener en cuenta antes de empezar su desarrollo, incluyendo el análisis de requerimientos del producto, una planificación de tiempo detallada, así como un análisis sobre las alternativas tecnológicas para llevar a cabo su implementación. El capítulo finaliza con una descripción detallada del análisis económico realizado inicialmente.

A continuación encontramos la descripción de cada una de las fases más comunes en el desarrollo de *software*. Primero se presenta la especificación, incluyendo los casos de uso de cada una de las funcionalidades de las dos aplicaciones a desarrollar, así como su modelo conceptual y de comportamiento. Después encontramos toda la fase de diseño, dividida en la arquitectura física del sistema, las tecnologías escogidas para el desarrollo de las aplicaciones, y la arquitectura lógica del sistema. Finalmente, en el capítulo de implementación y en el de pruebas se documentan todas las funcionalidades desarrolladas durante el proyecto y las pruebas realizadas de cada una de las aplicaciones a desarrollar, y específicamente en la aplicación web, tanto en el servidor de desarrollo como en el servidor de producción.

Una vez se han descrito las etapas más comunes en el desarrollo de *software*, se da un repaso a la planificación realizada inicialmente analizando las posibles desviaciones y comprobando si los objetivos iniciales marcados, se han cumplido. También se expresan las sensaciones y experiencias vividas, así como mejoras de las aplicaciones, todo ello en el capítulo de conclusiones, ampliaciones y mejoras.

Finalmente, encontramos un glosario, donde se describen los acrónimos utilizados durante toda la documentación, bibliografía, y anexos, donde se encuentran pequeños documentos que permiten la ampliación y concretización de algunas partes mencionadas durante todo el documento, así como otra documentación diversa y un manual de usuario.

1.2. Situación

En la era en que vivimos, el rápido avance en el ámbito tecnológico ha permitido mejorar considerablemente las herramientas de trabajo en el ámbito laboral y, más concretamente en el ámbito de gestión de proyectos, ha habido una mejora sustancial en la forma en que se gestiona y distribuye uno de los recursos más importantes, y a su vez, pilar de todo proyecto: **el tiempo**.

Como muy bien se comenta en el artículo **Project Management & Time Tracking, ¿un matrimonio perfecto?** [1] cuando estamos inmersos en el desarrollo de un proyecto es muy importante conocer el tiempo que le estamos dedicando. Principalmente, el tiempo como recurso es muy valioso por muchos motivos; es muy escaso, no se puede comprar, no se puede almacenar, no se puede multiplicar, etc. Y de esta importancia es de la que deriva el uso de herramientas de control de tiempo (*time tracking*) que aporta 4 beneficios fundamentales tanto para el jefe proyecto, como para el resto de participantes:

Aprender del pasado. Saber cuánto tiempo se ha de invertir en desarrollar un proyecto, así como las horas extraordinarias que han sido necesarias para acabarlo, es información fundamental una vez finalizado el proyecto. Además, disponer del historial del tiempo requerido servirá para tomar futuras decisiones en proyectos similares.

Visión general de la situación actual o lo que viene a ser el seguimiento del proyecto. Tener un control del tiempo invertido en realizar ciertas tareas y ver si éste supera o no al previsto inicialmente, es primordial. Mediante el primer caso, podremos saber si tenemos que realizar nuevas planificaciones, contratar más recursos, delegar o externalizar. En el segundo caso nos permite comprobar si realmente se están realizando las tareas marcadas. Es importante monitorizar el proyecto para saber si nos estamos esforzando lo suficiente para alcanzar los objetivos.

Valoración económica del proyecto y de los recursos: Saber el coste total previsto para el desarrollo de proyecto es muy importante para medir su rentabilidad. El uso de un sistema *time tracking* nos permitirá tener un justificante de las horas extras realizadas y, si así estuviera establecido, se podrían compensar.

Mejorar la productividad: nos ayuda a saber en qué empleamos el tiempo, lo que puede influir en la motivación de los trabajadores, al ser conscientes del tiempo dedicado a tareas relevantes para el proyecto.

Como hemos podido observar, es realmente necesario este tipo de herramientas, ya que los beneficios aportados son sustanciales, y si se utilizan como es debido, pueden llegar a proporcionar mejoras para evitar pérdidas considerables de dinero a las empresas.

1.3. Motivación

En estos tiempos, ser autónomo o tener una pequeña empresa en el mundo de la informática no es nada fácil. Con la crisis, algunos clientes dejan facturas por pagar o incluso se niegan a pagar proyectos una vez están en sus manos. Este problema sumado a la difícil tarea de la gestión de proyectos, hace que sea realmente necesario tener una herramienta que permita obtener los beneficios mencionados en el apartado anterior.

Para la empresa **Atenea tech**, compañía de análisis, diseño y programación web con sede en Barcelona, y expertos en el gestor de contenidos **Drupal**, extraer información para obtener los beneficios anteriores era una tarea ardua, por no decir imposible. *Google Calendar*¹, *Open Atrium*² así como otras herramientas de gestión de proyectos, no permiten la obtención de ese tipo de información, y las soluciones existentes o son demasiado complicadas, o demasiado caras.

En base a ello nace **The Time Bird**, una herramienta de gestión del tiempo que se dedica a cubrir todas las necesidades mencionadas, haciendo hincapié en la necesidad de **aprender del pasado**, que será útil para la planificación, la definición del alcance y la fijación de precios, además de permitir llegar a acuerdos más sólidos y creíbles con los clientes, como en la **valoración económica del proyecto**, ya que será posible hacer facturas de acuerdo al esfuerzo realizado. Precisamente por eso, ha de ser una herramienta que facilite el acceso y la manutención de un registro de la información relacionada con todas las tareas que realiza la empresa.

Para hacer **The Time Bird** realidad, se necesita tener una herramienta fácil de utilizar y de gestionar, por eso se ha pensado que la solución ideal es el desarrollo de una aplicación móvil que transfiera datos a una aplicación con **Drupal** que contenga toda la

¹ *Google Calendar* es una agenda que permite tener diferentes calendarios diferenciados por colores. Permite compartir los calendarios privados con otros usuarios pudiendo darles diferentes niveles de permisos.

² *Open Atrium* es una intranet empaquetada que permite que diferentes equipos tengan su propio espacio de trabajo y conversaciones. Proporciona seis funcionalidades básicas: blog, wiki, calendarios, lista de tareas, micro-blog y un panel para gestionarlo todo.

lógica de negocio, y que permita al usuario tener a mano sus propias estadísticas sobre los datos previamente enviados.

1.4. Alcance

Una de las primeras tareas en todo proyecto, es la definición de su alcance, o dicho de otra forma, delimitar las funcionalidades a desarrollar (en el caso de un producto software) para poder cumplir con todos sus objetivos, al finalizar el proyecto.

Se podría decir que un proyecto en el que no se defina su alcance de forma correcta, tendría muchas posibilidades de ser un proyecto condenado a grandes problemas en su planificación, ejecución y control, por lo que su probabilidad de éxito se reduciría considerablemente. En el caso de **The Time Bird**, es muy importante definir su alcance de forma correcta, ya que una ampliación considerable de sus funcionalidades, podría hacerlo un producto no deseado, y ha de ser una herramienta muy fácil de usar.

Se pretende que los clientes puedan gestionar su tiempo y puedan ver una serie de estadísticas que relacionen ciertos usuarios, ciertas etiquetas y el tiempo invertido en éstas. Para ello, se pretende desarrollar una aplicación web instalada en un servidor con acceso garantizado mediante el dominio thetimebird.com, que permitirá al usuario y a todos los usuarios a los que se le haya dado permiso para ello, una serie de estadísticas sobre las tareas realizadas mediante un cliente externo. Cabe tener en cuenta que el servidor se podría llegar a sobrecargar si hubiera una cantidad elevada de peticiones y envíos de información, aunque el plan inicial de marketing para **The Time Bird** evite este problema, ya que se pretende vender instalaciones de las aplicaciones web a cada uno de los clientes, ya sean autónomos o pequeños grupos de trabajo. Aun así, se realizarán pruebas de carga correspondientes en el servidor real, para saber cómo reacciona ante una cierta cantidad de peticiones instantáneas.

Por otro lado se desarrollaría una aplicación móvil que permita gestionar (crear, editar, activar, desactivar y deshabilitar) etiquetas, que representen diferentes conceptos (procesos, proyectos, clientes, etc.) para nuestros clientes (garantizando que solo él tiene acceso a sus etiquetas), y así poder tener un control del tiempo dedicado a cada una de ellas.

Por último, será necesaria la elaboración de unas librerías que permitan la comunicación de datos entre la aplicación móvil y la aplicación web. Todos los datos enviados por el cliente externo, serán transformados en la aplicación web para ser mostrados en forma de gráficos con estadísticas diversas.

1.5. Objetivos

El objetivo principal de **The Time Bird** es **facilitar el registro y la explotación** de la información del tiempo al que los usuarios dedican a ciertos procesos, proyectos, clientes u otros conceptos, materializados en forma de etiquetas. Para ello, se necesitan alcanzar los siguientes 4 grandes objetivos:

- Desarrollar una aplicación móvil.
- Desarrollar una aplicación web.
- Crear un sistema que permita la comunicación entre ambos.
- Que el *software* desarrollado sea Código Libre mediante la Licencia GNU *GPL* [2].

1.5.1. Objetivos desglosados

- De la **aplicación móvil**, con Sistema Operativo **Android 2.3**:
 - Inicio de sesión para cada uno de nuestros clientes, lo que permitirá gestión de etiquetas personalizada.
 - Pantalla de gestión de diferentes conceptos como proyectos, procesos y clientes, materializados en etiquetas.
 - Opciones secundarias y personalización de la aplicación.
 - Desarrollo de librerías en *Java*³ necesarias para proveer a la aplicación móvil de los métodos necesarios para la transacción de datos entre las aplicaciones.
 - Compatibilidad con todas las versiones posibles del Sistema Operativo (incluyendo como mínimo las más utilizadas; versión 2.2 (*Froyo*) y versión 2.3 (*Gingerbread*)).
- De la **aplicación web**, con **Drupal**:
 - Creación del contenido necesario para almacenar los datos de los usuarios.
 - Instalación y configuración de los módulos necesarios que harán posible la gestión de ciertos tipos de contenido.
 - Configuración del módulo *Services* que hará posible el desarrollo de librerías para el envío y recepción de datos para permitir la sincronización de datos con el móvil.
 - Desarrollo de librerías en *PHP* necesarias para proveer a la web de los métodos necesarios para la transacción de datos entre aplicaciones.

³ Java es un lenguaje de programación orientado a objetos, desarrollado por *Sun Microsystems* a principios de los años 90.

- Automatización en *background* del proceso de análisis de las tareas realizadas hasta el momento (proceso a realizar cada 5 minutos).
- Gestión de los diferentes tipos de estadísticas de los datos enviados por el usuario.
- Aplicaciones **simples, bonitas, usables y útiles**.
- Análisis del potencial del SO **Android** y el CMS **Drupal**.
- Ambas aplicaciones tendrán asociada una licencia libre *GPL*.
- Realizar toda la documentación necesaria para que nuestros clientes puedan entender el funcionamiento de la aplicación perfectamente.

1.6. Metodología de desarrollo

Una vez vistos los objetivos que hay que alcanzar, es necesario desarrollar un método de trabajo que nos permita llevar a cabo todo lo planteado.

Existe una cantidad considerable de modelos de desarrollo *software*, por lo que solo se estudiarán algunos de los más conocidos en ingeniería de *software*.

Solo se realizará una breve explicación de cada uno de ellos, pero no se hará un análisis extenso ya que no se ha creído necesario.

Las metodologías se han dividido en dos vertientes.

Metodologías genéricas [3]

Las siguientes metodologías son consideradas de enfoque más genérico en el desarrollo de proyectos *software*. Todas ellas pueden clasificarse en *frameworks*⁴ lineales y/o iterativos.

- **Modelo en cascada** (lineal): Proceso secuencial de desarrollo de las fases genéricas de desarrollo de *software* (análisis de requerimientos, especificación, diseño, implementación, pruebas y mantenimiento), que permite cierta superposición y flujo entre fases. Hace hincapié en la utilización de documentación escrita para llevar un control riguroso del proyecto.

Lo bueno de este modelo es que el tiempo invertido en la especificación y diseño de

⁴ En el desarrollo de *software*, un marco de trabajo o *framework* en inglés, es una estructura conceptual y tecnológica de soporte, definida normalmente con artefactos o módulos *software* concreto. Puede incluir soporte de programas, bibliotecas, y un lenguaje interpretado entre otros programas para ayudar a desarrollar y unir los diferentes componentes de un proyecto.

software dará más posibilidades al proyecto para alcanzar sus objetivos, así como evitará posibles desvíos de planificaciones realizadas. Además al no haber paralelismo, el cliente especifica como quiere el producto.

Lo malo es su estrecha rigidez, adecuada en casos de desarrollos estables, sin cambios en requerimientos, y desvíos de planificación fácilmente gestionables. Además si al cliente no le gusta el producto final, probablemente se tendría que volver a comenzar todo el proceso, por lo que es una metodología lenta.

- **Prototipado** (iterativo): Es un *framework* de actividades dedicado al desarrollo de *software* prototipo, que debe ser construido en poco tiempo y con el uso de pocos recursos, y es una vez aprobado, cuando se puede iniciar el verdadero desarrollo de *software*. Se centra en las necesidades del cliente lo que le permite al desarrollador entender mejor lo que se ha de hacer.

Este modelo es útil cuando el cliente conoce los objetivos generales para el *software*, pero no identifica sus requerimientos de entrada o salida. También ofrece un mejor enfoque cuando los responsables del desarrollo no están seguros de la eficacia de ciertos procesos de su desarrollo.

Lo malo es el tiempo requerido en un proyecto de estas características, y que la rápida implementación de sistemas para que el cliente pueda ver ciertas funcionalidades, puede hacer que se desatiendan otros aspectos importantes como la calidad y el mantenimiento del proyecto a largo plazo.

- **Incremental** (lineal e iterativo): Esta metodología provee de una estrategia para controlar la complejidad y los riesgos en el desarrollo de un producto, ya que se van desarrollando ciertas partes que después son revisadas. De esta forma se va produciendo el *software* de forma incremental. Para ello se definen los requerimientos antes de proceder con lo evolutivo y se realiza una mini-cascada de desarrollo de cada uno de los incrementos del sistema.

Modelo útil, al ser un desarrollo incremental donde se va aprendiendo del desarrollo de partes anteriores.

- **Espiral** (lineal e iterativo): Las tareas a realizar según esta metodología, conforman una espiral en la que cada bucle o iteración representa un conjunto de actividades no fijadas a priori, sino que van eligiendo en función del análisis de riesgo. Cada viaje atraviesa cuatro cuadrantes básicos: (1) determinar objetivos, alternativas y desencadenantes de la iteración, (2) evaluar alternativas, identificar y resolver riesgos, (3) desarrollar y verificar resultados de la iteración y (4) plan de la próxima iteración.

Lo positivo del modelo, es que se centra en la evaluación y reducción del riesgo del proyecto dividiéndolo en segmentos más pequeños y proporcionando más facilidad de cambio durante el desarrollo.

Metodologías ágiles [4]

La alternativa a los métodos genéricos, que suelen ser menos adaptativos, pesados y laboriosos, es el desarrollo ágil de *software*, un concepto de ingeniería de *software* que promueve iteraciones en el desarrollo a lo largo del ciclo de vida del proyecto. La mayoría de metodologías ágiles minimizan riesgos, mediante el desarrollo en cortos lapsos de tiempo de ciertas funcionalidades. Cada una de las iteraciones incluye el ciclo de vida genérico de un proyecto (análisis de requerimientos, especificación...) y al final de cada una de ellas se han de evaluar las prioridades del proyecto.

Estos métodos enfatizan las comunicaciones entre individuos en vez de la documentación. La mayoría de equipos ágiles mantienen una colaboración con sus clientes de tal forma que prevalezca sobre la negociación de contratos, además de enfatizar el software funcional como primera medida del progreso. Es por lo que este tipo de métodos son criticados como "indisciplinados" por la falta de documentación técnica.

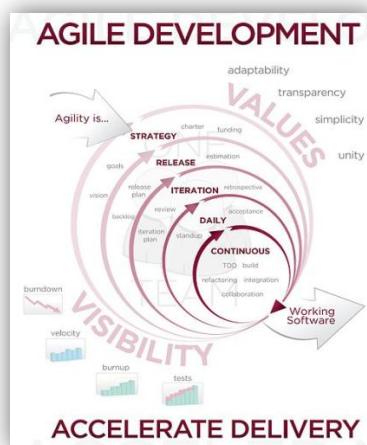


Ilustración 1: Definición gráfica para los métodos de desarrollo ágiles

Algunas de las metodologías ágiles más conocidas son *Scrum*⁵, *Programación Extrema*⁶,

⁵ *Scrum* es un modelo de referencia que define un conjunto de prácticas y roles, y que puede tomarse como punto de partida para definir el proceso de desarrollo que se ejecutará durante un proyecto.

⁶ Enfoque de ingeniería de *software*. Se diferencia de las metodologías tradicionales en que pone más énfasis en la adaptabilidad que en la previsibilidad.

Kanban⁷, Test-Driven Design⁸ y Open Unified Process⁹, la mayoría de ellos basados en principios similares en torno al proceso iterativo e incremental.

1.6.1. Metodología escogida

Aunque, como acabamos de ver, las metodologías genéricas son métodos pesados, laboriosos y rígidos, se podrían adaptar perfectamente a ***The Time Bird***, ya que los requerimientos no son cambiantes. Tanto la definición de proyecto, como su alcance, han sido específicos, por lo que puede haber pequeñas desviaciones, que puedan causar alguna iteración, pero no será la norma.

Por otra parte, también se podrían utilizar metodologías ágiles, debido a que **Drupal**, como CMS y gracias a su extensibilidad y modularidad, permite adaptar su desarrollo en cortos periodos de tiempo, en los que se van añadiendo funcionalidades junto a su posterior diseño.

La metodología seguida, aunque nunca de forma rigurosa, ha sido la de **modelo en cascada** con algunos tintes de **metodologías ágiles** en la parte de diseño.

La realización de una documentación inicial definiendo el proyecto y su alcance, el análisis de requerimientos, así como la especificación de funcionalidades, diseño, implementación y pruebas, se ha realizado acorde al modelo en cascada.

En cuanto a la fase de mantenimiento, se podría decir que hay expectativas puestas en el futuro de ***The Time Bird***, por lo que habrá un mantenimiento del producto durante un tiempo indeterminado.

Debido a pequeñas desviaciones en la toma de decisiones sobre el diseño de las aplicaciones (como por ejemplo las librerías a utilizar para mostrar los gráficos, o la metodología de comunicación entre aplicaciones), se han aplicado metodologías ágiles, ya que se tuvo que realizar una iteración de alguna funcionalidad completamente nueva debido a motivos de diseño que se podrán estudiar en el capítulo de conclusiones, al ser desviaciones sobre la planificación que no se habían previsto.

⁷ Sistema de información que controla de modo armónico la fabricación de los productos necesarios en la cantidad y tiempo necesarios en cada uno de los procesos.

⁸ Práctica de programación que involucra otras dos prácticas. En primer lugar, se describe una prueba y se verifica que las pruebas fallen, luego se implementa el código que haga que la prueba pase satisfactoriamente y seguidamente se refactoriza el código escrito.

⁹ Es un proceso mínimo y suficiente, lo que significa que solo se incluye el material fundamental y necesario. Por lo tanto, no provee lineamientos para todos los elementos que se manejan en un proceso, pero tiene los componentes básicos que pueden servir de base a procesos específicos.

A continuación podemos ver una descripción gráfica del modelo en cascada utilizado.

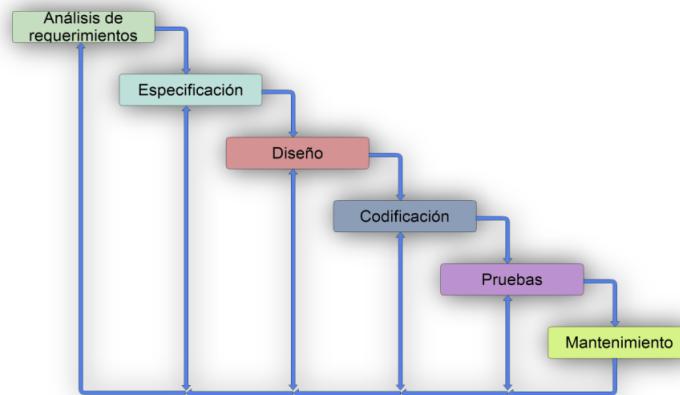


Ilustración 2: Descripción gráfica del modelo en cascada

1.7. Productos similares

A día de hoy, el mercado de las herramientas de gestión de tiempo es muy amplio. Existe una cantidad inmensa de aplicaciones, algunas de ellas muy competentes y maduras, algunas otras difíciles de usar, poco útiles o con interfaces poco amigables. También hay un amplio mercado de herramientas que no son exclusivamente para la gestión de tiempo, ya que lo incorporan como un añadido a la gestión de proyectos en general, por lo que herramientas de este tipo (algunas tan potentes como *Microsoft Project*) no se van a analizar ya que sus objetivos distan de los de ***The Time Bird***.

A continuación se va a describir un análisis específico de algunas de estas herramientas de gestión de tiempo, todas ellas seleccionadas por ser de las más utilizadas del mercado. Se van a analizar las características que puedan hacer de estas herramientas, productos altamente, medianamente o poco competentes en comparación con nuestro producto. También cabe remarcar que una gran cantidad de herramientas han sido descartadas para el análisis, al disponer única y exclusivamente de un cliente móvil, algo que al igual que las herramientas de gestión de proyectos, distan de los objetivos de ***The Time Bird***, que permite contabilizar tiempo rápidamente mediante la aplicación móvil, y permite el estudio del tiempo contabilizado a ciertas etiquetas mediante estadísticas, en la aplicación web.

Finalmente también se realizará un análisis de nuestro producto, mencionando sus puntos débiles, fuertes, y por qué se cree que es una alternativa muy interesante.

Cabe mencionar que cada una de los productos analizados tendrá una clasificación final, dependiendo del grado de competencia respecto a ***The Time Bird***. Las características

analizadas son comunes en todas las aplicaciones, de tal forma que se pueda realizar una comparación genérica, ya que no es nuestro objetivo el análisis detallado de cada una de ellas.

1.7.1. *Tick*

Solución de características similares a *Google Calendar*.

Parte del concepto básico de creación de proyectos, con tareas y clientes asociados. Se pueden crear cada uno de ellos, y asociar manualmente el tiempo dedicado a ciertas tareas pudiendo ver el resumen de tareas realizadas en el calendario, por lo que es un gestor de tiempo bastante simple.

Características destacadas disponibles en *tick*:

Multi-idioma	No
Diseño	No es un diseño llamativo ni muy atractivo, pero si es amigable
Plataforma/s	Web
Uso de gráficos	No
Ayuda online	Sí
Multi-tarea	No
Facilidad de uso	Medianamente fácil.

Tabla 1: Características destacadas, disponibles en *Tick*

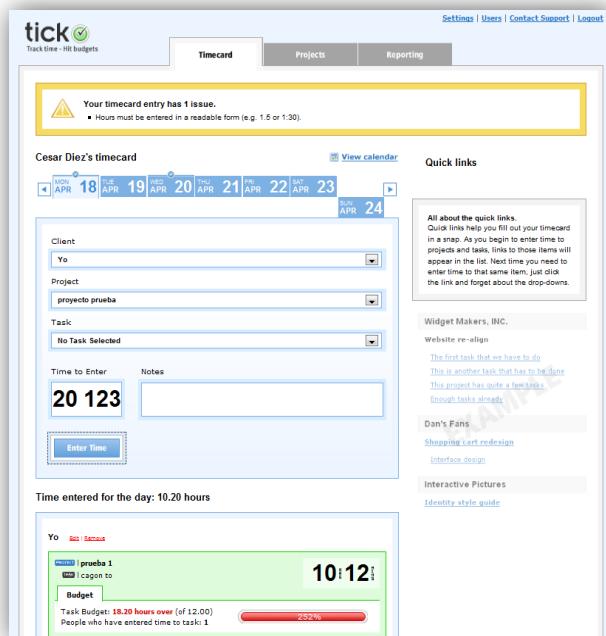


Ilustración 3: Visualización de ejemplo de *tick*

Otras características de menor importancia:

Elaboración de facturas	Sí
Generación de informes exportables	Sí
Integración con otras herramientas	<i>Freshbooks</i> (solo de pago)
Geolocalización	No

Tabla 2: Otras características de *tick*

Por otro lado, *tick* carece de algunas funcionalidades importantes, como la posibilidad de contabilizar etiquetas de forma automática, y otras plataformas desde la que poder enviar datos. Además, no ofrece gráficos estadísticos ni la posibilidad de crear múltiples proyectos a no ser que se de alta en su versión de pago, llegando a costar hasta \$79 al mes.

Conclusión:  No parece un producto que sea competencia directa con ***The Time Bird***.

1.7.2. *Timr*

Otro gestor de tiempo sencillo, con la característica destacada de ofrecer una gran ayuda *online*, además de ser el único gestor de tiempo encontrado que permite la localización por GPS de las tareas contabilizadas. Otra buena característica de este producto, es que dispone de clientes en la mayoría de plataformas de importancia del mercado actual.

Como el anterior, también parte del concepto básico de creación de proyectos, con tareas y clientes asociados.

Características destacadas disponibles en *timr*:

Multi-idioma	2 idiomas
Diseño	Tiene una interfaz muy trabajada, aunque es poco llamativa
Plataforma/s	Web, Blackberry, iOS, Android, Windows Mobile
Uso de gráficos	Limitado a barras de progreso
Ayuda online	Sí
Multi-tarea	No
Facilidad de uso	Medianamente fácil

Tabla 3: Características destacadas, disponibles en *timr*

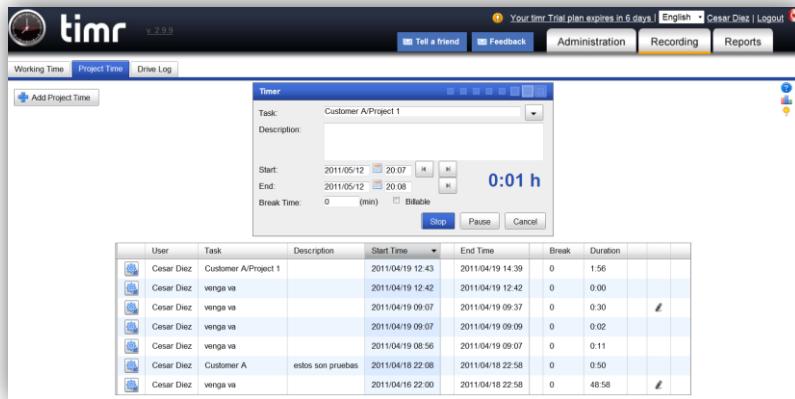


Ilustración 4: Visualización de ejemplo de *timr*

Otras características de menor importancia:

Elaboración de facturas	Sí
Generación de informes exportables	Sí
Integración con otras herramientas	Freshbooks (solo de pago)
Geolocalización	Sí

Tabla 4: Otras características de *timr*

En su contra encontraríamos problemas como la poca usabilidad en su versión gratuita, ya que ofrece la posibilidad de crear hasta 5 tareas por usuario registrado, dando en la versión de pago la posibilidad de que otras personas puedan acceder a nuestra cuenta, y tareas ilimitadas, todo por 39\$, por lo que hace que casi obligatoriamente se obtenga la versión de pago. Además la interfaz del contador de tiempo que dispone no está muy trabajada y no dispone de gráficos estadísticos trabajados.

Conclusión:  No parece un producto que sea competencia directa con ***The Time Bird***

1.7.3. *Task Coach*

Gestor de tiempo gratuito e integrado en diversas plataformas, que ofrece la posibilidad de contabilizar el tiempo para cada tarea, de forma rápida y sencilla, además, como característica propia, permite arrastrar las tareas entre carpetas, creando una jerarquización fácilmente manejable, además de ofrecer la posibilidad de cortar, copiar y pegar tareas, útil para tareas repetitivas en ciertos proyectos.

Task Coach, parte del concepto básico de creación de proyectos, con tareas y clientes

asociados.

Características destacadas disponibles en *Task Coach*:

Multi-idioma	Muchos idiomas
Diseño	Interfaz muy poco trabajada
Plataforma/s	Windows, Mac, Linux, iPhone
Uso de gráficos	No
Ayuda online	No
Multi-tarea	No
Facilidad de uso	Medianamente fácil

Tabla 5: Características destacadas, disponibles en *Task Coach*

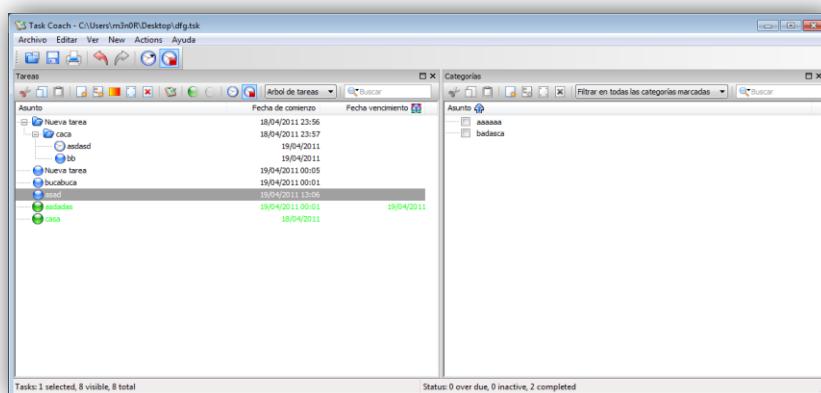


Ilustración 5: Visualización de ejemplo de *Task Coach*

Otras características de menor importancia:

Elaboración de facturas	No
Generación de informes exportables	Sí
Integración con otras herramientas	No
Geolocalización	No

Tabla 6: Otras características de *Time Coach*

Uno de sus principales defectos, es no disponer de un cliente web donde poder trabajar, así como clientes móviles, ya que, aunque se pueda instalar en diversas plataformas, no suele ser agradable necesitar tener el cliente instalado en cada ordenador donde lo vayamos a utilizar. Además dispone de una interfaz poco trabajada y sin ningún tipo de gráficos.

Conclusión: 😊 No es considerado un producto que sea competencia directa con **The Time Bird**.

1.7.4. *Officetime*

Sencilla pero funcional herramienta, que ofrece la posibilidad de realizar copias de seguridad del estado actual de las tareas.

Como todos los gestores de tiempo vistos hasta ahora, sigue el concepto básico de creación de proyectos, con tareas y clientes asociados. Además permite contabilizar el tiempo dedicado a las tareas de forma rápida y sencilla.

Características destacadas disponibles en *Officetime*:

Multi-idioma	No
Diseño	Interfaz no muy llamativa pero algo trabajada
Plataforma/s	Mac OS, Windows, Linux
Uso de gráficos	Sí
Ayuda online	Sí
Multi-tarea	No
Facilidad de uso	Fácil

Tabla 7: Características destacadas, disponibles en *Officetime*

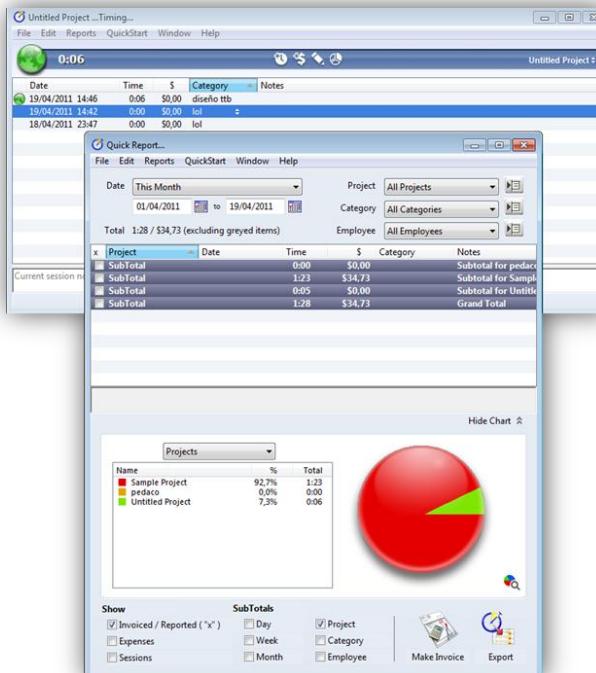


Ilustración 6: Visualización de ejemplo de *Officetime*

Otras características de menor importancia:

Elaboración de facturas	Sí
Generación de informes exportables	Sí
Integración con otras herramientas	iCalendar y Outlook
Geolocalización	No

Tabla 8: Otras características de *Officetime*

Por otro lado, *Officetime* no es gratuito. Sólo ofrece su versión de prueba durante cuatro meses. Su licencia cuesta 47\$ por usuario, que para las funcionalidades que actualmente ofrece, es un poco caro. Además, no dispone de clientes en plataformas móviles ni web, por lo que la facilidad de uso del programa, se ve de algún modo contrapuesta a la necesidad de tener la aplicación integrada en un sistema operativo. A parte de todo esto, y debido al hecho de que esté desarrollado en Java, hace que la interfaz no esté trabajada. Otro pésimo defecto del producto, es que una vez cerrado el programa, el contador no sigue contabilizando tiempo, por lo que es necesario tener abierto el programa durante su uso.

Conclusión: 😊 Aunque parece una herramienta con potencial, hay algunas características que hacen que no esté a la altura de los gestores de tiempo más actuales, por lo que no es considerado un producto que sea competencia directa con ***The Time Bird***.

1.7.5. FreshBooks

FreshBooks es, casi con toda seguridad, una de las soluciones más completas para la gestión de tiempo del mercado. De hecho, se podría decir que es una buena herramienta de gestión de proyectos incluyendo facturación, con una sección para la gestión de tiempo, integrada. Al disponer de una tienda solamente para añadir opciones al producto, tienen compatibilidad con otros buenos gestores de tiempo del mercado, como *tick*, *toggl* o *Timer Pro*, así como compatibilidad con diversos *CRMs*, plataformas de *marketing*, etc.

Cabe mencionar que esta herramienta, sigue el concepto básico de creación de proyectos, con tareas y clientes asociados

Características destacadas disponibles en *FreshBooks*:

Multi-idioma	No
Diseño	Interfaz amigable y diseño personalizado a gusto del cliente

Plataforma/s	Web, Windows Mobile, Android, iOs, Blackberry
Uso de gráficos	No
Ayuda online	Sí
Multi-tarea	No
Facilidad de uso	Media

Tabla 9: Características destacadas, disponibles en *tick*



Ilustración 7: Visualización de ejemplo de *FreshBooks*

Algunas otras características de menor importancia:

Elaboración de facturas	Sí
Generación de informes exportables	Sí
Integración con otras herramientas	Muchas, entre ellas, otros gestores de tiempo conocidos como <i>Toggl</i> , <i>Tick</i> y <i>Slife</i>
Geolocalización	No

Tabla 10: Otras características de *FreshBooks*

A pesar de ser una herramienta increíblemente potencial, tiene algunas carencias básicas para la gestión de tiempo que se pueden tapar con la integración de otras herramientas. Además un producto de estas características tendría que ser multilenguaje y solo está disponible en inglés.

Se podría decir que *Freshbooks* es una herramienta con un público objetivo diferente del que pretende llegar ***The Time Bird***, ya que se centra más en temas de relaciones con clientes (CRM), facturación y gestión de proyectos

El precio, no es ni su mejor ni su peor baza, dando la versión gratuita la posibilidad de trabajar con tan solo un cliente, hasta tener clientes ilimitados por 40\$ al mes.

Conclusión:  Herramienta completa para medianas y grandes empresas, por lo que no está dirigido al mismo público objetivo que **The Time Bird** y por ello no parece un producto que le haga competencia directa.

1.7.6. *Toggl*

Toggl es una de las herramientas, que por su facilidad de uso, interfaz amigable y trabajada, y obtención de gráficos estadísticos, se podría considerar una alternativa o competencia directa a **The Time Bird**.

Además, este producto no sigue el concepto básico de los anteriores, absteniéndose de clientes y centrándose en creación de proyectos asociados a tareas, lo que permite dejar claro que está más enfocada a la gestión de tiempo que a facturación.

Algunas de sus virtudes, es el número de idiomas configurables (más de 20 incluyendo el catalán), diversidad de gráficos estadísticos, y clientes en plataforma web y móviles, además de ser muy fácil de usar.

Características destacadas disponibles en *Toggl*:

Multi-idioma	20 idiomas, incluido el catalán
Diseño	Interfaz amigable y bonito diseño
Plataforma/s	Web, Android, iOs
Uso de gráficos	Sí
Ayuda online	Sí
Multi-tarea	De forma limitada
Facilidad de uso	Muy fácil

Tabla 11: Características destacadas, disponibles en *toggl*



Ilustración 8: Visualización de ejemplo (1) de *toggl*

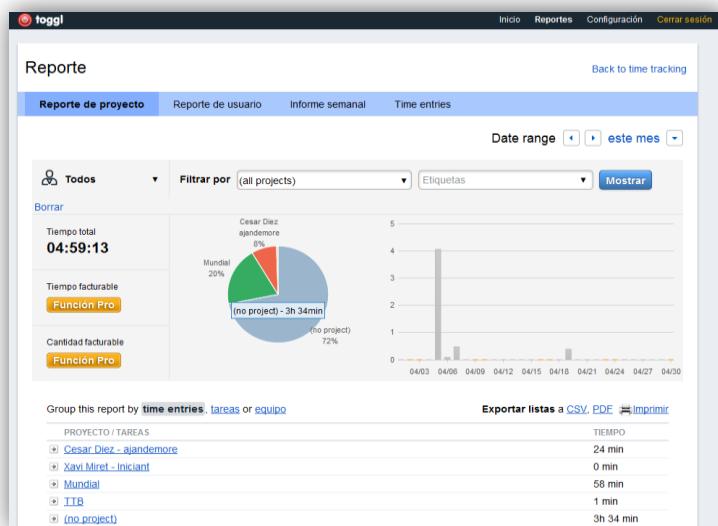


Ilustración 9: Visualización de ejemplo (2) de *toggl*

Otras características de menor importancia:

Elaboración de facturas	Sí
Generación de informes exportables	Sí
Integración con otras herramientas	Basecamp, RSS, iCalendar
Geolocalización	No

Tabla 12: Otras características de *toggl*

Aunque tenga algunos defectos, como el no poder cambiar el correo de usuario, o una interfaz, o que la versión de pago sea de hasta \$79 mensuales dando soporte a 40 usuarios, cabe destacar que es una buena solución para autónomos y pequeñas empresas. Es funcional y fácil de usar.

Otro aspecto que no se ha mencionado en ningún otro gestor, es el concepto de multitarea, o lo que es lo mismo, la posibilidad de tener más de una etiqueta contabilizándose al mismo tiempo. En esta herramienta encontramos la posibilidad de crear tareas asociadas a varias etiquetas, pero estas no podrán variar durante la duración de la tarea. El concepto de **The Time Bird** es diferente, aunque podría ser perfectamente una alternativa.

Conclusión: 🧐 Se aproxima al concepto de multitarea de **The Time Bird**, además de ofrecer buenas opciones al usuario, por lo que este producto se considera competencia directa con **The Time Bird**.

1.7.7. *Klok*

Sencilla pero eficaz herramienta, que permite sincronizar las tareas creadas con *Google Calendar*, lo que lo convierte en una herramienta interesante. También dispone de una licencia a un precio aceptable, ya que por 12\$ podremos comprar el programa, aunque si tuvieran que utilizar el producto muchos usuarios, al tener que pagar por cada uno de ellos, se podría convertir en un programa caro para los servicios que ofrece.

Como muchos otros, parte del concepto básico de creación de proyectos, con tareas y clientes asociados. Además las tareas se pueden jerarquizar y arrastrarse entre proyectos, por lo que en este aspecto es muy similar a *Task Coach*.

Características destacadas disponibles en *Klok*:

Multi-idioma	No
Diseño	Interfaz trabajada
Plataforma/s	Adobe air (herramienta de Adobe disponible para todas las plataformas)
Uso de gráficos	Sí, de pago
Ayuda online	No
Multi-tarea	No
Facilidad de uso	Fácil

Tabla 13: Características destacadas, disponibles en *Klok*

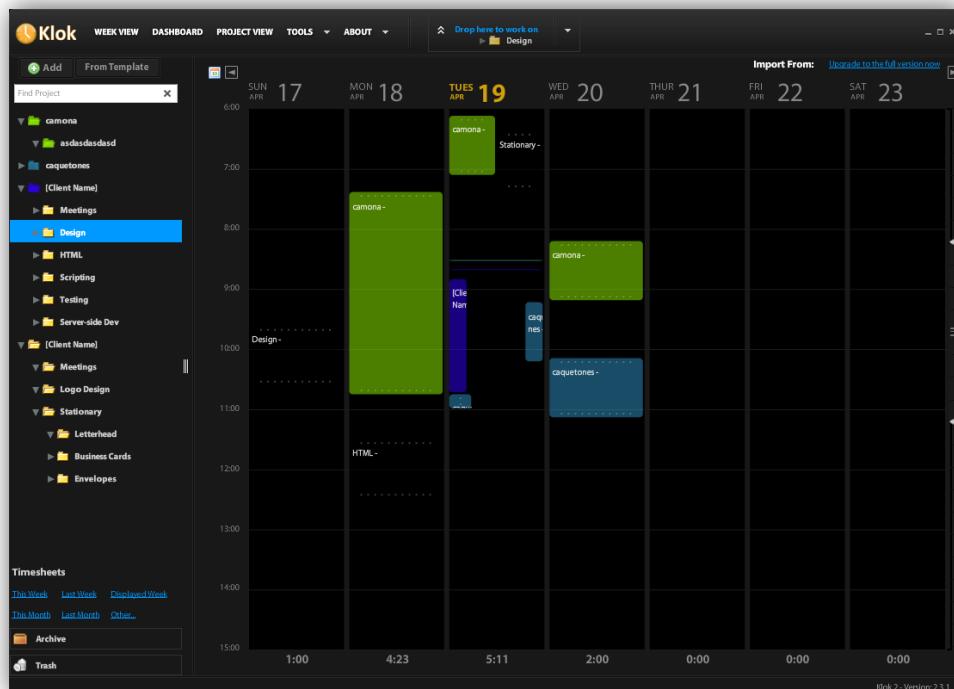


Ilustración 10: Visualización de ejemplo de *Klok*

Otras características de menor importancia:

Elaboración de facturas	Sí
Generación de informes exportables	Sí
Integración con otras herramientas	FreshBooks, Harvest, Basecamp y Paymo
Geolocalización	No

Tabla 14: Otras características de *Klok*

Por otro lado, cabe mencionar que todas las funcionalidades interesantes son de pago. Además es necesario tener *Adobe Air* instalado para su funcionamiento. Otro de sus mayores defectos, es no disponer de clientes en plataformas móviles ni web, por lo que lo acaba convirtiendo en otra de las decenas de gestores que no serían útiles al no tener una pantalla delante.

Conclusión:  No parece un producto que sea competencia directa con ***The Time Bird***

1.7.8. *Cashboard*

Herramienta de características similares a *toggl*, aunque no ofrece una interfaz tan amigable, ni su gran facilidad de uso.

También parte del concepto básico de creación de proyectos, con tareas y clientes asociados.

Características destacadas disponibles en *Cashboard*:

Multi-idioma	No
Diseño	Interfaz algo trabajada. Ofrece personalización al cliente
Plataforma/s	Web
Uso de gráficos	Sí
Ayuda online	Sí
Multi-tarea	No
Facilidad de uso	Media

Tabla 15: Características destacadas, disponibles en *Cashboard*

The screenshot shows the Cashboard application's interface. At the top, there is a navigation bar with links for Home, Contacts, Estimates, Projects, Timesheet (which is highlighted in blue), Income, and Expenses. There are also buttons for Settings, Help, Logout, and a user named Cesar Diez. Below the navigation bar, there are tabs for Daily and Weekly views, with 'Daily' selected. The main area is titled 'Daily Timesheet' and contains instructions about tracking time for projects. It shows a table for Cesar Diez on April 19, 2013, with columns for Project / Task, Hours, Notes, and Billable. A dropdown menu 'Select a task' is open, showing several entries like 'Carlos cifuentes - bacaaaaaa' with a duration of 0:05. On the right side, there is a sidebar titled 'Time Entry' with sections for 'Time can be entered in two formats' (Hours:Minutes and Hours.Percent of Minutes), and 'Time display' which states that time will always be shown in Hours:Minutes format.

Ilustración 11: Visualización de ejemplo de *Cashboard*

Algunas otras características:

Elaboración de facturas	Sí
Generación de informes exportables	Sí
Integración con otras herramientas	Basecamp
Geolocalización	No

Tabla 16: Otras características de *Cashboard*

Al ser un producto de características similares a *toggly* pero con algunas carencias como el poco soporte con las plataformas móviles, la interfaz no está a su altura y tampoco está disponible en multi-idioma, por lo que hace a *toggly* una mejor opción.

Conclusión: No parece un producto que sea competencia directa con ***The Time Bird***.

1.7.9. *Harvest*

Herramienta de características muy similares a *Cashboard*, sobre todo por su interfaz. Además tiene alguna mejora respecto a éste último, ya que dispone de clientes en

plataformas móviles.

También parte del concepto básico de creación de proyectos, con tareas y clientes asociados.

Características consideradas de importancia:

Multi-idioma	No
Diseño	Interfaz amigable aunque no muy trabajada.
Plataforma/s	Web, Android, iOS
Uso de gráficos	Sí
Ayuda online	No
Multi-tarea	No
Facilidad de uso	Media

Tabla 17: Características destacadas, disponibles en *Harvest*

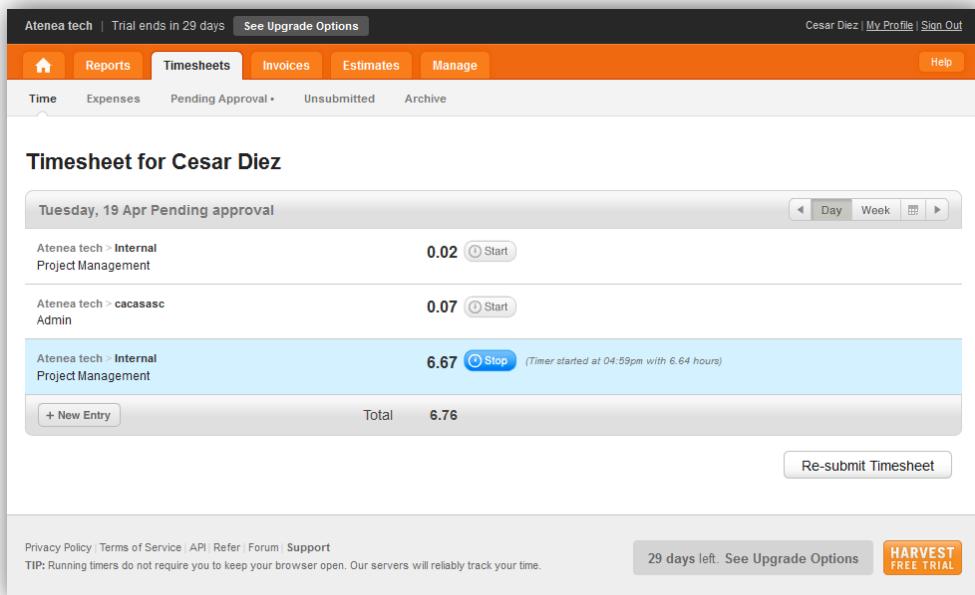


Ilustración 12: Visualización de ejemplo de *Harvest*

Otras características de menor importancia:

Elaboración de facturas	Sí
Generación de informes exportables	Sí
Integración con otras herramientas	Basecamp
Geolocalización	No

Tabla 18: Otras características de *Harvest*

Aunque sea posible que *Harvest* sea una mejor opción que *Cashboard*, todavía no llega al nivel de *toggl*, por lo que nos encontramos en la misma situación que antes. Además no es gratuito y puede llegar a ser muy caro, llegando a alcanzar los \$90 al mes para un equipo con 10 usuarios.

Conclusión:  No parece un producto que sea competencia directa con ***The Time Bird***.

1.7.10. Paymo

Paymo es, posiblemente, la herramienta analizada que ofrece más seguridad para sus usuarios, proporcionándoles una *API KEY* única y exclusivamente para cada registro realizado. De esta forma, se podría evitar que criminales informáticos pudieran crear nuevas tareas, o modificar el tiempo dedicado de las tareas así como modificación de otro tipo de datos.

Otra de las características que nos llama la atención de este gestor de tiempo es que ofrece un margen de error al usuario de un minuto en la creación de nuevas tareas, al igual que ***The Time Bird***. Algo que se considera básico por el mero hecho de no tener que ir borrando tareas con ciertas etiquetas seleccionadas por error.

Como otros muchos gestores de tiempo, también parte del concepto básico de creación de proyectos, con tareas y clientes asociados.

Características destacadas disponibles en *Paymo*:

Multi-idioma	Sí, aunque no incluye el español.
Diseño	Interfaz amigable aunque no muy trabajada.
Plataforma/s	Web, Android, Mac Os, Windows
Uso de gráficos	Sí
Ayuda online	Sí
Multi-tarea	No
Facilidad de uso	Media

Tabla 19: Características destacadas, disponibles en *Paymo*

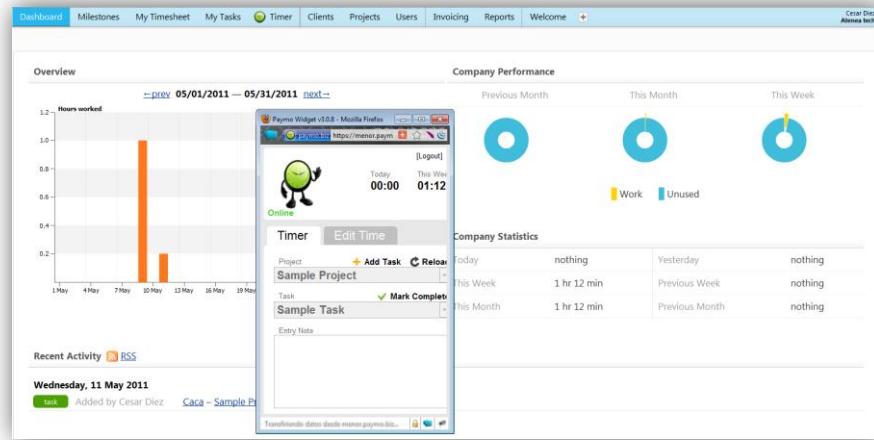


Ilustración 13: Visualización de ejemplo de *Paymo*

Otras características de menor importancia:

Elaboración de facturas	Sí
Generación de informes exportables	Sí
Integración con otras herramientas	iCalendar
Geolocalización	No

Tabla 20: Otras características de *Paymo*

Por otro lado, al igual que algunos otros gestores de tiempo, dispone de un contador con ventana externa, algo que hace pierda usabilidad. Otro defecto de este gestor, es que tampoco incluye el español.

Conclusión: 😕 Aunque tenga muchas semejanzas con otros gestores de tiempo analizados, tiene características similares a ***The Time Bird*** a excepción de su gran facilidad de uso, así como su concepto de multi-tarea. De todas formas, se podría considerar un producto de competencia directa con ***The Time Bird***.

1.7.11. Freckle

Potente, agradable y sencilla herramienta, que permite gestionar tareas creadas y usuarios de forma ágil. Es probablemente, la herramienta que más se aproxime al concepto de multitarea que ***The Time Bird*** quiere mostrar, ya que, como vemos en la ilustración 14, se pueden seleccionar tareas existentes mediante autocompletado, con la posibilidad también de asociar a las tareas una facturación, avisos a otros usuarios y otro

tipo de características mediante caracteres especiales. Es una muy buena opción por su facilidad de uso, así como las características que ofrece.

Freckle, a diferencia de otros productos, pretende unir el concepto de cliente y proyecto, como un solo elemento, asociado a las tareas que se realizan.

Características destacadas disponibles en *Freckle*.

Multi-idioma	No
Diseño	Interfaz bonita y trabajada
Plataforma/s	Web
Uso de gráficos	Sí
Ayuda online	Sí
Multi-tarea	Sí
Facilidad de uso	Muy fácil

Tabla 21: Características destacadas, disponibles en *Freckle*



Ilustración 14: Visualización de ejemplo (1) de *Freckle*

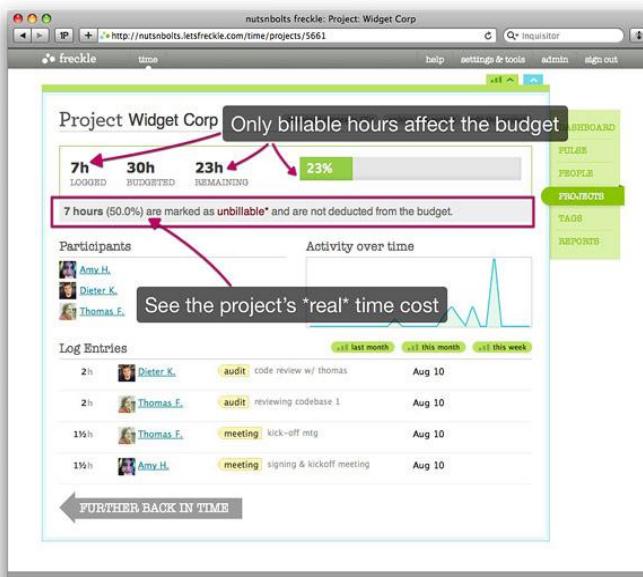


Ilustración 15: Visualización de ejemplo (2) de *Freckle*

Otras características de menor importancia:

Elaboración de facturas	Sí
Generación de informes exportables	Sí
Integración con otras herramientas	Freckel, Beanstalk
Geolocalización	No

Tabla 22: Otras características de *Freckle*

Cabe mencionar, que pese a lo buen gestor de tiempo que es, tiene carencias importantes, como la no utilización de clientes en plataformas móviles, que solo esté disponible en inglés, y que no sea gratuito, ascendiendo su precio hasta los \$48 al mes por el uso asociado a quince usuarios.

Conclusión: 🚫 Pese a los defectos vistos en este producto, *Freckle* se considera como una de las mejores alternativas a ***The Time Bird*** actualmente.

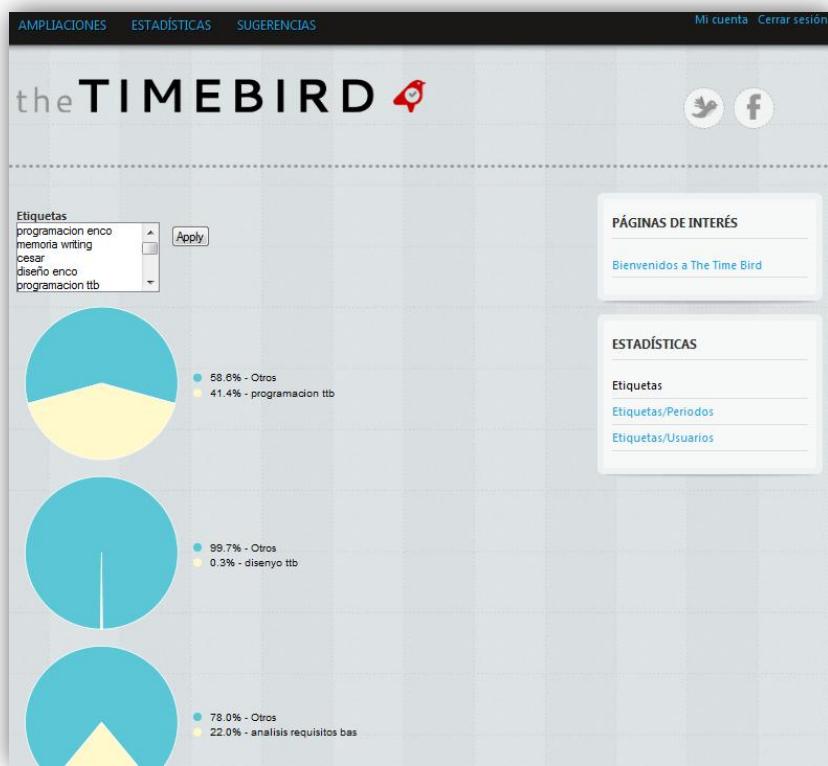
1.7.12. ***The Time Bird***

Sencilla y bonita herramienta, con la característica principal, de poder crear tareas con múltiples etiquetas, pudiendo éstas representar diversos conceptos, como proyectos, procesos o clientes. Además de dar la oportunidad al usuario de tener un margen de error de un minuto, a la hora de contabilizar sus etiquetas. ***The Time Bird*** centra todas sus funcionalidades en la elaboración de estadísticas con diferentes tipos de gráficos, por lo que tiene un buen aspecto visible, además de ser útil.

Características destacadas disponibles en ***The Time Bird***.

Multi-idioma	No
Diseño	Interfaz bonita y trabajada
Plataforma/s	Web, Android
Uso de gráficos	Sí
Ayuda online	Sí
Multi-tarea	Sí
Facilidad de uso	Muy fácil

Tabla 23: Características destacadas, disponibles en *The Time Bird*

Ilustración 16: Visualización de ejemplo de *The Time Bird*

Otras características de menor importancia:

Elaboración de facturas	No
Generación de informes exportables	No
Integración con otras herramientas	No
Geolocalización	No

Tabla 24: Visualización de ejemplo de *The Time Bird*

The Time Bird no está enfocado en la gestión de clientes, por lo que no incluye ninguna funcionalidad de facturación. Con **The Time Bird** lo que se pretende es aprender del tiempo, y a partir de ello, poder realizar facturas más acorde al tiempo que se le ha dedicado a las tareas realizadas.

En definitiva, este producto lanzado por la empresa **Atenea tech**, ofreciéndolo inicialmente de forma gratuita, le da un atractivo que no se puede encontrar en ningún otro gestor de tiempo actual, por lo que creemos que puede ser una muy buena opción para autónomos y pequeñas empresas, que no tengan la necesidad de utilizar herramientas potentes y gestión de usuarios.

2. Análisis inicial

En este capítulo se detalla el análisis realizado durante la fase inicial del proyecto. En primer lugar, y como fase inicial de un proyecto de desarrollo de *software*, se realizó un análisis de requerimientos, basados en los objetivos del proyecto, y que nos definen el sistema que tenemos que construir. Seguidamente se realizó una planificación inicial de las tareas a llevar a cabo, con su respectivo análisis económico. Finalmente se realizó un análisis de las tecnologías que se podían utilizar, y así justificar la tecnología que finalmente se escogió.

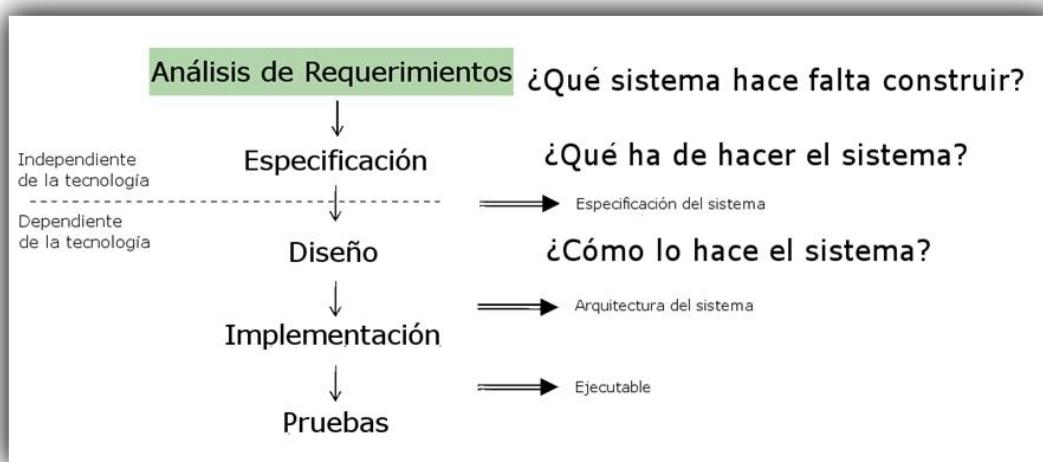


Ilustración 17: Etapas de desarrollo *software* (Análisis de Requerimientos)

2.1. Análisis de Requerimientos

2.1.1. Requerimientos funcionales

De la aplicación móvil:

- Acceso a los datos personales de un usuario previamente registrado.
- Creación, edición y posibilidad de deshabilitar etiquetas.
- Importación de etiquetas (con su respectivo estado).
- Activación/desactivación de etiquetas.
- Visualización de etiquetas así como el estado actual de éstas.

De la aplicación web:

- Visualización de estadísticas sobre las dimensiones: tiempo, etiquetas y usuario.

- Exportación de estado de etiquetas (en uso/ en desuso).
- Registro de etiquetas creadas.
- Registro de usuarios.
- Acceso de usuarios registrados.
- Registro de cambio de estados en etiquetas.
- Proceso de análisis de los datos, en *background* (**cron**¹⁰).

2.1.2. Requerimientos no funcionales

Genéricos:

- Es necesaria la elaboración de un manual de usuario, para que todos nuestros clientes, a pesar de ser una aplicación fácil de usar, sepan todos los servicios ofrecidos.
- Usabilidad: Han de tener una interfaz intuitiva y simple.
- Escalabilidad: Ha de ser posible añadir nuevas funcionalidades de forma sencilla.
- Utilidad: Las funcionalidades de la aplicación han de ser útiles para el usuario.

Aplicación móvil:

- Uso del SO móvil adecuado para trabajar como cliente, así realizar el envío de datos.
- Diseño: Un gran incentivo en la utilización de aplicaciones móviles, es que aparte de funcional, sea bonita. Se ha trabajado el diseño de la aplicación para que el usuario se sienta a gusto usándola.
- Margen de error: Se ha de proporcionar un margen de error de un minuto a los usuarios, para poder rectificar sus decisiones durante la creación de tareas, y así dejar la posibilidad de cambiar las etiquetas a usar.
- Activación simultánea de múltiples etiquetas (multitarea).

Aplicación Web:

- Servidor web: Un servidor web para permitir el acceso al contenido correcto, mediante el dominio `thetimebird.com`.
- Diseño: Un diseño atractivo de la aplicación hará que al usuario le dé gusto usarla.
- Utilización de **Drupal**, por razones comerciales de la empresa.

¹⁰ Cron es un administrador regular de procesos en segundo plano, que ejecuta procesos a intervalos regulares de tiempo.

- Módulos bien escritos en **Drupal**.
- Ha de contener toda la lógica de negocio, para que los clientes de **The Time Bird**, sean fácilmente ampliables a otras plataformas.
- Compatibilidad con navegadores Internet Explorer 6.0+, Safari 3.0+, Firefox 3.0+ y Opera 9.5+.
- Ofrecer las herramientas necesarias para estar en continuo contacto con los usuarios finales.

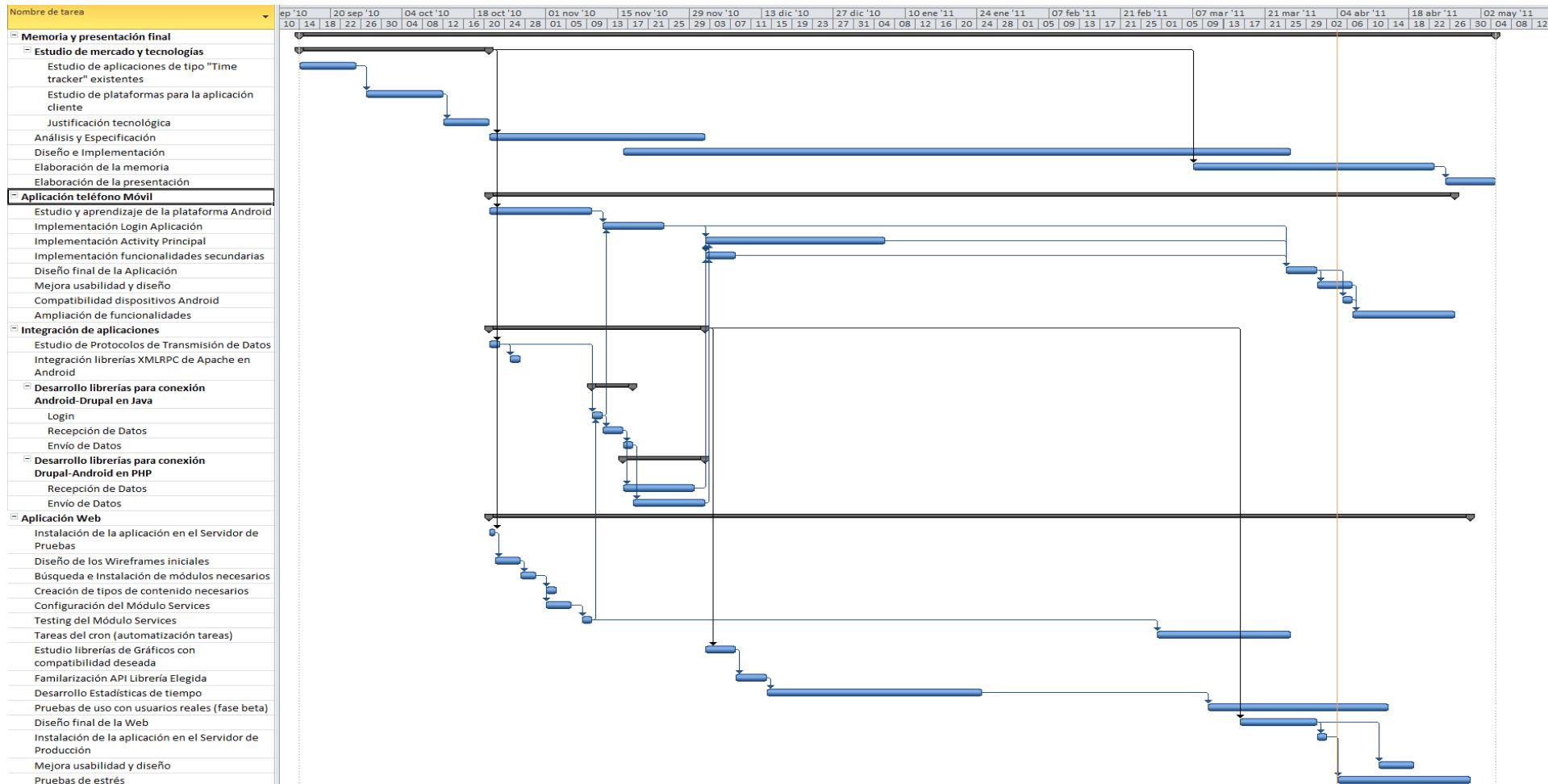
2.2. Planificación

Una vez definido el alcance de proyecto y sus requerimientos, la elaboración de la planificación es más realista. Aunque la planificación ha sido elaborada en la fase inicial de proyecto, se incluyen todas las tareas a llevar a cabo para cumplir con éxito todos los objetivos fijados.

Al no tener una fecha fija de finalización de proyecto, ni tampoco clientes de antemano, la planificación del tiempo que va durar todo el desarrollo, se ha estimado en base al primer día de trabajo en **Atenea tech**. El tiempo invertido se puede dividir en dos grandes secciones, detalladas a continuación:

- **Documentación:** La memoria es el documento base donde se encuentra toda la información relacionada con el proyecto. Su elaboración está presente desde el día de inicio de proyecto, con el fin de entender y justificar todas las decisiones tomadas en las diferentes fases de proyecto, ya sea el estudio de mercado realizado, o la elección de la tecnología a utilizar en cada una de las aplicaciones, así como toda la documentación técnica (análisis de requerimientos, especificación, diseño, etc.).
- **Desarrollo** de las dos aplicaciones y de las librerías necesarias para la comunicación de datos, incluyendo su **testeo**. Existe una gran interdependencia entre el desarrollo de las dos aplicaciones y la conexión de datos, debido a que cada una de las partes están muy ligadas a la hora de realizar las pruebas de las diferentes funcionalidades, y si no se van implementando a la vez, no se pueden llevar a cabo.

A continuación podemos ver de forma detallada, toda la planificación inicial representada por un diagrama de Gantt¹¹, todas las tareas planificadas inicialmente:



¹¹ El **diagrama de Gantt**, gráfica de **Gantt** o **carta de Gantt** es una popular herramienta gráfica cuyo objetivo es mostrar el tiempo de dedicación previsto para diferentes tareas o actividades a lo largo de un tiempo total determinado.

2.3. Análisis económico

Con el cómputo de todas las horas dedicadas al proyecto, se puede llevar a cabo el análisis económico total. Fundamentalmente va a estar compuesto por el valor de los recursos humanos y recursos materiales. A continuación se va a detallar el coste de cada uno de ellos.

- **Recursos humanos:** Durante todo el desarrollo del proyecto, se ha utilizado un único recurso humano que ha asumido cada uno de los roles que podamos encontrar un equipo de desarrollo de *software*, aunque en un caso real estos roles serían desempeñados por personas calificadas para ello. Los costes han sido calculados mediante el cómputo total de horas de desarrollo, y la asignación de recursos a cada una de las tareas realizadas ha sido realizada de forma aproximada, con 30 horas de carga de trabajo semanal, y sobreesfuerzo los últimos meses de proyecto.

Rol	Horas de trabajo	€/hora	Coste total en €
Gestor de proyecto ¹²	135	40	5400
Analista	327	30	9810
Programador	372	20	7440
Diseñador	53	20	1060
Tester	78	20	1560
Total recursos			25270

Tabla 25: Costes totales de recursos humanos

A continuación se puede ver el detalle del tiempo dedicado por cada uno de los recursos disponibles. Si una vez finalizado el proyecto hubiera desvíos de planificación, se tendría que revisar el coste total del proyecto.

¹² Se ha incluido el tiempo utilizado en la elaboración de la documentación de proyecto.

Análisis inicial

Nombre del recurso	Trabajo	Detalles	Septiembre 06/09	20/09	04/10	18/10	01/noviembre 01/11	15/11	01/diciembre 29/11	13/12	27/12	10/01	24/01	07/02	21/02	07/03	21/03	04/04	18/04	01 mayo 02/05	
- Gestor de proyecto	135 horas	Trabajo				2h											38h	35h	27h	21h	12h
Elaboración de la memoria	109 horas	Trabajo															38h	31h	27h	13h	
Elaboración de la presentación	20 horas	Trabajo																	8h	12h	
Instalación de la aplicación en el Servidor de Pruebas	2 horas	Trabajo				2h															
Instalación de la aplicación en el Servidor de Producción	4 horas	Trabajo																	4h		
- Analista	327 horas	Trabajo	24h	60h	60h	52h	28h	7h	13h			9h	8h	23h	10h	20h	6h	7h			
Estudio de aplicaciones de tipo "Time tracker" existentes	54 horas	Trabajo	24h	30h																	
Estudio de plataformas para la aplicación cliente	66 horas	Trabajo		30h	36h																
Justificación tecnológica	42 horas	Trabajo			24h	18h															
Análisis y Especificación	31 horas	Trabajo				12h	8h	7h	4h												
Diseño e Implementación	83 horas	Trabajo										9h	8h	23h	10h	20h	6h	7h			
Estudio y aprendizaje de la plataforma Android	28 horas	Trabajo					12h	16h													
Estudio de Protocolos de Transmisión de Datos	4 horas	Trabajo					4h														
Búsqueda e Instalación de módulos necesarios	6 horas	Trabajo					6h														
Creación de tipos de contenido necesarios	4 horas	Trabajo					4h														
Estudio librerías de Gráficos con compatibilidad deseada	9 horas	Trabajo							9h												
- Programador	372 horas	Trabajo				4h	26h	53h	45h	57h	52h	49h	6h			18h	14h	6h	22h	20h	
Implementación Login Aplicación	9 horas	Trabajo					2h	7h													
Implementación Activity Principal	56 horas	Trabajo							19h	21h	16h										
Implementación funcionalidades secundarias	7 horas	Trabajo							7h												
Integración librerías XMLRPC de Apache en Android	4 horas	Trabajo				4h															
Login	10 horas	Trabajo					10h														
Recepción de Datos	8 horas	Trabajo					3h	5h													
Envío de Datos	4 horas	Trabajo						4h													
Configuración del Módulo Services	11 horas	Trabajo				11h															
Recepción de Datos	19 horas	Trabajo					16h	3h													
Envío de Datos	30 horas	Trabajo					21h	9h													
Familiarización API Librería Elegida	10 horas	Trabajo						7h	3h												
Desarrollo Estadísticas de tiempo	124 horas	Trabajo							33h	36h	49h	6h									
Compatibilidad dispositivos Android	8 horas	Trabajo																	8h		
Tareas del cron (automatización tareas)	38 horas	Trabajo														18h	14h	6h			
- Ampliación de funcionalidades	34 horas	Trabajo																14h	20h		
- Diseñador	53 horas	Trabajo				7h										2h	26h	16h	2h		
Diseño final de la Aplicación	10 horas	Trabajo															10h				
Mejora usabilidad y diseño	8 horas	Trabajo														4h	4h				
Diseño de los Wireframes iniciales	7 horas	Trabajo				7h															
Diseño final de la Web	14 horas	Trabajo														2h	12h				
Mejora usabilidad y diseño	14 horas	Trabajo																12h	2h		
- Tester	78 horas	Trabajo				6h											12h	14h	14h	32h	
Testing del Módulo Services	6 horas	Trabajo				6h											12h	14h	6h		
Pruebas de uso con usuarios reales (fase beta)	32 horas	Trabajo															12h	14h	6h		
Pruebas de estrés	40 horas	Trabajo															8h	32h			

● **Recursos físicos:** Podemos clasificar los recursos físicos en recursos *software* y *hardware*. En cuanto a los recursos *software*, uno de los objetivos era el utilizar todo el material gratuito (software libre), o versiones en periodo de prueba en algún caso específico para minimizar costes, y por ello el *software* utilizado, en gran parte ha sido libre, a excepción de *Microsoft Office* y *Project* (con licencias de estudiante) y *Sparx Architect* (en su versión de prueba). En cuanto al material *hardware*, los mayores costes son el equipo utilizado para el desarrollo de la aplicación, y el teléfono móvil, muy importante para la parte de testeo de funcionalidades. Por otro lado, una vez acabado desarrollado todo el proyecto en el servidor de pruebas atenealabs.com se va a necesitar un servidor de producción, así como un dominio para poder acceder a ***The Time Bird***, thetimebird.com, dominio que por suerte estaba libre y se pudo adquirir.

Software	Tipo	Coste total en €
Ubuntu	Gratis	0
Dia	Gratis	0
Eclipse	Gratis	0
Microsoft Office 2010	Licencia universitaria	0
Microsoft Project 2010	Licencia universitaria	0
Gimp	Gratis	0
Sparx Architect	Periodo de prueba	0
Cacoo	Gratis	0
Total recursos		0

Tabla 26: Costes del *software* utilizado

Hardware	€/u.t	Coste total en €
Nexus S	-	440€
Servidor	20/mes	160
Dominio	10/año	10
Equipo portátil	-	600
Total recursos		1210

Tabla 27: Costes del *hardware* utilizado

Entonces, el coste total de los recursos presupuestados a **Atenea tech** asciende a **26.480€**.

2.4. Análisis tecnológico

Dadas las necesidades del producto a desarrollar, transformadas en requerimientos funcionales y no funcionales vistos anteriormente, se necesita hacer una elección entre las diferentes tecnologías para cada una de las aplicaciones.

El análisis tecnológico a realizar, va a estar ligado a los requerimientos no funcionales del proyecto. En primer lugar se realiza un análisis de plataformas móviles, seguido del de plataformas web.

2.4.1. Plataforma móvil

En los últimos años hemos vivido toda una revolución en el sector de la telefonía móvil e Internet, en el que el concepto de teléfono móvil ha cambiado radicalmente. Hace 10 años tener un teléfono móvil era considerado un capricho, y estaba fuera del alcance de mucha gente. Más adelante se empezaría a extender su uso entre usuarios, pero sobretodo en empresas, hasta llegar al punto en que hoy día hay tanta dependencia que no nos podríamos separar de ellos, debido a la aparición cada vez más frecuente de nuevas funcionalidades como la cámara de fotos, el *GPS*, *Bluetooth*, *Wifi* y finalmente la red 3G. A los teléfonos con estas características, se les empezó a llamar *Smartphones*, aunque la verdadera revolución fue con la salida del primer iPhone, marcando tendencias en el mercado. Después llegó Android, entrando con aspiración debido a la fragmentación de teléfonos en los que se podía instalar (algo que con el tiempo también ha jugado un papel en su contra). Además, cada vez son más las empresas que dedican sus negocios al desarrollo de aplicaciones móviles, debido a la facilidad con la que personas con pocos conocimientos técnicos pueden empezar a familiarizarse con el entorno de desarrollo y desarrollar pequeñas aplicaciones.

Dadas las posibilidades existentes y teniendo en cuenta nuestros objetivos, se puede escoger desarrollar la aplicación móvil con uno de los siguientes sistemas operativos, excluyendo Symbian porque se cree que es una tecnología a punto de extinguirse (ha pasado a ser una franquicia de **División Mobile Phones** como bien se comenta en el artículo **Nokia adopta Windows Phone 7. Todas las claves entre la alianza de Microsoft y Nokia [5]**.

En la siguiente ilustración, y para familiarizarnos con las plataformas, se van a ver los correspondientes logotipos de cada uno de los sistemas operativos a analizar.



Ilustración 18: Logotipos de los diferentes sistemas operativos a analizar

Se va a proceder con el análisis [6] tecnológico correspondiente, haciendo hincapié en ciertas características que se han creído importantes, para la tomar una decisión adecuada a las necesidades de proyecto. A continuación se muestra una tabla comparativa entre los sistemas operativos escogidos.

Características por SO	Android	webOS	iOs	Blackberry y	Windows Mobile
Fabricante	Google	Palm	Apple	Rim	Microsoft
Kernel	Linux con Máquina Virtual Dalvik	Linux	OS X	Propietario	Windows CE
Licencia	Apache y GPL	Cerrada	Código cerrado	Código cerrado	Cerrada
Lenguaje de Desarrollo	Java	Javascript ¹³	Objective C	Java	Visual C++ en .NET

Tabla 28: Comparativa entre sistemas operativos

Kernel

El Kernel es el núcleo de un sistema operativo, o dicho de otro modo, el *software* responsable de facilitar a los distintos programas acceso seguro al ordenador, o en forma más básica, es el encargado de gestionar recursos.

Una de las mayores diferencias entre un kernel de libre distribución y uno propietario, radica en el hecho de que los primeros cuentan con una amplia y experimentada

¹³ Lenguaje de programación interpretado, orientado a objetos e imperativo. Se utiliza principalmente del lado del cliente, implementado como parte de un navegador web, permitiendo mejoras en la interfaz de usuario y páginas web dinámicas.

comunidad de desarrolladores, que están dando soporte, muchos de ellos, en comunidades, foros o portales que están creados expresamente para ello. De esta forma se pueden detectar rápidamente problemas de seguridad, realización de mejoras, fallos, etc. En los sistemas cerrados o propietarios, es más costoso hacer este tipo de cosas y serán los propios desarrolladores del sistema los que tengan que abordar los problemas que puedan surgir. En este aspecto son tanto **Android** como **webOS** los que se llevan la palma, ya que a la hora de desarrollar aplicaciones, habrá mucho más material disponible por lo en principio sería menos costoso desarrollar una aplicación que en **iOs**, **Blackberry** y **Windows Mobile**.

Otro aspecto a destacar y que guarda una estrecha relación con el anterior, es la **adaptabilidad** de la plataforma (portabilidad), la capacidad o facilidad para poder adaptarla a diferentes terminales o máquinas. En este aspecto, es **Android** el que más adaptabilidad presenta, ya que cada vez está empleando más dispositivos.

Licencia

Como hemos podido observar, solo **Android** en este caso, es un sistema operativo totalmente abierto. Lo que permite que todas las versiones mejoradas de este sistema sigan siendo *software libre*. Esto significa que se evitará el riesgo de tener que competir con una versión modificada privativa de su propio trabajo.

Lenguaje de desarrollo

A la hora de desarrollar una aplicación para un teléfono móvil, dependiendo del Lenguaje nativo del SO, puede que cueste de forma sustancial el desarrollo, eso es debido a que, por ejemplo, empresas como **Apple** y **Microsoft** han desarrollado su propio lenguaje de desarrollo, que dependiendo lo abstracto que pueda llegar a ser, su aprendizaje será más o menos complicado. En el caso de Apple, el tiempo dedicado al aprendizaje de *Objective-C*¹⁴ y a la familiarización del entorno de su *SDK*, puede ser elevado.

Java, en cambio, ha sido el lenguaje más usado en las diferentes asignaturas de programación de la carrera, por lo que resulta más placentero poder seguir utilizándolo en el desarrollo de otras aplicaciones. Precisamente por eso, sería más sencillo desarrollar aplicaciones para **Android** y **Blackberry**, que para **iOs**, **webOS** y **Windows Mobile**.

¹⁴ Lenguaje de programación orientado a objetos, creado como un superconjunto de C pero que implementase un modelo de objetos parecido al de *Smalltalk*

Desarrollo de terceros

Para poder incluir **nuevas aplicaciones**, es importante que la plataforma admita desarrollo a terceros. Se puede decir que todas las plataformas que se utilizan para el análisis, disponen de un *SDK* público para desarrollar todo tipo de aplicaciones.

En el caso de **Apple**, cabe mencionar que si una empresa quiere desarrollar aplicaciones para uso interno, no hace falta pagar para entrar a formar parte del *iPhone Developer Program*, pero de otra forma hace falta pagar 99 dólares, por lo que subir aplicaciones para **iOS** puede ser algo caro, al igual que en **Blackberry** donde los desarrolladores deben pagar unos 200\$ para registrar hasta 10 aplicaciones. En cambio, una cuenta de desarrollador para **Android** cuesta 18€, por lo que es una muy buena opción.

Además, todas las aplicaciones deben estar disponibles en algún lugar de la red para que otros usuarios puedan realizar las descargas correspondientes. Apple, mediante **iOS**, fue el que revolucionó esta forma de dar a conocer las aplicaciones, mediante un mercado virtual, llamado *App Store*, siendo hoy día el más conocido y que mayor número de beneficios reporta. A éste le siguió el mercado de **Google**, **Android Market**, con mucho éxito aunque con una cuarta parte del número de aplicaciones disponibles en la *AppStore*. **Windows Mobile** y **Palm** han seguido los mismos pasos, y hace poco también lo ha hecho **Blackberry** con su *App World*, aunque a mucho de llegar al nivel de madurez de los dos primeros.

Interfaz de usuario

Es uno de los aspectos más importantes en los teléfonos de hoy día, ya que la interfaz es la herramienta con la que el usuario final tienen contacto directo y con lo que realiza su trabajo.

En este aspecto **Apple** con su iPhone y mediante **iOS** ha llevado siempre la delantera, con su pantalla táctil sin ningún tipo de botón a excepción del *Home* y sin necesidad del uso de punteros. Lo que es más importante y conocido, es su estilo de iconos y fácil acceso a las aplicaciones. **Android** ha seguido la línea y tiene una interfaz sencilla, intuitiva y amigable, aunque el tiempo de respuesta de los terminales, no es comparable al disponible en iOS. Por otro lado, **Windows Mobile** siempre ha recomendado (aunque no de forma necesaria) el uso de puntero. Además su interfaz es totalmente distinta a la iPhone y Android, a excepción de su última versión, que van teniendo características similares. **Rim** en cambio, dispone de teclados físicos en la mayoría de sus terminales, por lo que con **Blackberry** se hace un poco pesado usar un terminal para realizar tareas de forma rápida.

Algunas estadísticas

Algunos datos interesantes de un estudio realizado por *Nielsen*¹⁵ y *TAP TAP Networks*¹⁶ en el cuarto trimestre de 2010 de tendencias en telefonía móvil:

- Más de 9,1 millones de españoles ya navegan desde sus dispositivos en el cuarto trimestre de 2010.
- La penetración de Internet en España se acerca al 23,8% sobre el total de suscriptores móviles.
- Android se posiciona como el tercer sistema operativo entre los usuarios de Internet móvil.
- Aunque **Symbian** sigue en primera posición en conexiones a Internet en España, seguido de Apple, **Android** ya desbanca a Blackberry y continúa ganando posiciones en el ranking con el lanzamiento de múltiples dispositivos que soportan este sistema operativo.

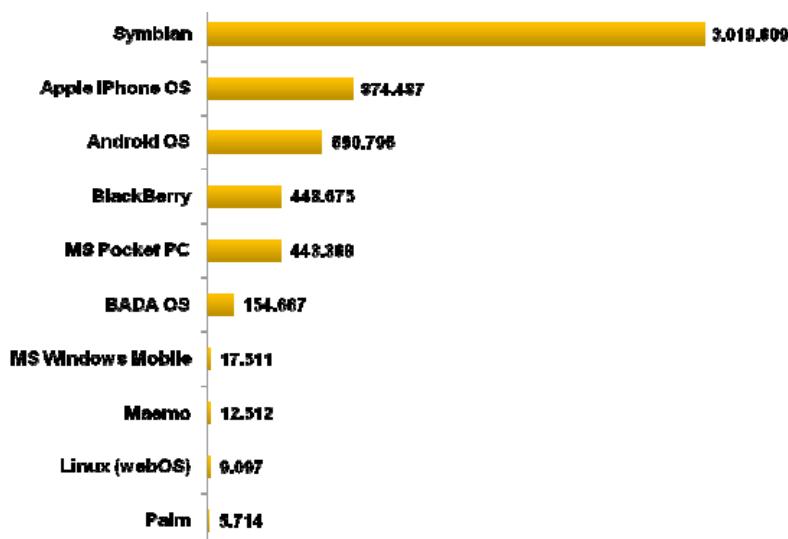


Ilustración 19: Audiencia de las diferentes plataformas móviles en el cuarto cuatrimestre de 2010 en España

Otro estudio interesante, es el realizado en Febrero del 2011 por la empresa *Lookout*¹⁷, llamado *AppGenome* [10] y relacionado con las dos plataformas que han conseguido ofrecer una tienda de una buena cantidad de aplicaciones y de calidad, Android e iOS.

¹⁵ *The Nielsen Company* [7] es una empresa de información y medios a nivel global, y es una de las fuentes líderes en información del mercado, información de medios de comunicación y audiencias de televisión, información online, aparatos móviles, publicaciones de negocios y entretenimiento.

¹⁶ *TAP TAP Networks* [8] es la Red móvil líder, para anunciantes y *Publishers Premium*. Es una empresa que se dedica a la publicidad en servicios móviles.



Ilustración 20: Iconos de las tiendas de aplicaciones *App Store* y *Android Market* respectivamente

El estudio concuerda que la *App Store* supera y por mucho en número de aplicaciones disponibles en el *Android Market*, aunque el número de aplicaciones de este último, está creciendo de tal forma, que aumentó un 127% desde Agosto de 2010, aumentando un 44% sólo en la *App Store*, por lo que son buenos datos para Google, ya que su mercado está creciendo con gran rapidez. Si se mantiene este ritmo, el mercado de aplicaciones de *Android Market* podría superar al de la *App Store* a mediados de 2012.

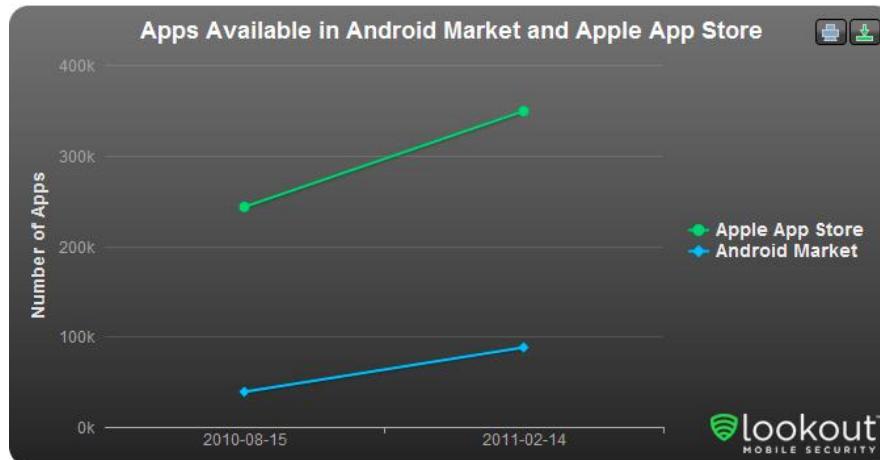


Ilustración 21: Aplicaciones disponibles en *Android Market* y *App Store* en Febrero de 2011

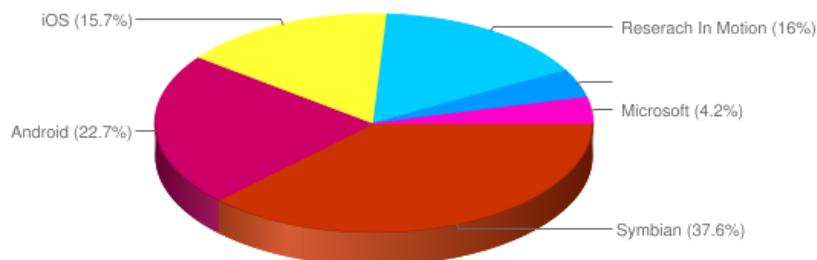
Otro análisis [11] interesante realizado por *Gartner*¹⁸ sobre las ventas de *smartphones* durante todo el 2010 en Estados Unidos, muestra que aunque *Symbian* siga dominando el mercado, la venta de terminales Android ha aumentado un 900% comparado al año 2009.

A continuación, se describe gráficamente el crecimiento de las diferentes plataformas, pudiéndose observar claramente el aumento de terminales con Android.

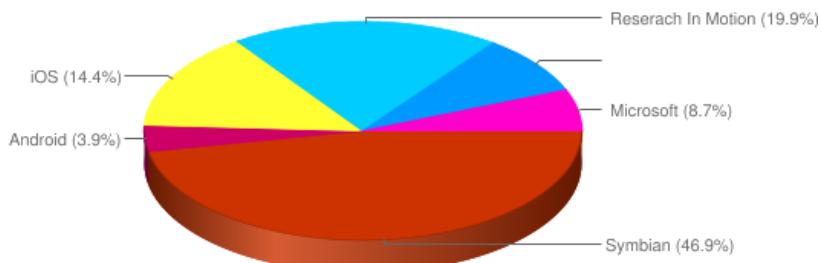
¹⁷ *Lookout* [9] es una empresa de seguridad para móviles dedicada a hacer la experiencia móvil segura para todos.

¹⁸ *Gartner S.A* es un proyecto de investigación de tecnologías de la información de firma consultiva con sede en Stamford, Connecticut, Estados Unidos.

Cuota de mercado mundial de Smartphones en 2010



Cuota de mercado mundial de Smartphones en 2009

Ilustración 22: Porcentaje de ventas de *smartphones* por sistema operativo, en todo el mundo

En términos generales, de estos estudios realizados, podemos obtener una serie de datos relevantes. *Symbian* sigue dominando el mercado de los *smartphones*. Apple es la que actualmente ofrece una mayor cantidad de aplicaciones para terminales iOS mediante su *App Store*, y Android es la plataforma que ha experimentado un mayor crecimiento además de ser de la que pronostica un mayor crecimiento en los próximos años.

Android, la mejor opción

De todo el estudio realizado, podemos comprobar que Android es la plataforma que además de tener toda una serie de características que la hacen muy interesante, es la que mayor expectativa de crecimiento tiene de cara al futuro sobre todos sus competidores, por lo que va a ser la elegida como principal cliente móvil para ***The Time Bird***.

A continuación se describe un pequeño resumen con los motivos por los cuales ha sido elegido el sistema operativo Android.

- Existen grandes comunidades que dan soporte, por lo que haría más fácil la resolución de problemas durante el desarrollo.
- La Licencia es abierta, por lo que su código está disponible a todo el público.
- La experiencia en el desarrollo de aplicaciones Java.
- El bajo coste que supone desarrollar una aplicación Android.

- Las expectativas de crecimiento previstas para este sistema operativo, lo hacen una opción realmente interesante.
- Proporciona una interfaz sencilla, intuitiva y amigable.



Ilustración 23: Figuras Android mostrándose vencedoras ante el padre de Apple, Steve Jobs

2.4.2. Plataforma web

Uno de los requerimientos para **The Time Bird** es el uso de **Drupal** como aplicación web. Pese a saber de antemano que se va a utilizar este gestor de contenidos, es necesario explicar por qué utilizar un *CMS*, así como las ventajas que ofrece éste, sobre otros gestores de contenido.

- Proporciona un conjunto de herramientas para desarrollar y mantener una web con relativa facilidad. El coste de mantener una aplicación web con un gran número de páginas o contenido sin un *CMS* puede llegar a ser muy alto.
- Realizar una aplicación web funcional y con un buen diseño desde cero, puede llegar a ser algo realmente costoso. Un gestor de contenido maduro, aporta las herramientas y añadidos necesarios para realizar una aplicación con muchas funcionalidades básicas.
- Un *CMS* aporta mucha flexibilidad. El número de funcionalidades puede ser fácilmente ampliable sin tener que preocuparnos de revisar lo anteriormente desarrollado. En cambio, hacer esto en una aplicación web hecha a mano puede llegar a ser una tarea realmente costosa.
- En cuanto a seguridad y control de acceso, la utilización de un *CMS* como Drupal, ahorra todo el trabajo que se tendría que realizar para garantizar ambas cosas. En la

comunidad, hay un grupo de seguridad [12] que colabora de forma activa en los fallos que van surgiendo. También cabe mencionar que realizar todo el control de acceso al contenido de una aplicación web desarrollada a mano, podría llegar a ser misión imposible.

- Los gestores de contenido proporcionan un entorno que permite realizar actualizaciones del propio sistema, así como de los añadidos instalados, sin tener que preocuparnos de las funcionalidades que hemos desarrollado.
- Haber adquirido experiencia con un gestor de contenido, normalmente facilita el manejo de otros gestores, ya que su concepto suele ser parecido, aunque cada uno de ellos tenga una estructura interna totalmente diferente.
- La consistencia en una página web es un aspecto fundamental. Ya sea el orden visual de páginas, como la disposición de contenido, dependiendo del navegador en que se visualice, puede ser una tarea dificultosa. Uno de los requerimientos de **The Time Bird**, es precisamente hacer compatible la aplicación web con el mayor número de navegadores posible, por lo que el uso de un *CMS*, al estar preparados para soportar la mayoría de los navegadores actuales, nos ahorrará trabajo de adaptabilidad.
- **Drupal**, hoy día, es una de las mejores herramientas para la elaboración de portales web, y la aplicación web de **The Time Bird** es lo que pretende ser.

Una vez justificado el uso de un gestor de contenido, cabe tener en cuenta que hay decenas de ellos, aunque en la actualidad solo hay unos pocos que estén en una fase de madurez avanzada, por lo que solo se van a analizar los más conocidos y/o utilizados; **Drupal, Joomla y Wordpress**.



Ilustración 24: Logotipos de Drupal, Joomla y Wordpress, respectivamente

Las versiones de los sistemas analizados son: Drupal 6.10, joomla 1.6 y wordpress 3.0.4, y la comparación entre ellos ha sido posibles gracias a **cms matrix**¹⁹. A continuación se procede a realizar una comparativa de los aspectos que suelen ser más importantes a la hora de elegir un gestor de contenido.

¹⁹ Página web que permite realizar comparaciones entre todos los gestores de contenido del mercado [13]

Seguridad

En cuanto a aspectos de seguridad se refiere, **Drupal** es el único gestor que ofrece pistas de auditoría, o lo que es lo mismo, que registra los cambios realizados en su sistema, sean actualizaciones o datos añadidos y borrados.

Otro aspecto importante, es la opción de añadir *CAPTCHA* durante el registro de usuarios en el sistema, algo posible tanto en **Drupal** como **Joomla**, pero no en **Wordpress**, lo cual podría prevenir de posibles ataques de robots al servidor.

Wordpress, a diferencia de los demás, provee de un área *Sandbox*, para realizar pruebas sin que afecte al resto del sitio web.

Soporte

Los tres gestores de contenido, tienen una gran comunidad de usuarios y desarrolladores detrás de ellos, dando todo tipo de soporte. Hay una gran cantidad de material en Internet, así como libros de desarrollo realizados por expertos, *APIs* que facilitan el desarrollo de nuevas funcionalidades, conferencias a nivel de estado y país, y más. Además, **Drupal** y **Joomla** proveen plantillas con código esqueleto para hacer más fácil el desarrollo de nuevas funcionalidades, a todo aquel que lo desee.

Facilidad de uso

Tanto **Wordpress** como **Drupal** permiten arrastre y jerarquización de contenido mediante *drag&drop*²⁰.

Drupal es el único sistema que le ofrece al usuario la posibilidad de crear áreas de contenido por defecto, estilos, u otros tipos de configuraciones, durante su configuración.

Tanto **Drupal** como **Wordpress** permite deshacer acciones de los usuarios en el caso en que éstos hayan realizado algún cambio.

Wordpress ofrece la posibilidad de subir archivos comprimidos al sitio de contenido estático para ser publicado en el sitio web. Característica muy útil en casos en que se necesite subir contenido en masa.

²⁰ Expresión informática que se refiere a la acción de mover con el ratón objetos de una ventana a otra o entre partes de una misma ventana. Los objetos arrastrados son habitualmente archivos, pero también pueden ser arrastrados otros tipos de elementos en función del programa.

Rendimiento

Los tres gestores en este caso, ofrecen mecanismos avanzados de almacenamiento de caché, que van más allá de la caché de simples páginas, ya sea navegación, formularios, etc.

Wordpress además, es el único que permite exportar contenido estático del sitio web, como HTML, lo cual podría ser ventajoso en según qué situaciones.

Flexibilidad

Tanto **Drupal** como **Joomla** permiten que su contenido sea reflejado en varios sitios web (multi-sitio).

Los tres gestores permiten añadir propiedades al perfil del usuario, metadatos y contenido multilenguaje y reescritura de *URLs*²¹.

Gestión

Drupal es el único gestor que incorpora un sistema de flujo de trabajo integrado, que puede ser usado en gestiones de procesos de negocio u otros tipos de procesos que van más allá de la aprobación de contenido. Además, es el único que permite crear contenido en un servidor y ser fácilmente replicado a otro servidor.

Por otro lado, son **Wordpress** y **Joomla**, los que tienen un sistema de recuperación de contenido borrado por administradores o usuarios.

Los tres sistemas proveen de administración en línea, estadísticas del sitio web, gestión de usuarios y de traducción y otras características de menor importancia.

Aplicaciones integradas

Los tres gestores ofrecen una gran cantidad de posibilidades a la hora de elaborar una aplicación web. Ofrecen la posibilidad de crear *blogs*, *chats*, gestión de contactos, *fórum*s, eventos de calendario, *wikis*, búsquedas, etc.

²¹ Técnica utilizada por los *webmasters* para optimizar el posicionamiento de sus portales con un contenido dinámico. Permite reescribir *URLs* automáticas, a *URLs* legibles y con significado para los seres humanos e identificables por los buscadores.

Por otro lado, **Joomla** tiene una ventaja respecto a los otros en cuanto a gestión estadística se refiere, dando la posibilidad a los usuarios de crear sus propias estadísticas a partir de grupos de datos, consultas SQL y otros.

Comercio electrónico

Tanto **Drupal** como **Joomla** ofrecen la posibilidad de utilizar añadidos para comercio electrónico. Este aspecto podría ser de relevancia para futuras ampliaciones de la aplicación web de **The Time Bird**, ya que, aunque no se haya realizado un plan de ventas específico, una posibilidad sería ofrecer suscripciones con una serie de ventajas para los usuarios, con posibilidad de pago integrado en la misma aplicación.

Características generales

Aunque sea difícil comparar los gestores por sus características, todos ellos son conocidos por aportar una serie de beneficios en diferentes tipos de aplicaciones web.

Wordpress, por ejemplo, es uno de los gestores de contenido más enfocados a la creación de *blogs*, por lo que, actualmente es el gestor con el que se realizan más páginas web.

Joomla es un sistema mejorado de *Mambo*²². Posee un gran número de extensiones que hace que tenga un gran número de funcionalidades. Al igual que **Drupal**, se utiliza sobre todo para la creación de portales web, aunque éste último haya llegado a ofrecer mucho más que eso, debido a su gran arquitectura flexible. Drupal también posee un gran número de módulos que permite ampliar enormemente las funcionalidades existentes.

Algunas estadísticas

Algunos datos muy interesantes son los que muestra **buildwith** en su *blog* [13] sobre gestores de contenido, actualizados a 9 de Mayo de 2011.

En los siguientes gráficos se muestran porcentajes de utilización de gestores de contenido en los *top sites* de internet (aunque la empresa no dice exactamente cómo ha sacado los datos, se supone que son los sitios más visitados en Internet).

²² *Mambo* es un sistema CMS basado en el lenguaje de programación PHP y base de datos SQL de código abierto. Basa todo su aspecto en plantillas o temas.

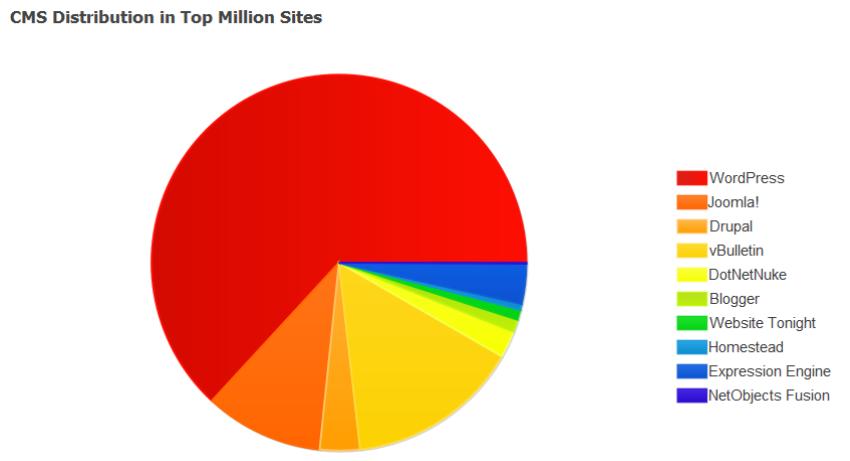


Ilustración 25: Distribución de CMS calculada sobre el top 1.000.000 de sitios web

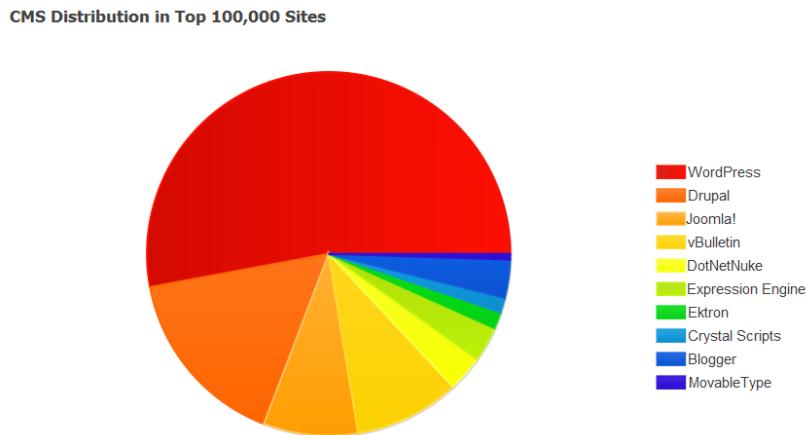


Ilustración 26: Distribución de CMS calculada sobre el top 10.000 sitios web

Los resultados de estos gráficos, nos hacen pensar que cuanto más acotemos a *top sites*, más terreno va ganando el desarrollo con Drupal. Eso es debido a la gran cantidad de *blogs* existentes en Internet. Al ser Wordpress la mejor plataforma para la elaboración de este tipo de páginas, hace que este sea el más utilizado con mucha diferencia.

Algunos otros datos muy interesantes, son los mostrados por las gráficas que veremos a continuación, sobre tendencias de uso de los tres gestores de contenidos, sobre una larga selección de páginas web consultadas por ***buildwith***, la empresa que ha realizado el análisis.

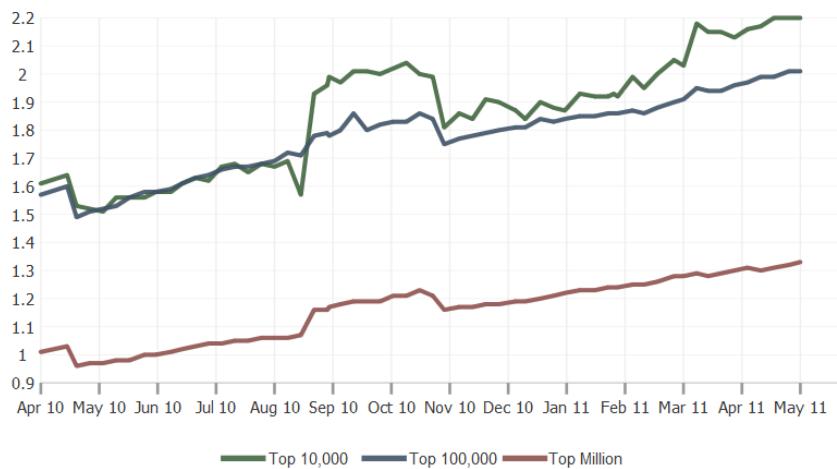


Ilustración 27: Tendencias de uso de páginas consultadas con Drupal

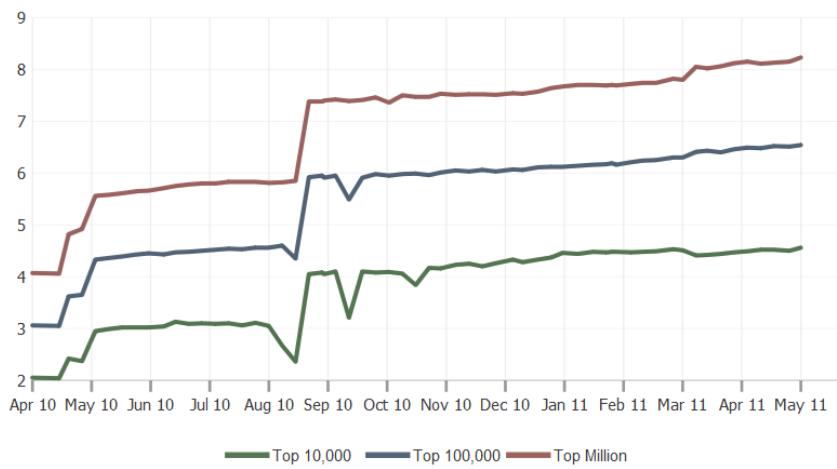


Ilustración 28: Tendencias de uso de páginas consultadas con Wordpress



Ilustración 29: Tendencias de uso de páginas consultadas con Joomla

La conclusión que podemos sacar de los gráficos anteriores, es el enorme **crecimiento** que está experimentando **Drupal** en relación a sus competidores, por lo que es muy buena señal para este gestor, ya que posee una expectativa de crecimiento genial, y una aceptación cada vez más notable en el mundo del desarrollo web.

Drupal, la mejor opción

Aunque del estudio realizado, podemos obtener información valiosa entorno al crecimiento de **Drupal** respecto a sus competidores, discutir cuál de estos gestores de contenido es mejor que otro para uso genérico, sería más bien imposible, por lo que es necesario estudiar el tipo de aplicación web a realizar y estudiar las ventajas que aportan cada uno de ellos. En nuestro caso, la elaboración de una aplicación web para **The Time Bird** para que los usuarios puedan analizar los datos enviados desde el cliente móvil, la convierte en un portal informativo, por lo que Drupal se adapta perfectamente a nuestras necesidades.

También cabe mencionar que, pese a que Joomla y Drupal sean CMS similares en cuanto a estructura base, una de las grandes diferencias como desarrollador, es la curva de aprendizaje a seguir para conseguir dominarlos, siendo Drupal bastante más complicado. Aunque podría ser un motivo más que suficiente para elegir *Joomla*, la experiencia del autor y de **Atenea tech** en la creación de aplicaciones web con Drupal, han sido vitales en la decisión. Además, su flexibilidad y adaptabilidad, así como la gran cantidad de módulos adicionales disponibles, hacen que sea idóneo en un proyecto de estas características, en el que es muy probable que se tenga que aumentar el número de funcionalidades en un futuro cercano.



Ilustración 30: Drupal es el gestor de contenido elegido

3. Especificación

La especificación es un documento técnico que describe, en la forma más precisa posible y en términos simples, las necesidades y funciones específicas de un sistema determinado.

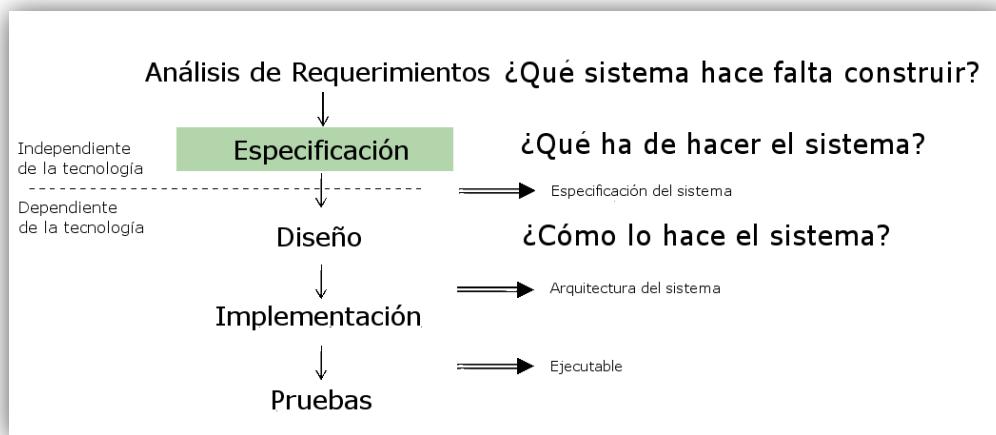


Ilustración 31: Etapas de desarrollo *software* (Especificación)

En el primer apartado se definirán los casos de uso, técnica que sirve para definir interacciones entre un sistema y sus actores, en respuesta a un evento que inicia un actor principal sobre el sistema y que surgen de la captura de requisitos funcionales del sistema. Primeramente se mostrarán los diagramas de casos de uso, que definirán de forma gráfica los casos de uso, y seguidamente se describirá cada uno de ellos de forma detallada.

En el segundo apartado se describirán los modelos conceptuales o diagramas de clases, diagramas estáticos que describen la estructura del sistema, mostrando clases, atributos y relaciones entre ellos.

Finalmente se describirá el modelo de comportamiento del sistema, formado por los diagramas de secuencia del sistema y los contratos de las operaciones.

3.1. Casos de uso

Acabada de ver la definición de los casos de uso, se va a proceder a mostrar los actores y diagramas para cada uno de los sistemas. Cabe destacar que hay dos sistemas diferentes, donde los actores (que son toda entidad externa al sistema que guarda relación con éste y que le demanda una funcionalidad) estarán diferenciados debido a los requerimientos funcionales de cada uno de los sistemas.

3.1.1. Aplicación móvil

Actores

En este sistema encontramos dos actores claramente diferenciados, el **usuario anónimo** que tendrá acceso a funcionalidades más bien informativas, y el **usuario autenticado** que podrá realizar la mayoría de operaciones de importancia del sistema.

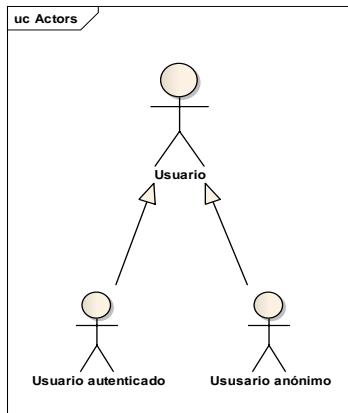


Ilustración 32: Jerarquización de actores de la aplicación móvil

Diagrama de casos de uso

A continuación podemos ver la siguiente figura que ilustra los casos de uso de la aplicación móvil:

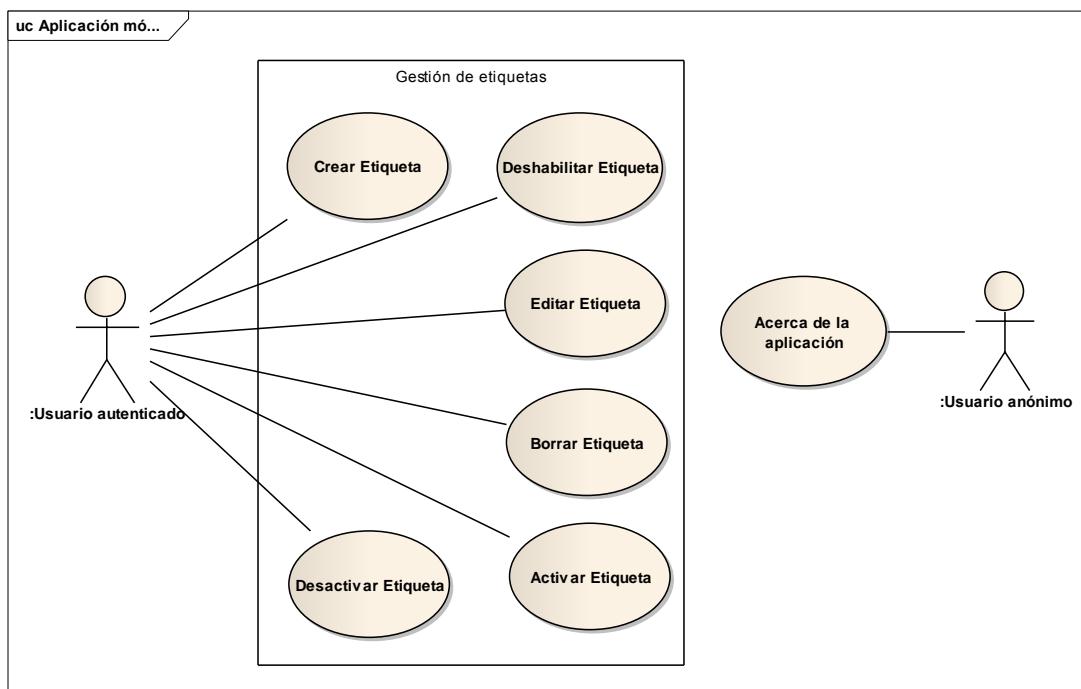


Ilustración 33: Diagrama de casos de uso de la aplicación móvil

Descripción de casos de uso

Seguidamente se detallan cada uno de los casos de uso de la aplicación móvil:

Acerca de la aplicación
Descripción: Se desea saber más información de la aplicación que se está utilizando. La información a mostrar se compone de agradecimientos e información de autoría.
Autores: Usuario anónimo.
Precondiciones: -
<u>Flujo Principal</u> <ol style="list-style-type: none"> 1. El usuario selecciona la opción para obtener más información sobre la aplicación. 2. El sistema muestra la información compuesta por agradecimientos del autor e información de la empresa autora de la aplicación. 3. Acaba el caso de uso.

Gestión de etiquetas

Crear etiqueta
Descripción: Se desea crear una etiqueta, para dar nombre a un proceso, proyecto o cliente para en el que el usuario quiera empezar a contabilizar el tiempo que le dedica.
Autores: Usuario autenticado.
Precondiciones: -
<u>Flujo Principal</u> <ol style="list-style-type: none"> 1. El usuario selecciona la opción de crear una nueva etiqueta. 2. El sistema muestra un campo de texto editable. 3. El usuario introduce el nombre deseado para la etiqueta. 4. El sistema registra los datos introducidos. 5. El sistema muestra un mensaje de confirmación al usuario. 6. Acaba el caso de uso.
<u>Flujo Alternativo { 3.Cancelación }</u> <ol style="list-style-type: none"> 1. El usuario cancela la edición de etiquetas. 2. El caso de uso vuelve al punto 1 del flujo principal.
<u>Flujo Alternativo { 4.Conexión de datos }</u> <ol style="list-style-type: none"> 1. Durante el registro, el sistema determina que no hay conexión de datos. 2. El caso de uso vuelve al punto 1 del flujo principal.

Editar etiqueta

Descripción: Se desea editar el nombre de una etiqueta, pudiendo así renombrar un proceso, proyecto o cliente con el que pudiéramos estar trabajando.
Autores: Usuario autenticado.
Precondiciones:

1. Ha de existir alguna etiqueta.
Flujo Principal
1. El usuario selecciona alguna de sus etiquetas.
2. El sistema muestra un conjunto de operaciones relacionadas a la etiqueta seleccionada.
3. El usuario selecciona la opción de editar la etiqueta.
4. El sistema muestra un campo de texto editable.
5. El usuario introduce el nombre deseado para la etiqueta.
6. El sistema registra los cambios introducidos.
7. El sistema muestra un mensaje de confirmación al usuario.
8. Acaba el caso de uso.
Flujo Alternativo { 5.Cancelación }
1. El usuario cancela la opción de editar etiquetas.
2. El caso de uso vuelve al punto 1 del flujo principal.
Flujo Alternativo { 6.Conexión de datos }
1. Durante el registro, el sistema determina que no hay conexión de datos.
2. El caso de uso vuelve al punto 1 del flujo principal.

Deshabilitar etiqueta
Descripción: Se desea deshabilitar una de las etiquetas creadas con anterioridad.
Autores: Usuario autenticado.
Precondiciones:
1. Ha de existir alguna etiqueta.
Flujo Principal
1. El usuario selecciona alguna de sus etiquetas.
2. El sistema muestra un conjunto de operaciones relacionadas a la etiqueta seleccionada.
3. El usuario selecciona la opción de deshabilitar la etiqueta.
4. El sistema registra los cambios introducidos.
5. El sistema muestra un mensaje de confirmación al usuario.
6. Acaba el caso de uso.
Flujo Alternativo { 3.Cancelación }
3. El usuario cancela la edición de deshabilitar etiquetas
4. El caso de uso vuelve al punto 1 del flujo principal.
Flujo Alternativo { 4.Conexión de datos }
3. Durante el registro, el sistema determina que no hay conexión de datos.
4. El caso de uso vuelve al punto 1 del flujo principal.

Activar etiqueta
Descripción: Una etiqueta se activa para empezar a contabilizar el tiempo que se invierte en el concepto al que representa.

Autores: Usuario autenticado.
Precondiciones:
1. Ha de existir alguna etiqueta.
<u>Flujo Principal</u>
1. El usuario selecciona una etiqueta. 2. El sistema registra el cambio de estado de la etiqueta. 3. Acaba el caso de uso.
<u>Flujo Alternativo { 2.Conexión de datos }</u>
1. Durante el registro, el sistema determina que no hay conexión de datos. 2. El caso de uso vuelve al punto 1 del flujo principal.

Desactivar etiqueta
Descripción: Una etiqueta se desactiva para parar la contabilidad del tiempo asociado al concepto al que representa.
Autores: Usuario autenticado.
Precondiciones:
1. Ha de existir alguna etiqueta. 2. La etiqueta ha de estar activada.
<u>Flujo Principal</u>
1. El usuario selecciona una etiqueta 2. El sistema registra el cambio de estado de la etiqueta. 3. Acaba el caso de uso.
<u>Flujo Alternativo { 2.Conexión de datos }</u>
1. Durante el registro, el sistema determina que no hay conexión de datos. 2. El caso de uso vuelve al punto 1 del flujo principal.

3.1.2. Aplicación web

Actores

En este sistema podemos encontrar tres actores diferentes. El **usuario anónimo**, que solo podrá tener acceso a la información de la página principal de la aplicación web. El **usuario autenticado** que podrá realizar la mayoría de operaciones de importancia del sistema, y el **administrador** que tendrá acceso a toda la aplicación, incluyendo todas sus opciones de configuración y mantenimiento.

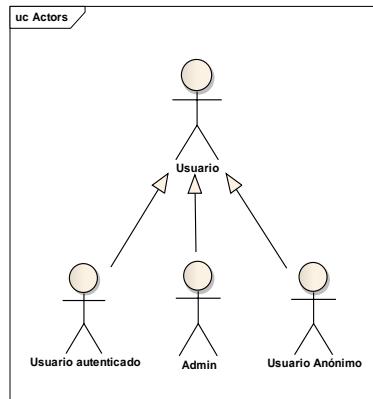


Ilustración 34: Jerarquización de actores de la aplicación web

Diagrama de casos de uso

A continuación podemos ver la siguiente figura que ilustra los casos de uso de la aplicación web:

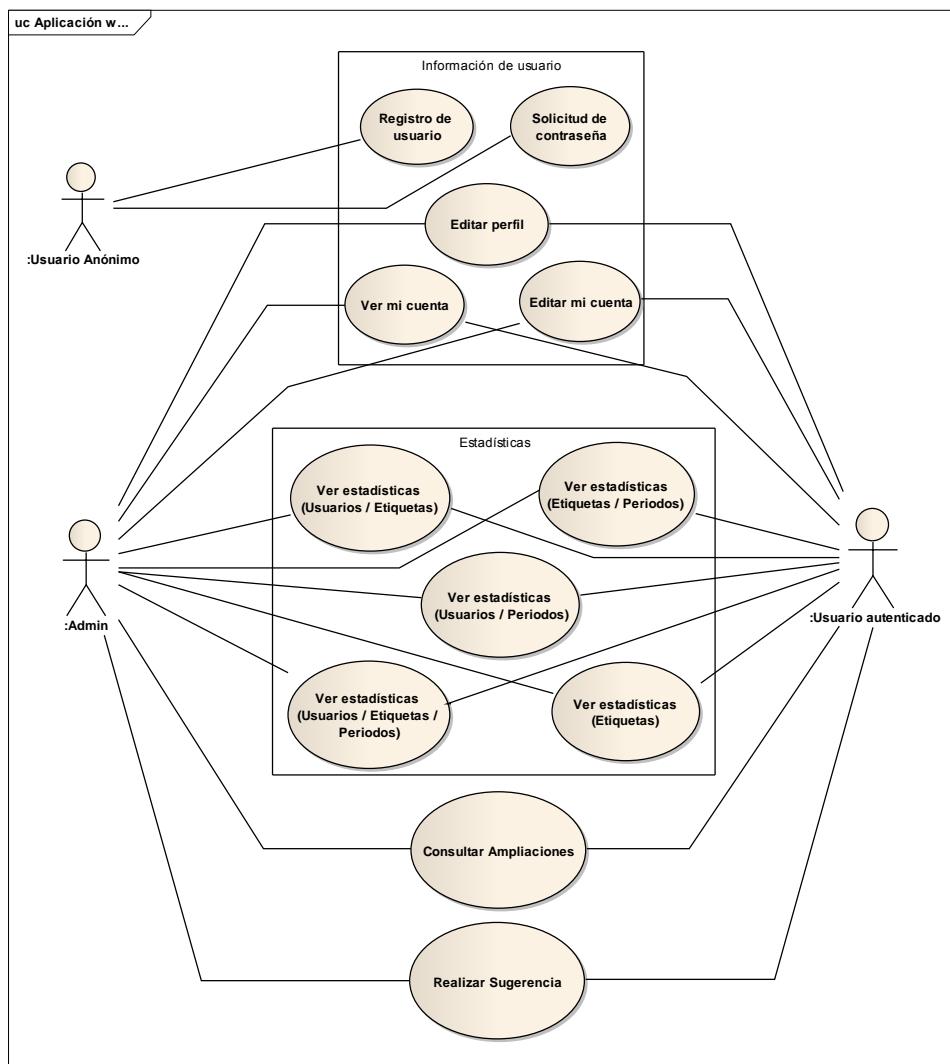


Ilustración 35: Diagrama de casos de uso de la aplicación web

Descripción de casos de uso

Seguidamente se detallan cada uno de los casos de uso de la aplicación web:

Consultar ampliaciones
Descripción: Se desea consultar las futuras ampliaciones de la aplicación web.
Autores: Usuario autenticado.
Precondiciones: -
<u>Flujo Principal</u> <ol style="list-style-type: none"> 1. El usuario selecciona la opción de Ampliaciones. 2. El sistema muestra la información correspondiente.

Realizar sugerencia
Descripción: Se desea realizar alguna sugerencia de mejora o nueva funcionalidad para <i>The Time Bird</i> .
Autores: Usuario autenticado.
Precondiciones: -
<u>Flujo Principal</u> <ol style="list-style-type: none"> 1. El usuario selecciona la opción de Sugerencias. 2. El usuario introduce su nombre. 3. El usuario introduce su correo electrónico. 4. El usuario introduce la sugerencia que tiene para <i>The Time Bird</i>. 5. El sistema registra los cambios. 6. El sistema envía un correo de confirmación al usuario. 7. Acaba el caso de uso.

Información de usuario

Registro de usuario
Descripción: Se desea registrarse en <i>The Time Bird</i> y poder disfrutar de sus funcionalidades.
Autores: Usuario anónimo.
Precondiciones: -
<u>Flujo Principal</u> <ol style="list-style-type: none"> 1. El usuario selecciona la opción de registro. 2. El usuario introduce su nombre. 3. El usuario introduce su correo electrónico. 4. El usuario selecciona a los usuarios que podrán tener acceso a sus datos. 5. El usuario selecciona la opción de crear nueva cuenta. 6. El sistema registra al usuario y envía un correo electrónico al usuario.

7. El sistema muestra un mensaje de confirmación al usuario.
8. Acaba el caso de uso.

Flujo Alternativo { 6.Error de datos }

1. Durante el registro, el sistema determina que no el correo electrónico introducido ya está registrado.
2. El caso de uso vuelve al punto 3 del flujo principal.

Solicitud de contraseña

Descripción: Se desea obtener una nueva contraseña, porque la anterior ha sido olvidada.

Autores: Usuario anónimo.

Precondiciones: -

Flujo Principal

1. El usuario selecciona la opción solicitar una nueva contraseña.
2. El usuario introduce su nombre o correo electrónico.
3. El usuario selecciona la opción de *Log in*.
4. El sistema registra los cambios y envía un correo electrónico al usuario.
5. El sistema muestra un mensaje de confirmación al usuario.
6. Acaba el caso de uso.

Flujo Alternativo { 4.Error de datos }

1. Durante el registro de cambios, el sistema detecta que el usuario o correo electrónico introducido no existe.
2. El caso de uso vuelve al punto 2 del flujo principal.

Ver mi cuenta

Descripción: Se desea ver los datos de la cuenta asociados al usuario.

Autores: Usuario autenticado.

Precondiciones: -

Flujo Principal

1. El usuario selecciona la opción de ver su cuenta.
2. El sistema muestra los datos asociados a la cuenta (nombre de usuario, lista de usuarios que pueden ver sus datos e historial de registro).
3. Acaba el caso de uso.

Editar mi cuenta

Descripción: Se desea modificar los datos de la cuenta asociados al usuario.

Autores: Usuario autenticado.

Precondiciones: -

Flujo Principal

1. El usuario selecciona la opción de editar su cuenta.
2. El usuario introduce un nuevo usuario.
3. El usuario introduce un nuevo correo electrónico.
4. El usuario introduce una nueva contraseña y la vuelve a repetir.
5. El usuario selecciona el estado que desea para su cuenta.
6. El usuario selecciona la opción de guardar los cambios.
7. El sistema registra los cambios realizados.
8. El sistema muestra un mensaje de confirmación al usuario
9. Acaba el caso de uso.

Flujo Alternativo { 7.Error de datos }

1. Durante el registro, el sistema determina que el usuario ya está registrado, el correo electrónico ya está registrado o los campos de contraseña no coinciden.
2. El caso de uso vuelve al punto 2 del flujo principal.

Editar perfil

Descripción: Se desea modificar los datos del perfil asociado al usuario.

Autores: Usuario autenticado.

Precondiciones: -

Flujo Principal

1. El usuario selecciona la opción de editar su perfil.
2. El usuario selecciona los usuarios que podrán acceder a sus datos.
3. El usuario selecciona la opción de guardar los cambios.
4. El sistema registra los cambios realizados.
5. El sistema muestra un mensaje de confirmación al usuario.
6. Acaba el caso de uso.

Estadísticas

Ver estadísticas (Etiquetas)

Descripción: Se desea saber el tiempo dedicado a cada una de las etiquetas creadas por el usuario

Autores: Usuario autenticado.

Precondiciones: -

Flujo Principal

1. El usuario selecciona la opción de ver las estadísticas de Etiquetas.
2. El usuario selecciona las etiquetas de las cuales quiere que se generen gráficos.
3. El sistema filtra la selección y muestra los datos correspondientes.
4. Acaba el caso de uso.

Ver estadísticas (Etiquetas / Periodos)
Descripción: Se desea saber el tiempo dedicado a cada una de las etiquetas creadas por el usuario, en un cierto periodo de tiempo determinado
Autores: Usuario autenticado.
Precondiciones: -
<u>Flujo Principal</u> <ol style="list-style-type: none"> 1. El usuario selecciona la opción de ver las estadísticas de Etiquetas / Periodos. 2. El usuario selecciona las etiquetas de las cuales quiere que se generen gráficos. 3. El usuario selecciona los periodos de inicio y final entre los que quiere filtrar el uso de las etiquetas. 4. El sistema filtra la selección y muestra los datos correspondientes. 5. Acaba el caso de uso.

Ver estadísticas (Etiquetas / Usuarios)
Descripción: Se desea saber el tiempo dedicado a cada una de las etiquetas creadas por usuarios que nos permiten ver sus datos.
Autores: Usuario autenticado.
Precondiciones: -
<u>Flujo Principal</u> <ol style="list-style-type: none"> 1. El usuario selecciona la opción de ver las estadísticas de Etiquetas / Usuarios. 2. El usuario selecciona los usuarios de los que quiere obtener los datos del tiempo dedicado a cada una de sus etiquetas. 3. El usuario selecciona las etiquetas de las cuales quiere que se generen gráficos. 4. El sistema filtra la selección y muestra los datos correspondientes. 5. Acaba el caso de uso.

Ver estadísticas (Usuarios / Periodos)
Descripción: Se desea saber el tiempo total de todas las etiquetas, dedicado para un cierto periodo de tiempo.
Autores: Usuario autenticado.
Precondiciones: -
<u>Flujo Principal</u> <ol style="list-style-type: none"> 1. El usuario selecciona la opción de ver las estadísticas de Usuarios / Periodos. 2. El usuario selecciona los usuarios de los que quiere obtener los datos del tiempo total dedicado a todas sus etiquetas. 3. El usuario selecciona los periodos de inicio y final entre los que quiere filtrar el tiempo total dedicado para los usuarios seleccionados. 4. El sistema filtra la selección y muestra los datos correspondientes. 5. Acaba el caso de uso.

Ver estadísticas (Etiquetas / Usuarios / Periodos)
Descripción: Se desea saber el tiempo dedicado a cada una de las etiquetas entre los periodos de tiempo seleccionadas, agrupadas por usuarios.
Autores: Usuario autenticado.
Precondiciones: -
<u>Flujo Principal</u> 1. El usuario selecciona la opción de ver las estadísticas de Etiquetas / Usuarios / Periodos. 2. El usuario selecciona las etiquetas de las cuales quiere que se generen gráficos. 3. El usuario selecciona los usuarios por los que quiere agrupar el uso de las etiquetas seleccionadas. 4. El usuario selecciona los períodos de inicio y final entre los que quiere filtrar el tiempo total dedicado a las etiquetas seleccionadas, por los usuarios seleccionados. 5. El sistema filtra la selección y muestra los datos correspondientes. 6. Acaba el caso de uso.

3.2. Modelo conceptual

Una vez determinados los casos de uso, se debe determinar un modelo de datos para el sistema. El modelo de datos está formado por clases, atributos y relaciones, que representarán ciertos conceptos (objetos) existentes en el sistema, los cuales tienen una identidad y son distinguibles entre ellos.

El modelo conceptual de los sistemas es algo parecido, ya que tanto los usuarios como las etiquetas son conceptos presentes en ambos.

3.2.1. Aplicación móvil

A continuación se muestra el modelo conceptual de la aplicación móvil. Para ello se mostrará el conjunto de objetos de dominio con relaciones bastante simples. Hay que tener en cuenta que la simplicidad de la aplicación como concepto, se ha de ver reflejada también en su lógica de negocio.

Este modelo de datos representa un sistema en el que un **usuario Autenticado** en la aplicación, **posee etiquetas** con un identificador único, un nombre y el tiempo dedicado (siendo éste inicialmente 0). Las etiquetas pueden estar **activas** o **inactivas**.

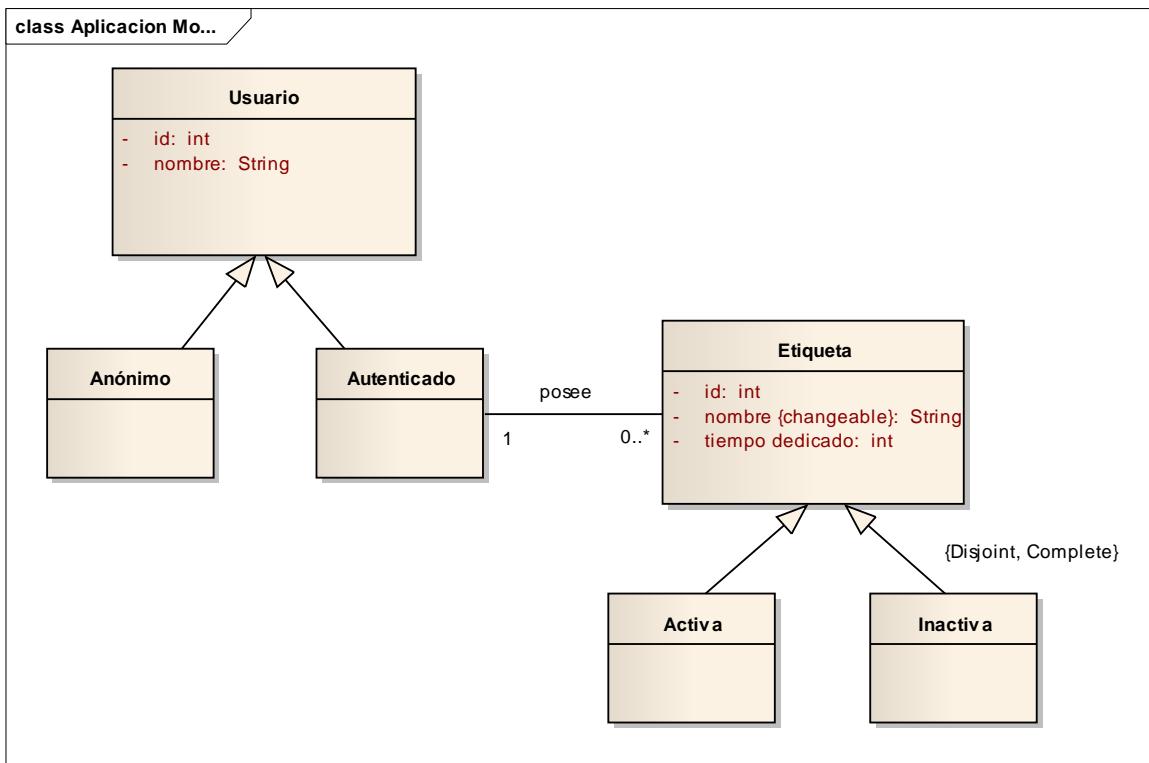


Ilustración 36: Modelo conceptual aplicación móvil

3.2.2. Aplicación Web

A continuación se muestra el modelo conceptual de la aplicación web. Para ello, al igual que en el modelo anterior, se mostrarán los objetos que intervienen en el sistema.

Para describir de la forma más real posible el modelo de la aplicación web, se mostrará nuestro modelo conceptual para la aplicación web, y después se mostrará en detalle, parte del gran modelo conceptual en que se estructura Drupal, viendo la relación entre ambos.

En el modelo, existen nodos o contenido (en términos de desarrollo web). Cada uno de los nodos, así como los usuarios, se pueden representar mediante clases de objetos. Algunos tipos de contenido o nodos de tipo **Página** e **Historia** pueden ser utilizados únicamente por el administrador y sirven para generar contenido informativo en la aplicación web, aunque en la lógica de negocio de ésta, juegan un papel importante los nodos de tipo **Tarea no procesada**, **Etiquetas**, **Periodos**, **Perfiles**, **Tareas procesadas** y **Sugerencias**.

La estructura necesaria para manipular la información se podría dividir en dos fases. En la primera, los usuarios **Autenticados** crean **Tareas no procesadas**, que contienen la información básica obtenida de la aplicación móvil, y que forman parte de un proceso puramente transaccional. En la segunda fase, a partir de la información de las tareas mencionadas, se crean **Tareas procesadas**, que mediante la relación **formada_por**,

enlaza con **Etiquetas**, el usuario **Autenticado** que creó la **Tarea no procesada** y el conjunto de **Periodos** de tiempo en que las etiquetas han sido utilizadas.

Además, cabe destacar que un usuario **Autenticado tiene un Perfil**, que también **tiene** una lista de usuarios **Autenticados** a los les permite obtener sus datos en formato gráfico.

Por último, un usuario **Autenticado tiene** tantas **Sugerencias**, como desee.

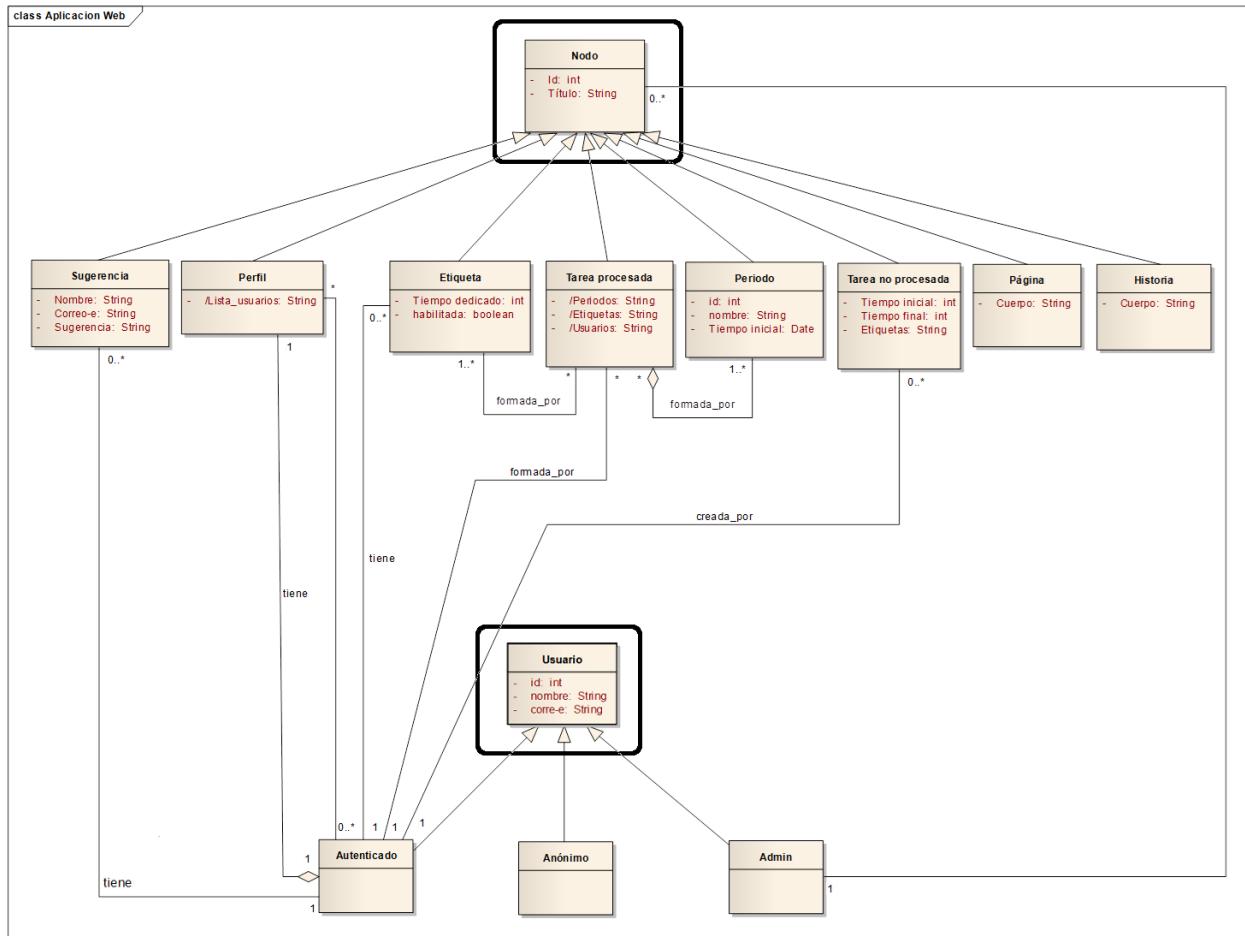


Ilustración 37: Modelo conceptual de la aplicación web

A continuación, se puede ver en detalle que las dos clases con borde, se encuentran inmersas detrás de un modelo base de **Drupal 6** mucho mayor. Obviamente el modelo iría aumentando en el caso en que se crearan nuevos tipos de contenido, roles, y otros elementos presentes en el diagrama presentado a continuación.

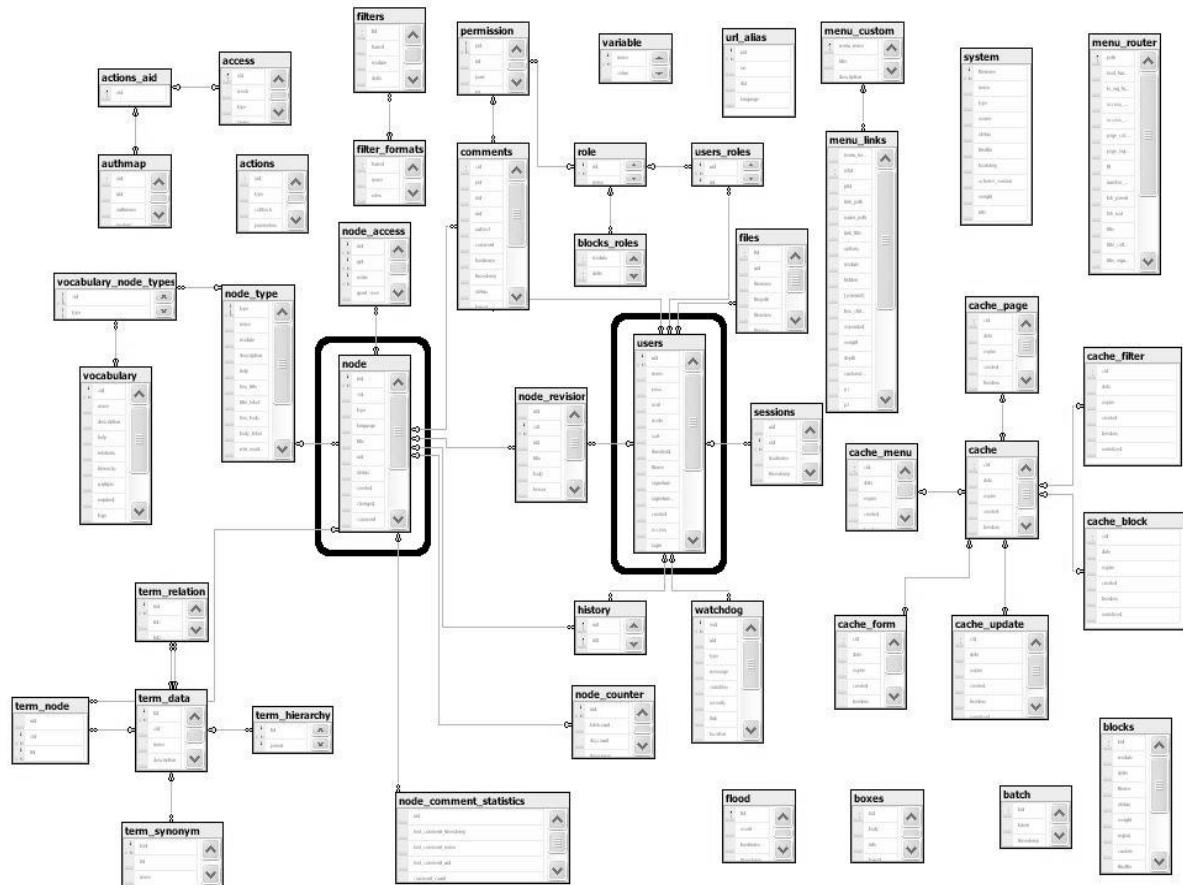


Ilustración 38: Parte del modelo conceptual de Drupal 6

3.3. Modelo del comportamiento

Los objetos del sistema se comunican mediante la invocación de operaciones entre otros objetos. Mediante la descripción del modelo de comportamiento para los sistemas, se pretende mostrar los diagramas de secuencia de dicho sistema, que muestran la secuencia de eventos entre los actores y el sistema, y permiten identificar las operaciones de este último. También se mostrarán los contratos de las operaciones del sistema, describiendo los efectos que puedan tener dichas operaciones, ya sean cambios de estado de la base de información y en qué condiciones se producen, así como las salidas que el sistema proporciona cuando se invoca la operación.

3.3.1. Aplicación móvil

Seguidamente se detallan cada uno de los diagramas de secuencia junto con los contratos de las operaciones, de los casos de uso para la aplicación móvil, descritos en el apartado anterior.

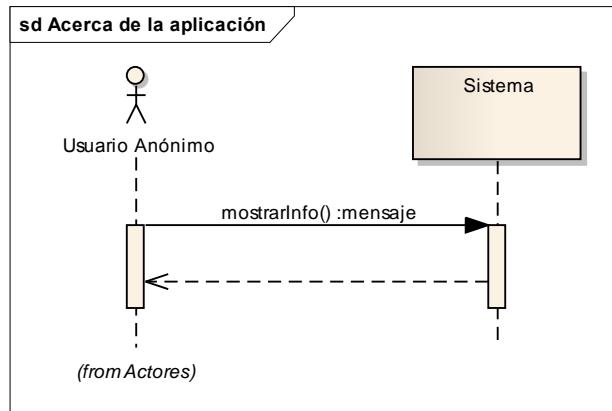


Ilustración 39: Diagrama de secuencia 1. Acerca de la aplicación

Acerca de la aplicación	
Precondiciones	-
Postcondiciones	-
Salida	Se muestra información referente a la autoría y agradecimientos.

Tabla 29: Contrato 1. Acerca de la aplicación

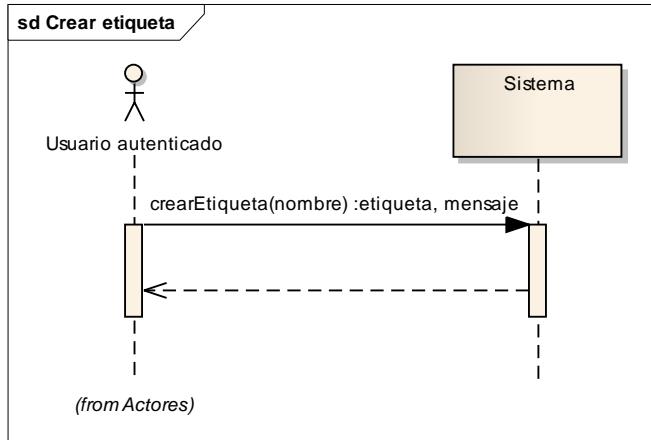


Ilustración 40: Diagrama de secuencia 2. Crear etiqueta

Crear etiqueta	
Precondiciones	-
Postcondiciones	Se ha creado una nueva instancia posee que asocia al usuario Autenticado y la nueva Etiqueta e con e.nombre=nombre, e.estado=1 y e.tiempo_dedicado = 0.
Salida	Se muestra mensaje de creación de etiqueta y se devuelve la nueva etiqueta.

Tabla 30: Contrato 2. Crear etiqueta

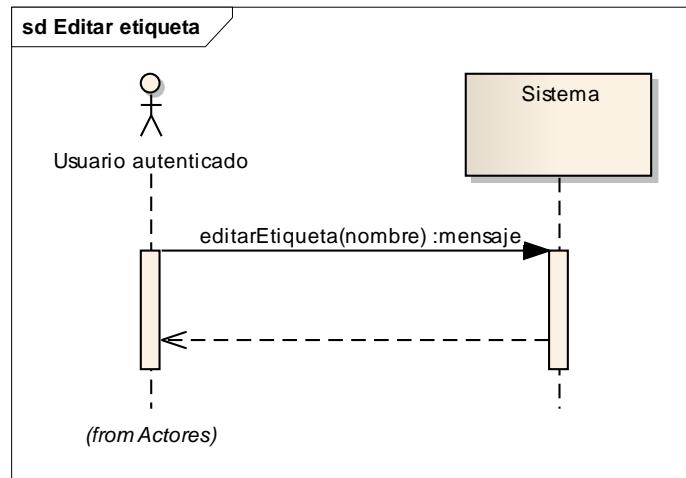


Ilustración 41: Diagrama de secuencia 3. Editar etiqueta

Editar etiqueta	
Precondiciones	Ha de existir alguna etiqueta.
Postcondiciones	La Etiqueta e ha sido modificada con el nuevo nombre, e.nombre=nombre.
Salida	Se muestra mensaje de edición de la etiqueta correspondiente.

Tabla 31: Contrato 3. Editar etiqueta

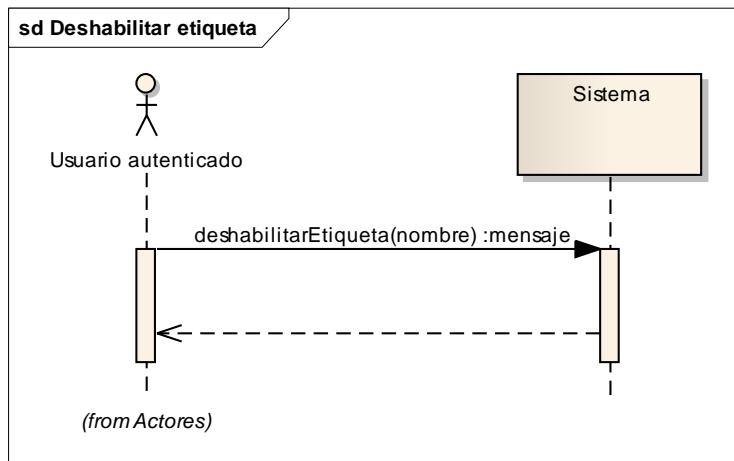


Ilustración 42: Diagrama de secuencia 4. Deshabilitar etiqueta

Deshabilitar etiqueta	
Precondiciones	Ha de existir alguna etiqueta.
Postcondiciones	La Etiqueta e ha sido modificada con el nuevo estado, e.estado=0.
Salida	Se muestra mensaje informando del estado actual de la etiqueta.

Tabla 32: Contrato 4. Deshabilitar etiqueta

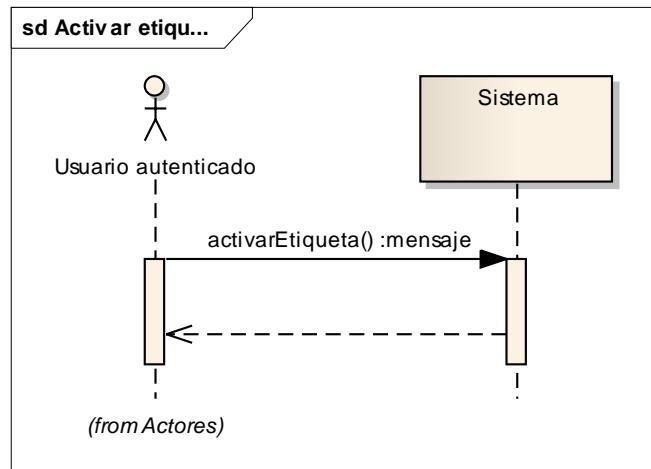


Ilustración 43: Diagrama de secuencia 5. Activar etiqueta

Activar etiqueta	
Precondiciones	Ha de existir alguna etiqueta.
Postcondiciones	La Etiqueta e ha pasado a tener instancia Activa .
Salida	Se muestra mensaje informando del estado actual de la etiqueta.

Tabla 33: Contrato 5. Activar etiqueta

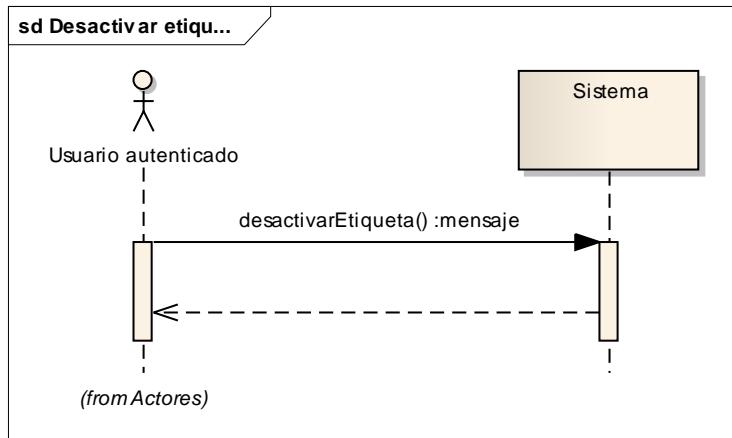


Ilustración 44: Diagrama de secuencia 6. Desactivar etiqueta

Desactivar etiqueta	
Precondiciones	- Ha de existir alguna Etiqueta. - La Etiqueta tiene que estar activada.
Postcondiciones	La Etiqueta e ha pasado a tener instancia Inactiva .
Salida	Se muestra mensaje informando del estado actual de la Etiqueta.

Tabla 34: Contrato 6. Desactivar etiqueta

3.3.2. Aplicación Web

Seguidamente se detallan cada uno de los diagramas de secuencia junto con los contratos de las operaciones de los casos de uso para la aplicación web, descritos en el apartado anterior.

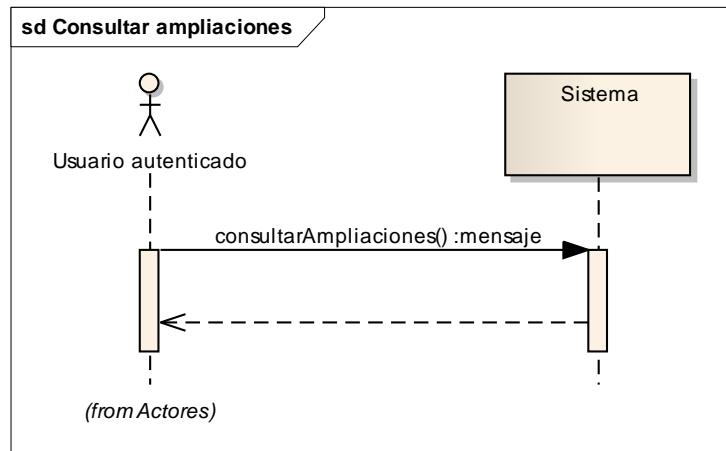


Ilustración 45: Diagrama de secuencia 7. Consultar ampliaciones

Consultar ampliaciones	
Precondiciones	-
Postcondiciones	-
Salida	Se muestra información referente a las próximas actualizaciones tanto de la aplicación móvil como la aplicación web, para The Time Bird .

Tabla 35: Contrato 7. Consultar ampliaciones

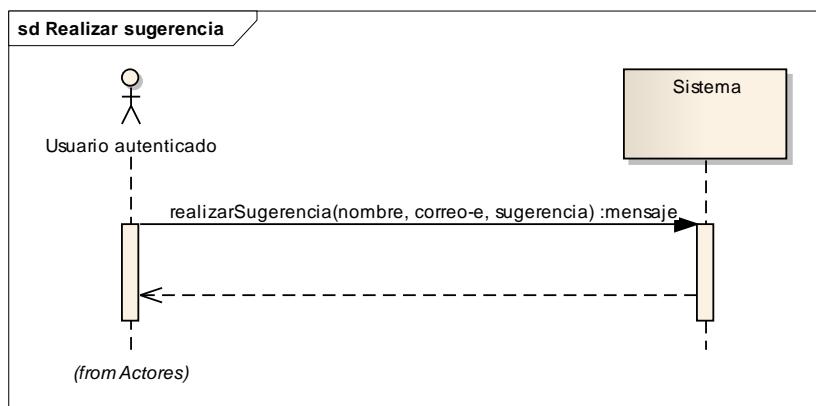


Ilustración 46: Diagrama de secuencia 8. Realizar sugerencia

Realizar sugerencia	
Precondiciones	-

Postcondiciones	Se ha creado una nueva instancia tiene que asocia al usuario Autenticado y a la nueva Sugerencia s , con s.nombre=nombre, s.email=email y s.sugerencia=sugerencia.
Salida	Se muestra un mensaje de confirmación de creación de la sugerencia.

Tabla 36: Contrato 8. Realizar sugerencia

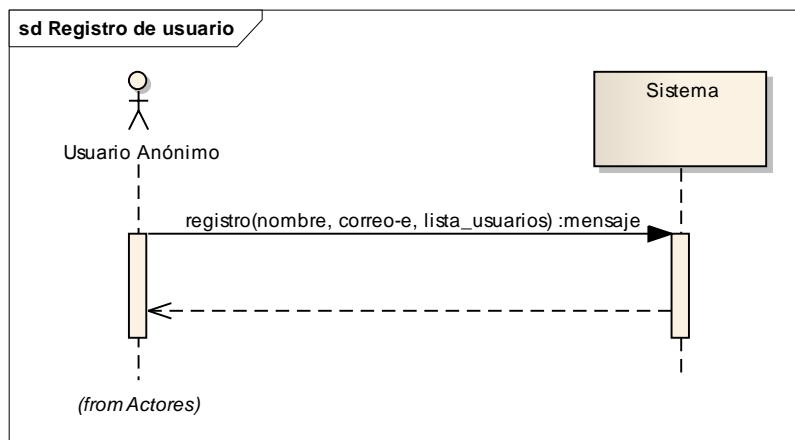


Ilustración 47: Diagrama de secuencia 9. Registro de usuario

Registro de usuario	
Precondiciones	-
Postcondiciones	Se ha creado un nuevo usuario Autenticado u con u.nombre=nombre y u.email = correo-e, así como una instancia tiene que asocia al usuario u y a un Perfil p , así como tantas instancias de usuarios Autenticados seleccionados por el usuario.
Salida	Se muestra un mensaje de confirmación de creación del usuario.

Tabla 37: Contrato 9. Registro de usuario

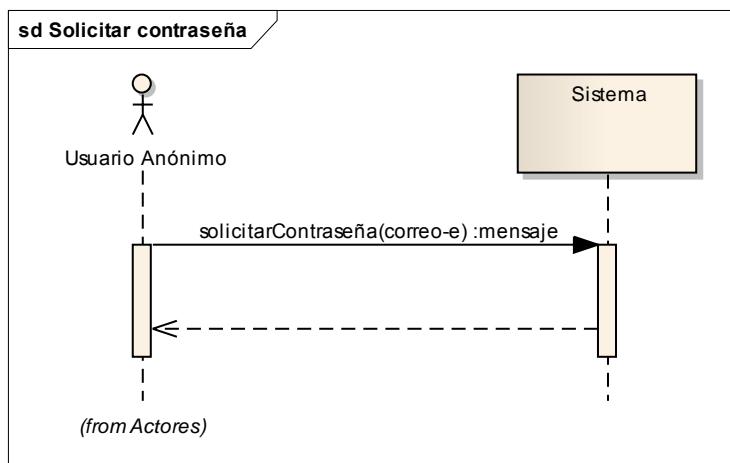


Ilustración 48: Diagrama de secuencia 10. Solicitud de contraseña

Solicitud de contraseña	
Precondiciones	-
Postcondiciones	-
Salida	Se muestra un mensaje de confirmación de envío de correo al usuario de una nueva contraseña.

Tabla 38: Contrato 10. Solicitud de contraseña

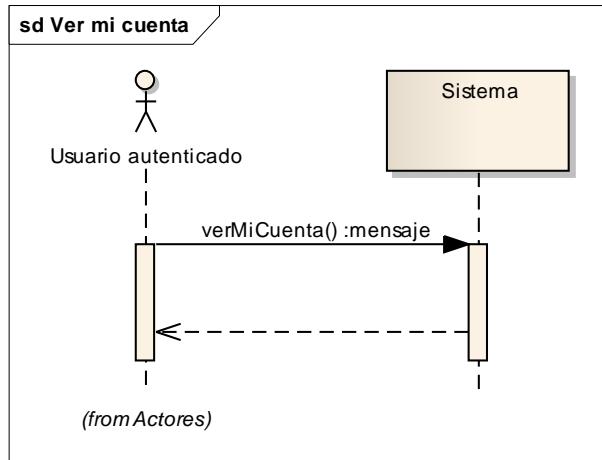


Ilustración 49: Diagrama de secuencia 11. Ver mi cuenta

Ver mi cuenta	
Precondiciones	-
Postcondiciones	-
Salida	Se muestra información relacionada con los datos del Usuario (nombre, hora de creación del usuario y lista de usuarios a los que les está permitido ver nuestras estadísticas).

Tabla 39: Contrato 11. Ver mi cuenta

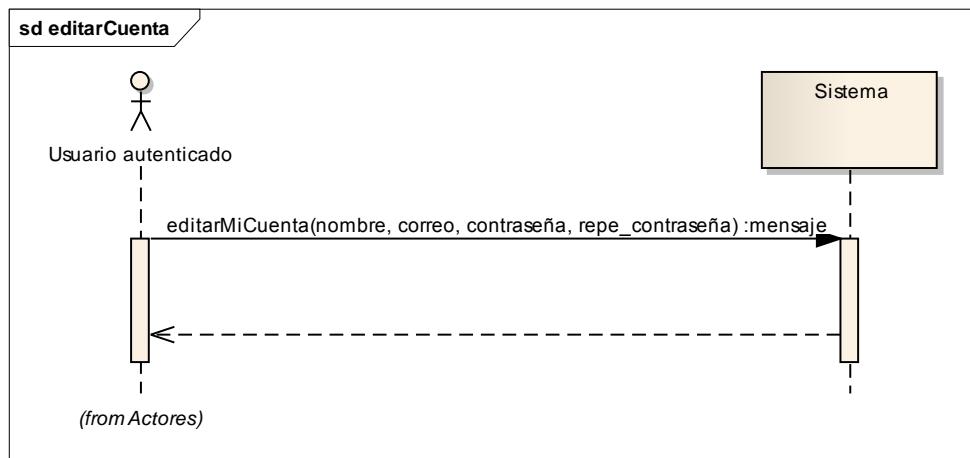


Ilustración 50: Diagrama de secuencia 12. Editar mi cuenta

Editar mi cuenta	
Precondiciones	-
Postcondiciones	El Usuario u ha sido modificado con un nuevo nombre, correo electrónico y contraseña, donde u.nombre = nombre, u.mail = correo y u.contraseña = contraseña.
Salida	Se muestra un mensaje de confirmación de los cambios realizados.

Tabla 40: Contrato 12. Editar mi cuenta

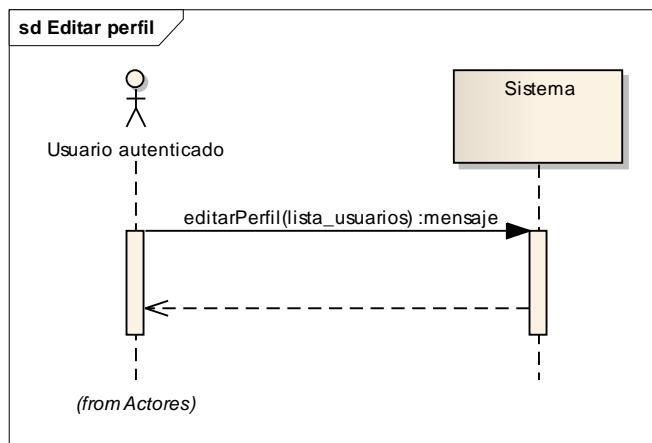


Ilustración 51: Diagrama de secuencia 13. Editar perfil

Editar perfil	
Precondiciones	-
Postcondiciones	El Perfil p ha sido modificado con la nueva lista de usuarios a los que se da permiso para ver nuestros datos, siendo así p.lista_usuarios = lista_usuarios.
Salida	Se muestra un mensaje de confirmación de los cambios realizados.

Tabla 41: Contrato 13. Editar perfil

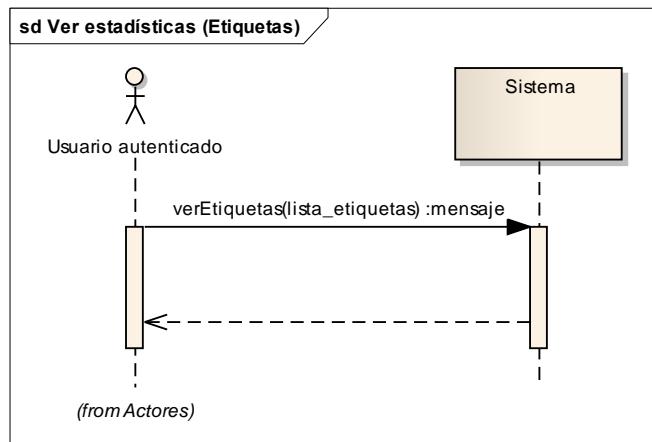


Ilustración 52: Diagrama de secuencia 14. Ver estadísticas (Etiquetas)

Ver estadísticas (Etiquetas)	
Precondiciones	-
Postcondiciones	-
Salida	Se muestran las estadísticas para las etiquetas seleccionadas, mediante gráficos y mensajes informativos.

Tabla 42: Contrato 14. Ver estadísticas (Etiquetas)

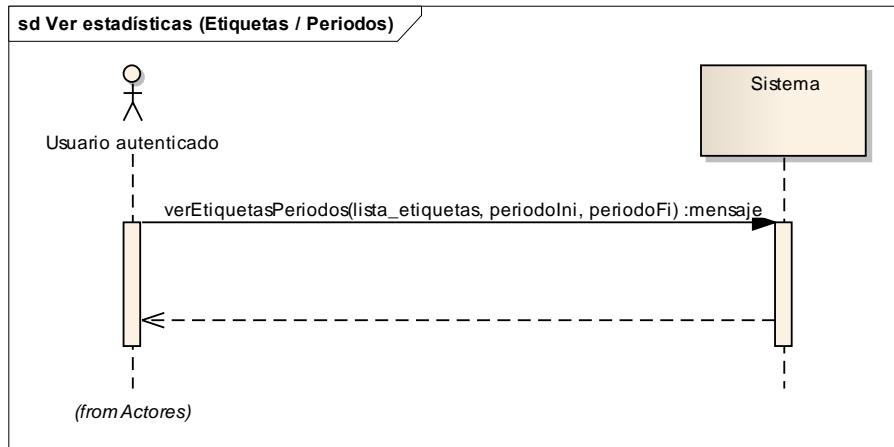


Ilustración 53: Diagrama de secuencia 15. Ver estadísticas (Etiquetas/Periodos)

Ver estadísticas (Etiquetas / Periodos)	
Precondiciones	-
Postcondiciones	-
Salida	Se muestran las estadísticas para las etiquetas y periodos seleccionados, mediante gráficos y mensajes informativos.

Tabla 43: Contrato 15. Ver estadísticas (Etiquetas/Periodos)

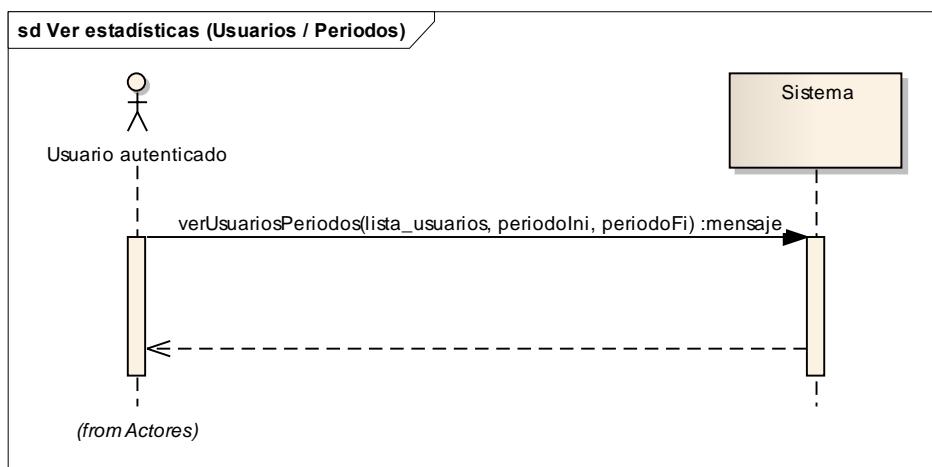


Ilustración 54: Diagrama de secuencia 16. Ver estadísticas (Usuarios/Periodos)

Ver estadísticas (Usuarios / Periodos)	
Precondiciones	-
Postcondiciones	-
Salida	Se muestran las estadísticas para los usuarios y periodos seleccionados, mediante gráficos y mensajes informativos.

Tabla 44: Contrato 16. Ver estadísticas (Usuarios/Periodos)

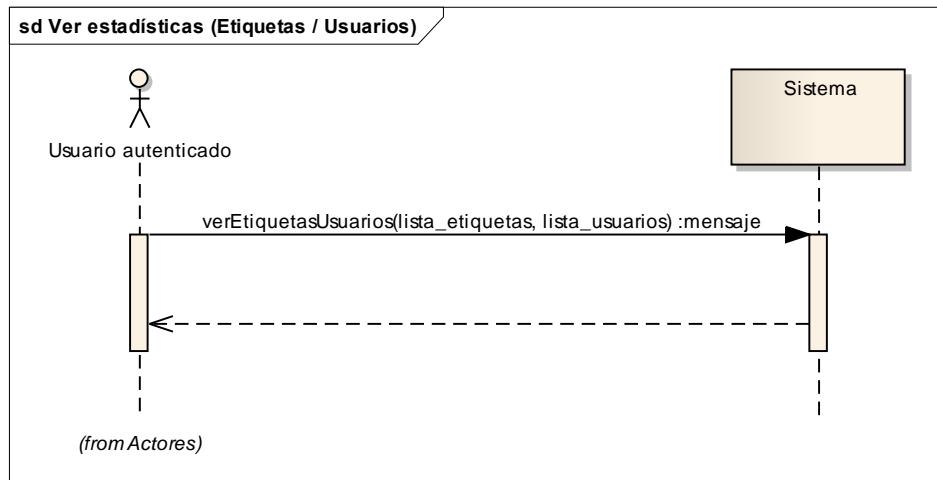


Ilustración 55: Diagrama de secuencia 17. Ver estadísticas (Etiquetas/Usuarios)

Ver estadísticas (Etiquetas / Usuarios)	
Precondiciones	-
Postcondiciones	-
Salida	Se muestran las estadísticas para los usuarios y las etiquetas seleccionadas, mediante gráficos y mensajes informativos.

Tabla 45: Contrato 17. Ver estadísticas (Etiquetas/Usuarios)

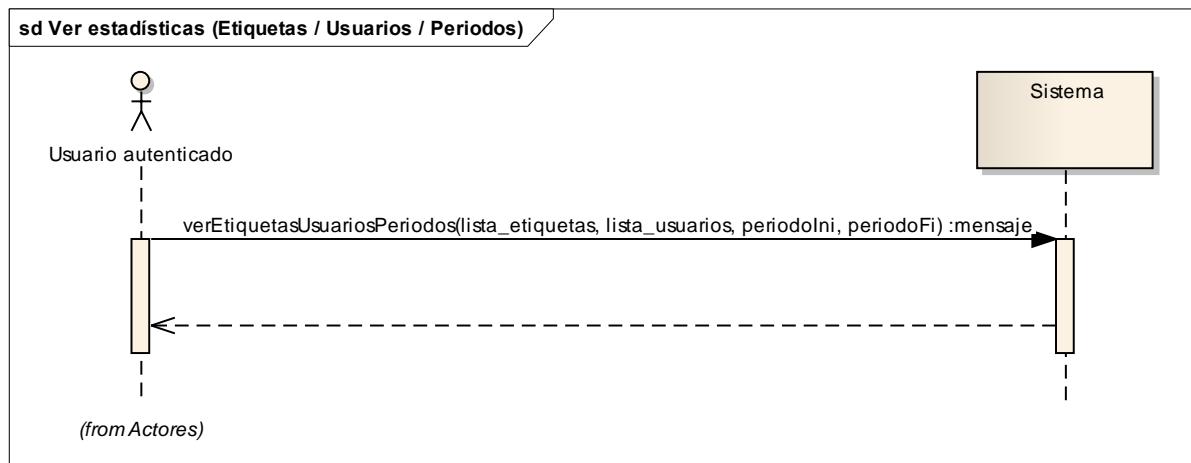


Ilustración 56: Diagrama de secuencia 18. Ver estadísticas (Etiquetas/Usuarios/Periodos)

Ver estadísticas (Etiquetas / Usuarios / Periodos)	
Precondiciones	-
Postcondiciones	-
Salida	Se muestran las estadísticas para los usuarios, las etiquetas y los periodos seleccionados, mediante gráficos y mensajes informativos.

Tabla 46: Contrato 18. Ver estadísticas (Etiquetas/Usuarios/Periodos)

4. Diseño

La fase siguiente a la especificación, en un proyecto de desarrollo *software*, es la fase de diseño, donde se va a estudiar y decidir cómo va a funcionar el sistema, basándonos en su arquitectura, siempre considerando todos los requerimientos no funcionales, con el propósito de definir un sistema con el suficiente detalle como para permitir su construcción física (implementación).

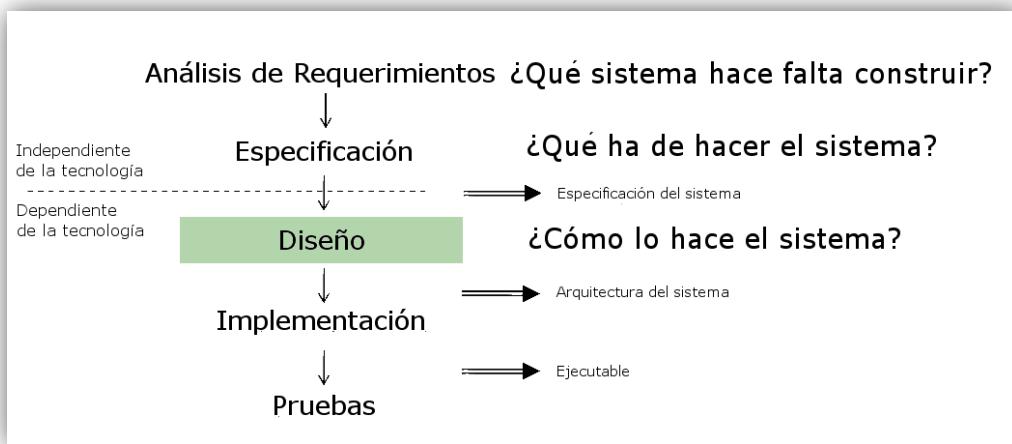


Ilustración 57: Etapas de desarrollo software (Diseño)

Los objetivos principales del diseño son:

- Construir el sistema en base al lenguaje de programación utilizado y/o plataforma.
- Conocer y analizar en profundidad los requerimientos no funcionales del sistema para poder así definir la relación entre cada uno de los componentes que forman el sistema y obtener una arquitectura de calidad acorde a los requerimientos establecidos.
- Descomponer la fase de implementación en varias partes facilitando así su posible gestión en equipos de desarrollo diferentes.
- Crear una abstracción del desarrollo de todos los componentes del sistema. Esto permitirá en un futuro la posibilidad de reingeniería inversa entre el diseño del sistema y su implementación.

En primer lugar se procederá a describir la **arquitectura física** de los sistemas para que éstos se adecuen de la mejor forma posible a las funcionalidades especificadas en el capítulo anterior. A continuación se describirán las **tecnologías** utilizadas que estarán estrechamente ligadas al diseño tanto de la aplicación web como de la aplicación móvil. Finalmente se realizará una descripción detallada de la **arquitectura lógica** del sistema global, de tal forma que se puedan mostrar los diferentes componentes *software*

existentes en el sistema.

4.1. Arquitectura física

Siendo algunos de los requerimientos no funcionales, la utilización de **tecnologías móviles**, así como la **utilización del gestor de contenidos Drupal**, no es de extrañar la decisión de utilizar como arquitectura básica del sistema el patrón Cliente-Servidor, que se justifica por las siguientes razones:

- **Escalabilidad:** La posibilidad de aumentar la capacidad de clientes y servidores por separado es una gran ventaja, ya que la idea para ***The Time Bird***, es exportarlo en un futuro a otras plataformas. Cualquier elemento puede ser aumentado o mejorado o se pueden añadir más nodos, ya sean clientes o servidores, a la red. Por un lado cuantos más clientes haya, más facilidades le damos al usuario para que use la aplicación. Esto supone que dependiendo de la carga que tenga el sistema, es posible que en un futuro se realice un balanceo de carga mediante varios servidores, o tomar las medidas que sean necesarias.
- La **centralización** de datos y la existencia de una **única lógica de negocio** en el servidor, hará que no sea necesaria la replicación de datos en cada una de las plataformas cliente. También permitirá fácilmente la realización de **cambios y actualizaciones** de los datos.
- La existencia y actualización de las tecnologías existentes aseguran cierto nivel de **seguridad** en las transacciones de datos, la **amigabilidad** de la interfaz propia de cada plataforma, así como la facilidad de uso.

El sistema global está compuesto por programas clientes (en este caso tanto la aplicación móvil como la aplicación web), que realizan una serie de peticiones a un servidor. Este último es el elemento central de la aplicación, ya que incorpora toda la lógica de negocio y los datos. Los clientes, realizarán ciertas tareas. Podemos dividir los clientes en dos sistemas; la aplicación móvil al realizar solo transacciones a la aplicación web, será considerado un **sistema transaccional**, en cambio la aplicación web, se encargara de transformar los datos enviados y mostrárselos al usuario final mediante gráficos y estadísticas, por lo que será un **sistema analítico**.

A continuación se resumen brevemente las tecnologías utilizadas (explicadas de forma detallada en el siguiente apartado) en cada sistema:

- La aplicación web estará desarrollada con Drupal, actuará tanto de cliente como servidor (más adelante se verá detallada su arquitectura).
- La aplicación móvil estará desarrollada con Android y actuará como cliente.

- Los comunicaciones se realizarán mediante datos en formato XML.

La aplicación web estará alojada en un servidor, que tanto durante la fase de desarrollo de **The Time Bird** como su puesta en marcha, será siempre gestionado por la empresa, y con ubicación externa a ésta, para así poder tener una completa disponibilidad de sus servicios y así poder trabajar con él cuando sea conveniente.

A continuación se puede observar de forma esquemática como está construido el sistema global.

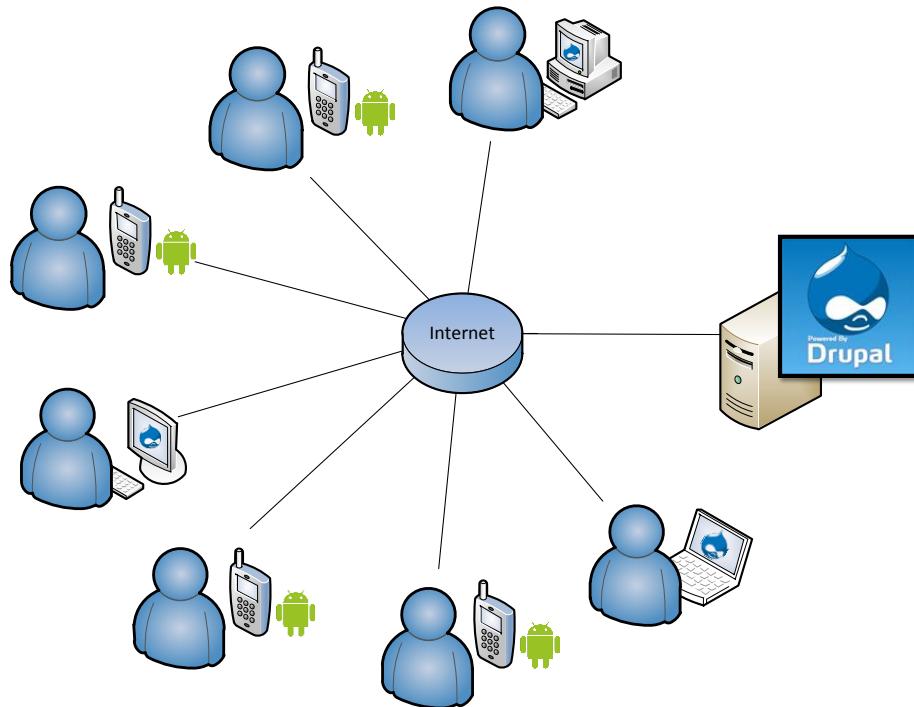


Ilustración 58: Arquitectura básica de los sistemas con varios clientes conectados simultáneamente.

4.2. Tecnologías utilizadas

Se va a proceder a definir las tecnologías escogidas en el capítulo de análisis inicial, usadas tanto en la aplicación web, aplicación móvil y comunicación de datos. Todas las tecnologías descritas afectarán directamente al diseño de las aplicaciones.

4.2.1. Drupal

Drupal es un sistema de gestión de contenidos, modular y multipropósito y muy configurable, que permite publicar artículos, imágenes, u otros archivos y añadir servicios como foros, encuestas, votaciones, *blogs* y administración de usuarios y permisos. Drupal es un sistema dinámico: en lugar de almacenar sus contenidos en archivos estáticos en el

sistema de ficheros del servidor de forma fija, el contenido textual de las páginas y otras configuraciones son almacenadas en una base de datos y se editan utilizando un entorno web.

Es un programa con licencia *GPL* de *GNU* y destacado por la calidad de su código respecto a estándares web, así como por su usabilidad y la consistencia de todo el sistema.

Los objetivos de diseño de Drupal son; la capacidad de funcionar bien en alojamientos web de bajo coste, y tener escalabilidad en sitios masivos distribuidos. El primero de los objetivos conlleva el uso de tecnología popular, y el segundo conlleva el uso de código ligero. A continuación se puede observar la pila tecnológica de Drupal.

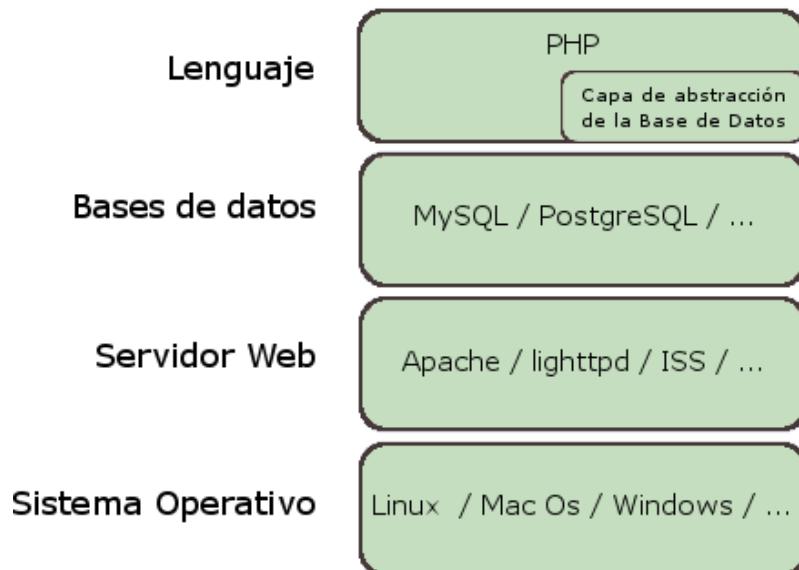


Ilustración 59: Pila tecnológica de Drupal

El sistema operativo está a un nivel tan bajo en la pila, que para Drupal no es ningún impedimento. Éste es capaz de ejecutarse correctamente en cualquier sistema operativo que soporte *PHP*.

El servidor web más utilizado con Drupal es Apache, aunque existen otros servidores web (incluido IIS de Microsoft) que pueden ser utilizados. Debido a la larga historia entre ambos, Drupal viene con un fichero *.htaccess* que protege su la instalación. Las *URLs* limpias²³ se consiguen gracias al módulo de Apache **mod_rewrite**. Este aspecto es muy importante ya que cuando hay migración de contenido en un sistema de estas características, las *URLs* necesitan estar de la misma forma, sin ningún cambio. Las *URLs* limpias están disponibles en otros servidores web con capacidad de poder ser reescritas.

Drupal interactúa con la siguiente capa de la pila (las bases de datos) a través de una

²³ URLs sin signos de interrogación, símbolos de unión u otros caracteres extraños

ligera capa de abstracción. Esta capa se encarga de sanear las consultas de *SQL* y hacer posible el uso de bases de datos de diferentes proveedores sin reescribir código. Las bases de datos más probadas son MySQL²⁴ y PostgreSQL²⁵.

Drupal está escrito en *PHP*, y aunque la calidad de código de un principiante en este idioma suele dar mala reputación, en *PHP* se puede escribir código sólido. Todo el código de su núcleo se adhiere a estrictas normas de codificación [14] y se somete a continuo examen a través de los procesos de código abierto. Para Drupal, la fácil curva de aprendizaje de *PHP* significa que hay una baja barrera de entrada a contribuyentes que están empezando, y la revisión de este proceso asegura la calidad del producto final. Además, y como se ha comentado en el capítulo del análisis inicial, gracias a la retroalimentación de la gran comunidad de desarrolladores, se va proporcionando ayuda para mejorar las habilidades de la gente de la comunidad.

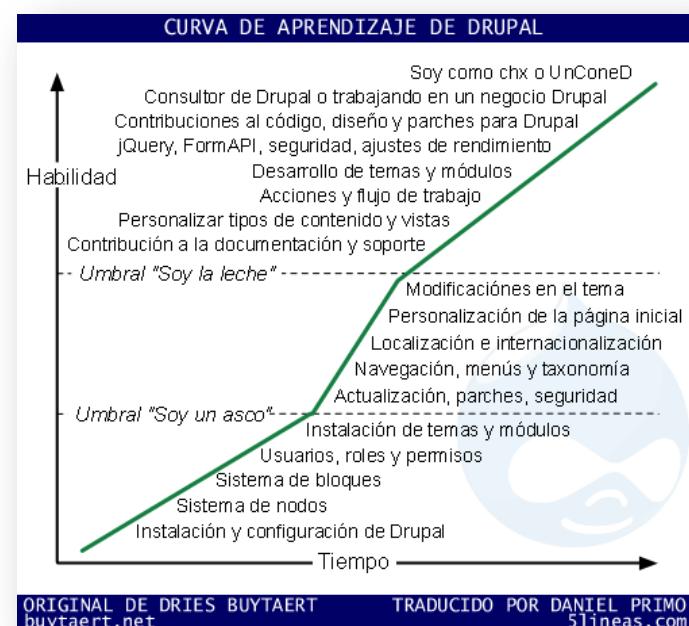


Ilustración 60: Curva de aprendizaje de Drupal

Cabe comentar que en la capa superior, Drupal produce contenido para las páginas en general en formato *XHTML*, aunque otros tipos de formato son compatibles. Los *CSS* se utilizan para controlar el diseño, colores y fuentes de una página determinada, y el

²⁴ MySQL es un sistema de gestión de bases de datos relacional, multi-hilo y multiusuario.

²⁵ PostgreSQL es un sistema de gestión de bases de datos relacional y orientado a objetos.

Javascript se lanza en elementos dinámicos, como capas desplegables, efectos *drag&drop* sobre ciertos elementos, o como en nuestro caso, todas las gráficas estadísticas.

También es necesario conocer los componentes más importantes de Drupal (y que a su vez formarán parte de la arquitectura lógica del sistema global), presentados a continuación.

Su núcleo

Su núcleo está compuesto por un *framework* ligero, que es responsable de proporcionar las funcionalidades básicas que darán soporte a las otras partes del sistema, como una biblioteca de funciones comunes utilizada por Drupal, y módulos que proveen funcionalidades básicas como gestión de usuarios, taxonomías y plantillas, que se muestran a continuación. Además su código permite hacer *bootstrap*²⁶ cuando recibe una petición,.



Ilustración 61: Algunas de las funcionalidades del núcleo de Drupal

Interfaz de Administración

La interfaz administrativa de Drupal está integrada con el resto del sitio web, y por defecto, los usuarios usan el mismo tema visual.

Módulos

Drupal es verdaderamente un *framework* modular. Los módulos son aplicaciones existentes o desarrolladas por miembros de la comunidad, con la finalidad de añadir nuevas funcionalidades o ampliar las ya existentes para conseguir un nuevo comportamiento deseado. Se pueden añadir tantos módulos como se desee.

²⁶ Hace referencia al proceso donde un simple sistema activa a otro más complejo para servir al mismo propósito. Es una forma de comenzar un cierto sistema sin tener el sistema todavía funcionando.

En la siguiente ilustración se puede observar un ejemplo en el que se añaden módulos con sus respectivas funcionalidades.



Ilustración 62: Activación de módulos que ofrecen nuevas funcionalidades

Hooks²⁷

Los *hooks* son considerados eventos internos de Drupal. También son llamados *callbacks²⁸*, aunque están construidos por convenciones de nombres y no por registro de escucha (más conocidos como *listeners*) que se ejecutan cuando se produce un evento que así lo indica (un clic, una fracción de tiempo, etc.).

Los *hooks* permiten a los módulos saber que está pasando en el resto de Drupal. Para entenderlo vamos a poner un ejemplo; supongamos que un usuario se registra en la aplicación web. En el momento en el que el usuario inicia una sesión, Drupal lanza el *hook* de usuario. Esto significa que cualquier función denominada según el nombre convencional que corresponderá al nombre del módulo (*hook_modulo*) será llamada. Por ejemplo, *comment_user()* en el módulo de comentarios, o *locale_user()* en el módulo local y así sucesivamente.

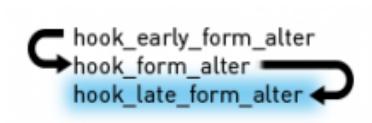


Ilustración 63: Ejemplo de ganchos que son llamados por otros ganchos

²⁷ Ganchos es su traducción, aunque es más utilizada la terminología *hook* [15].

²⁸ Es una función que recibe como argumento la dirección o puntero de otra función, cuando la retrollamada es llamada esta recurre al punto de la función y la ejecuta. Esto permite desarrollar capas de abstracción de código genérico a bajo nivel que pueden ser llamadas desde una sub-rutina (o función) definida en una capa de mayor nivel.

Temas

Cuando se crea una página web y se envía al navegador, existen dos preocupaciones básicas: el ensamblaje de los datos adecuados y el “maquillaje” que se le da a estos datos. En Drupal, la capa de tema es la responsable de crear el HTML (o JSON, XML, etc.) que recibirá el navegador. Puede usar diversos métodos populares para hacer plantillas, como *Smarty*²⁹ y *PHPTemplate*³⁰. A mencionar que Drupal fomenta la separación de contenido y maquillaje, permitiendo así el hoja de estilos CSS para sobrescribir algunas de sus clases e identificadores.

Nodos

El diferente tipo de contenido en Drupal es el derivado de un tipo básico llamado nodo. Ya sean páginas, entradas de *blog*, noticias, etc. Su estructura de datos subyacente es la misma gracias a su **extensibilidad**. Además se pueden añadir características, ya sean calificaciones, fechas, comentarios, archivos, y así sucesivamente, sea del tipo de contenido que sea. Los nodos también contienen un conjunto básico de propiedades que heredan todos los tipos de contenido. Cualquier nodo puede ser promovido a la página principal del sitio web, publicado o no publicado, o incluso se da la posibilidad de indexar o esconder contenido.

Bloques y menús

Un bloque y un menú son información que puede estar activada o desactivar en un lugar determinado en el sitio web. El contenido a mostrar en un bloque o un menú es totalmente personalizable y son colocados generalmente en los lados laterales, encabezados o pies de página. A menudo se utilizan para presentar información personalizada para el usuario actual, ya sea enlaces a áreas administrativas a las que el usuario tiene acceso, sesión, etc.

Permisos de usuario

Cualquier aplicación web tendría que poder diferenciar los distintos usuarios para permitir el acceso a unas zonas u otras de la web (en el caso que fuese necesario). El control de los

²⁹ *Smarty* es un motor de plantillas de código libre para PHP que lleva muchos años en el mercado. Con él podremos realizar aplicaciones web de calidad separando el código (PHP) de la presentación (HTML/CSS).

³⁰ *PHPTemplate* es un motor de temas que permite utilizar archivos de plantillas en PHP. Estos archivos no tienen que ser procesados por el motor del tema, para así poder ser procesados con mayor velocidad que otros motores de plantillas

usuarios, así como la seguridad, son ambas, clave para garantizar la integridad de los datos. Drupal no es una excepción, y dispone de un sistema de registro de usuarios y asignación de roles que permitirá a cada uno de ellos, realizar cualquiera de las tareas a las que sean asignadas, así como restricción en el acceso a ciertos tipos de datos.

Cron

Es la herramienta que tiene Drupal para ejecutar tareas periódicamente, y que se usa frecuentemente para la indexación de contenidos, tareas de limpieza, que consuman muchos recursos al ejecutarse de una sola vez o que dependan del tiempo, ya sean alarmas, encuestas, etc.

4.2.2. Android

Android es un sistema operativo basado en Linux y dedicado mayoritariamente a dispositivos móviles y *tablets*. Fue desarrollado inicialmente por Android Inc., una firma comprada por Google en 2005. Es el principal producto de la *Open Handset Alliance*, un conglomerado de fabricantes y desarrolladores de *hardware*, *software* y operadores de servicio.

Android tiene una gran comunidad de desarrolladores escribiendo aplicaciones para extender la funcionalidad de los dispositivos. Además Google posee un mercado llamado *Android Market*, donde se pueden colgar todas las aplicaciones desarrolladas, escritas en lenguaje de programación Java.



Ilustración 64: Logotipo del sistema operativo Android

El anuncio del sistema Android se realizó el 5 de Noviembre de 2007 junto con la creación de la *Open Handset Alliance*, un consorcio de 78 compañías de *hardware*, *software* y telecomunicaciones dedicadas al desarrollo de estándares abiertos para dispositivos móviles. Google liberó la mayoría del código de Android bajo la licencia de Apache, una licencia libre y de código abierto.

La estructura del sistema operativo Android se compone de aplicaciones que se ejecutan en un *framework Java* de aplicaciones orientadas a objetos sobre el núcleo de las bibliotecas de Java en una máquina virtual Dalvik con compilación en tiempo de ejecución. Las bibliotecas escritas en lenguaje C incluyen un administrador de interfaz gráfica (*surface manager*), un *framework* OpenCore, una base de datos relacional *SQLite*³¹, una *API* gráfica *OpenGL*³² ES 2.0 3D, un motor de renderizado WebKit, un motor gráfico *SGL*, *SSL*, y una biblioteca estándar de C. El sistema operativo está compuesto por 12 millones de líneas de código, incluyendo 3 millones de líneas de *XML*, 2.8 millones de líneas de lenguaje C, 2.1 millones de líneas de Java y 1.75 millones de líneas de C++.

Para describir un poco mejor el sistema, vamos a proceder a mencionar las partes, divididas en 4 capas, por los cuales se compone Android.

- **Núcleo:** Android depende de Linux para los servicios base del sistema así como seguridad, gestión de procesos, pila de red y modelo de controladores de *hardware* de un terminal (cámara, teclado, audio, etc.). El núcleo también actúa como una capa de abstracción entre el hardware y el resto de la pila *software*.
- **Bibliotecas:** Android incluye un conjunto de bibliotecas de C/C++ usadas por varios componentes del sistema expuestas para su uso por los desarrolladores, así como un conjunto de bibliotecas base que proporcionan la mayor parte de las funciones disponibles en las bibliotecas base del lenguaje Java. Cada aplicación Android corre su propio proceso, con su propia instancia de la máquina virtual Dalvik. Su función es de hacer de capa de abstracción entre las capacidades de cada terminal y las aplicaciones que pueda crear un desarrollador, ya sea por el uso de la cámara, el GPS, etc.
- **Marco de trabajo de aplicaciones:** Los desarrolladores tienen acceso completo al *SDK* de Android para poder facilitar la creación de aplicaciones. Los componentes básicos que podemos encontrar son los siguientes:
 - **Actividades:** Componentes de interfaz que normalmente corresponden a las diferentes pantallas de una aplicación. Su ciclo de vida es parecido al de las páginas web, permitiéndonos el movimiento a la siguiente pantalla (empezando una nueva Actividad) o a la anterior (volviendo a la Actividad previa).
 - **Servicios:** Componentes sin interfaz, que permiten la ejecución de tareas en *background* durante un largo periodo de tiempo, como por ejemplo, una tarea en la que se descargan datos periódicamente.

³¹ *SQLite* es un sistema de gestión de bases de datos relacional contenida en una relativamente pequeña biblioteca en C.

³² Es una especificación estándar que define una *API* para desarrollar aplicaciones que produzcan gráficos 2D y 3D.

- **Proveedores de contenido:** Componentes con el fin de ofrecer una vía para compartir datos entre aplicaciones. Los datos pueden estar guardados en una base de datos o en ficheros.
- **Receptores de transmisiones:** Tienen la tarea de responder a eventos externos y puede hacer despertar a los procesos que deseas de tu aplicación, como por ejemplo, la actualización de correo o la actualización de datos de un servicio contratado, como Twitter.
- **Aplicaciones:** La capa de aplicaciones, es el entorno de instalación de las diferentes aplicaciones en Android. Por defecto existen aplicaciones instaladas por defecto en un terminal, y su cantidad dependerá de la compañía fabricante del teléfono.

Para el desarrollo de la aplicación Android **The Time Bird**, se ha tenido que pasar por todas las capas mencionadas.

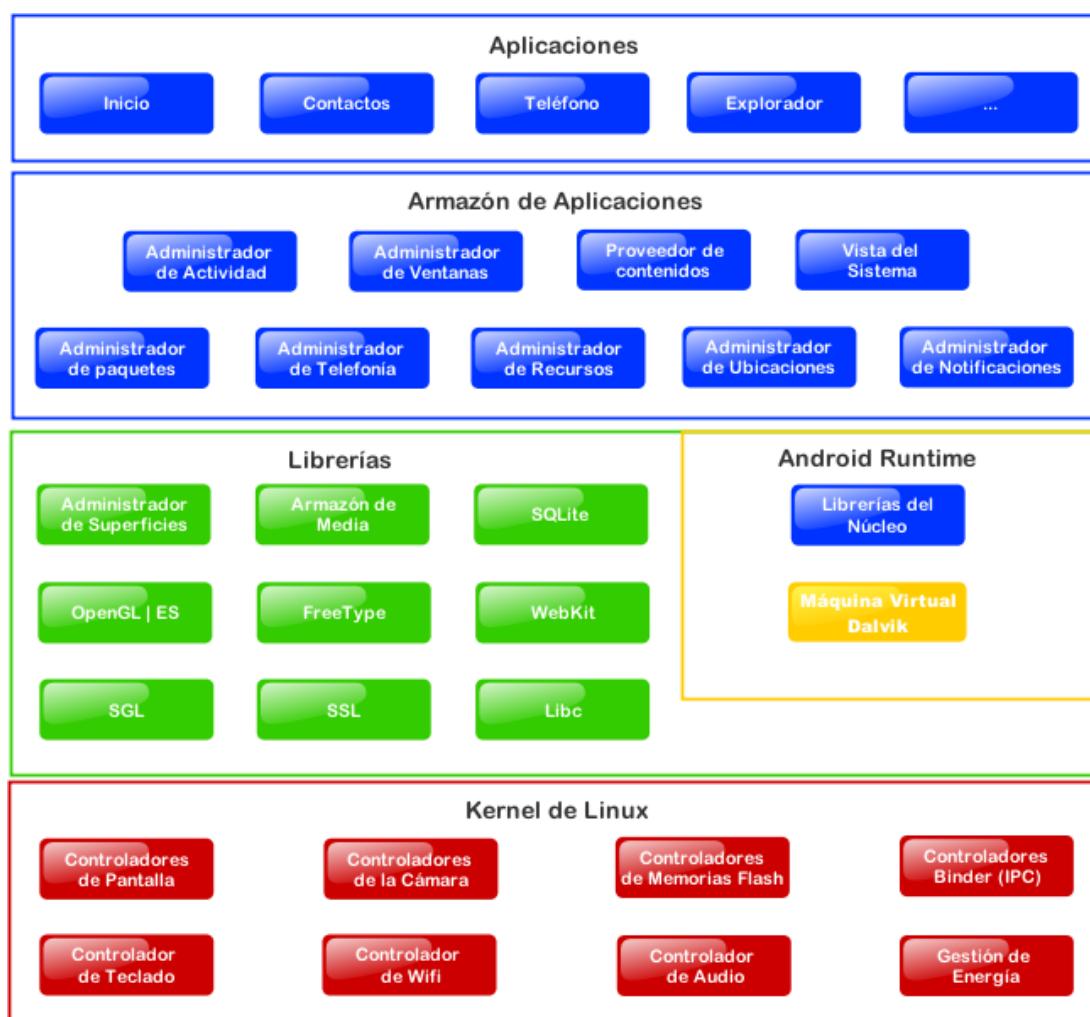


Ilustración 65: Componentes principales de Android

Ciclo de vida de sus componentes

Ya sean las Actividades, proveedores de contenido o receptores de transmisiones, todos ellos tienen un tiempo de vida, desde que son iniciados hasta el momento que son destruidos, aunque durante la vida de cada uno de estos componentes, pasan por una serie de estados.

En cada uno de estos estados, son ejecutados diferentes métodos, para así poder llevar a cabo las tareas que sean necesarias.

Como ejemplo, mostramos el ciclo de vida de las Actividades, dividido en cuatro posibles estados:

- Activa (*Running*): La Actividad está encima de la pila. Actualmente es visible.
- Pausada (*Paused*): La Actividad es visible pero no tiene el foco. Se alcanza este estado cuando pasa a activa otra Actividad transparente o que no ocupa toda la pantalla. Cuando una Actividad es tapada por completo pasa a estar parada.
- Parada (*Stopped*): Cuando la Actividad no es visible.
- Destruida (*Destroyed*): Cuando la Actividad termina.

Los métodos utilizados en las distintas transiciones de estados son:

- *onCreate(Bundle)*: Se invoca cuando la Actividad se inicia por primera vez.
- *onStart()*: Se invoca cuando la Actividad va a ser mostrada al usuario.
- *onResume()*: Se invoca cuando la Actividad va a empezar a interactuar con el usuario.
- *onPause()*: Se invoca cuando la Actividad va a pasar a *background* porque otra Actividad ha sido lanzada.
- *onStop()*: Se invoca cuando la Actividad va a dejar de ser visible y no se necesitará durante un tiempo.
- *onRestart()*: Se invoca cuando una Actividad parada pasa a estar activa.
- *onDestroy()*: Se invoca cuando la Actividad va a ser destruida.
- *onSaveInstanceState(Bundle)*: Se invoca para permitir a la Actividad guardar su estado.
- *onRestoreInstanceState(Bundle)*: Se invoca para recuperar el estado guardado por *onSaveInstanceState()*;

A continuación se muestra de forma gráfica, todas las asociaciones entre las distintas transiciones del ciclo de vida de un componente.

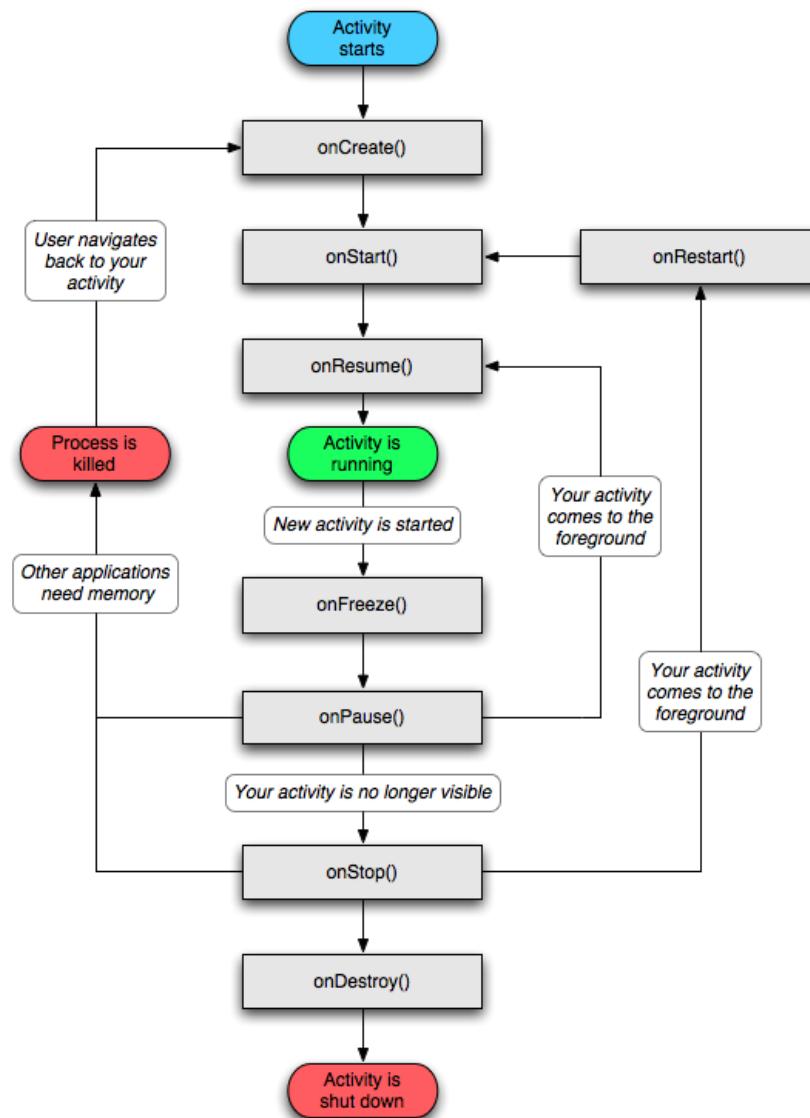


Ilustración 66: Ciclo de vida de una Actividad en Android

Algunas características destacadas

A continuación podemos ver algunas características y especificaciones de Android, que son y podrían ser de relevancia para **The Time Bird**.

Conectividad	Android soporta las siguientes tecnologías de conectividad: <i>GSM/EDGE, IDEN, CDMA, EV-DO, UMTS, Bluetooth, Wi-Fi, LTE, WiMAX.</i>
Navegador web	El navegador web incluido en Android está basado en el motor de renderizado de código abierto <i>WebKit</i> , emparejado con el motor <i>JavaScript V8</i> de Google Chrome. El navegador obtiene una puntuación de 93/100 en el test <i>Acid3</i> .
Soporte de Java	Aunque las aplicaciones están escritas en Java, no hay una Máquina

	Virtual de Java en la plataforma. El código no es ejecutado, simplemente se compila en el ejecutable Dalvik y corre en la Máquina Virtual Dalvik, que es una máquina virtual especializada diseñada específicamente para Android y optimizada para dispositivos móviles que funcionan con batería y que tienen memoria y procesador limitados.
Soporte para hardware adicional	Android soporta cámaras de fotos, de vídeo, pantallas táctiles, GPS, acelerómetros, giroscopios, magnetómetros, sensores de proximidad y de presión, termómetro, aceleración 2d y 3d.
Entorno de desarrollo	Incluye un emulador de dispositivos, herramientas para depuración de memoria y análisis del rendimiento del software. El entorno de desarrollo integrado es Eclipse usando el <i>plugin</i> de herramientas de desarrollo de Android.
Market	El Android Market es un catálogo de aplicaciones que pueden ser descargadas e instaladas en dispositivos Android sin la necesidad de un PC.
Multitarea	Multitarea real de aplicaciones disponible.

Tabla 47: Algunas características de Android relevantes para **The Time Bird**

Nivel de API escogido

Uno de objetivos del proyecto, es que la aplicación Android sea compatible con todas las versiones posibles del dispositivo. Para ello, hay que escoger la *API* de desarrollo más adecuada. Cada versión nueva de Android incluye las actualizaciones de la *API* anterior, por lo que dependerá de la versión más utilizada del sistema operativo, que lleguemos a más o menos cantidad de dispositivos móviles. En la página de desarrolladores de Android [16] podemos encontrar algunas estadísticas interesantes sobre las versiones *API* que tienen actualmente los dispositivos que están en el mercado.

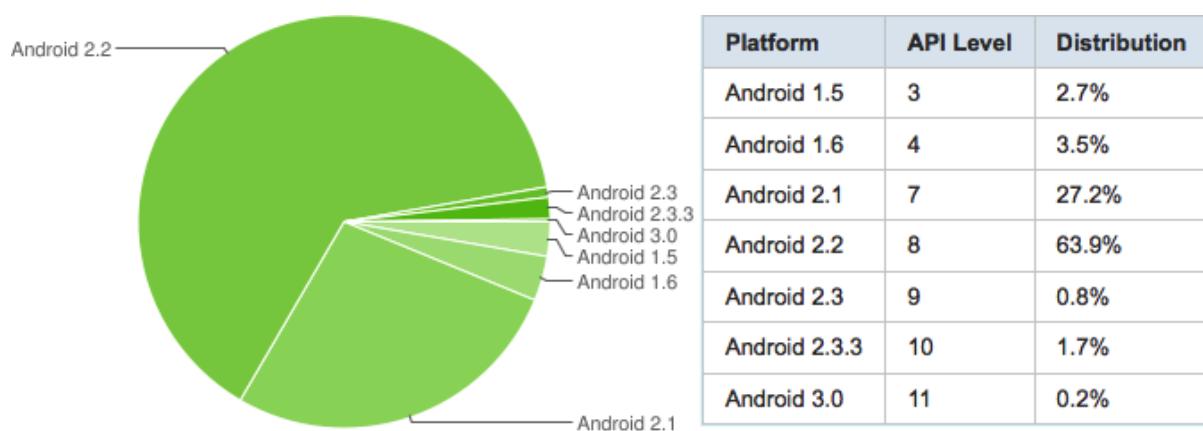


Ilustración 67: Datos de dispositivos por nivel de API a 1 de Abril de 2011

Por los resultados, obviamente todo desarrollador utilizaría la *API* 2.2 de Android, ya que el 63,9% de los dispositivos la utilizan. Aunque hay que tener en cuenta que la versión

estable más reciente es la 2.3.3, y dado que los dispositivos Android con versión 2.2 (Froyo) soportan los programas en su versión 2.3.3 (Gingerbread) se ha decidido desarrollar para esta plataforma. Actualmente la mayoría de compañías están ofreciendo actualizaciones a 2.3 de sus terminales, por lo que durante el verano del 2011 ya habrá un porcentaje de dispositivos con 2.3.3 mucho mayor.

4.2.3. Comunicación entre la aplicación móvil y web.

La conexión de datos se realiza mediante servicios web³³. La flexibilidad de Drupal permite el uso de diversas soluciones para comunicaciones de datos. El módulo más completo que existe en la actualidad se llama *Services*. La funcionalidad básica de este módulo, es que toda la información sea debidamente transmitida de una aplicación a otra. Además, proporciona una *API* por defecto para poder trabajar con protocolos de servicios web como *XMLRPC*, *SOAP* o *REST*, aunque soporta a muchos otros.

De los tres, *REST* es, posiblemente el protocolo más seguro, pero a la vez el más difícil de utilizar a la hora de parsear datos, por lo que dadas las necesidades de proyecto, y la seguridad que se puede añadir mediante el propio módulo, se ha descartado esta opción.

XMLRPC o SOAP

XMLRPC fue el primer mecanismo que surgió para invocar procedimientos remotos vía XML, ofrece una forma sencilla de invocar operaciones en sistemas heterogéneos a través de una estructura simple; *SOAP* es una implementación más robusta para llevar a cabo una intercomunicación en XML, a diferencia de *XMLRPC* o *SOAP* se le han integrado mecanismos que le permiten operar en ambientes distribuidos más complejos, algo que convierte a *XMLRPC* en el protocolo adecuado para nuestro producto.

XMLRPC o JSON

El *XML* es un estándar para el intercambio de información estructurada en diferentes plataformas. Es una tecnología sencilla y permite compartir información entre sistemas de una forma segura, fiable y fácil. Además existen numerosas herramientas tanto para clientes como para servidores que permiten procesar y transformar fácilmente su contenido.

³³ Un servicio web (*web service* en inglés) es una pieza de *software* que utiliza un conjunto de protocolos y estándares que sirven para intercambiar datos entre aplicaciones.

```
<menu id="file" value="File">
  <popup>
    <menuitem value="New" onclick="CreateNewDoc()" />
    <menuitem value="Open" onclick="OpenDoc()" />
    <menuitem value="Close" onclick="CloseDoc()" />
  </popup>
</menu>
```

Ilustración 68: Ejemplo de código escrito en XML

El *JSON* es un formato ligero para el intercambio de información. Su simplicidad y compactación ha dado lugar a la generalización de su uso, alternativamente a *XML*.

```
{"menu": {
  "id": "file",
  "value": "File",
  "popup": {
    "menuitem": [
      {"value": "New", "onclick": "CreateNewDoc()"},
      {"value": "Open", "onclick": "OpenDoc()"},
      {"value": "Close", "onclick": "CloseDoc()"}
    ]
  }
}}
```

Ilustración 69: Ejemplo de código escrito en JSON

Cabe mencionar que son formatos similares y que cada uno cuenta con seguidores y detractores, por lo que la pregunta de cuál de los dos es mejor, es un poco ambigua. La diferencia más notable entre ambos, es que para transferencia de una mayor cantidad de datos, *JSON* al ser más ligero, tiene un tamaño menor.

Vistas las diferencias, podemos concluir que la optimalidad dependerá del tamaño de datos, así como de su complejidad, pero como los dos tienen características similares, otro punto fuerte serían las herramientas de soporte de cada uno de ellos.

En el caso de ***The Time Bird***, la mayoría de funcionalidades se realizan mediante la comunicación de datos entre la aplicación web y móvil, aunque el tamaño de los datos es escaso, ya que se ha elaborado un diseño que permite transferir cadenas de caracteres únicamente. Además existen unas librerías en *Java* para *XMLRPC* que facilitan la elaboración del resto de métodos necesarios, por lo que se ha optado por el uso de este último.

Arquitectura de XMLRPC

En *XMLRPC* siempre se habla en términos de cliente/servidor, existe un sistema que realiza la solicitud (cliente) y otro que atiende (servidor), y como es de imaginarse un elemento

clave en ambos puntos es: el parser *XML*.

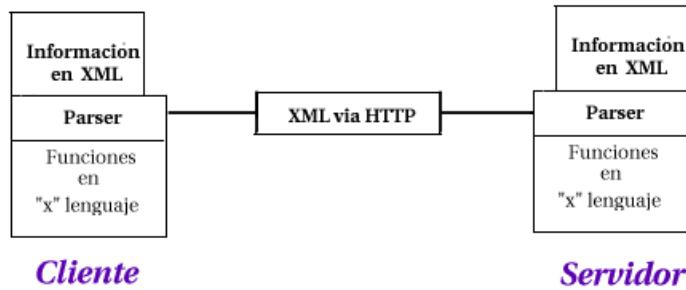


Ilustración 70: Funcionamiento de la comunicación mediante XML

Implementaciones de XMLRPC

Aunque es posible diseñar desde cero cualquier tipo de cliente y/o servidor para emplear *XML*, hoy en día ya existen diversas implementaciones para diversos ambientes y lenguajes. Es a través de estas implementaciones que se logran ahorrar diversas labores como configuraciones del **parser**, cuestiones de seguridad, integración a servidores de páginas y otros detalles secundarios. Utilizando estos *frameworks* hay que concentrarse en los procedimientos específicos y no en detalles comunes o secundarios que están siempre presentes en este protocolo.

Nosotros usamos, de todas las implementaciones, la siguiente:

- XMLRPC Apache [17] para el lenguaje de programación Java.

4.3. Arquitectura lógica

La arquitectura lógica pretende describir la estructura interna de cada uno de los sistemas o dicho de otra forma, cómo están diseñados, mediante el uso de patrones arquitectónicos comunes, o propios de la tecnología. De este modo, el uso de patrones de diseño mantiene una estrecha relación con la tecnología utilizada.

El hecho de que uno de los requerimientos no funcionales es el desarrollo de una aplicación web con Drupal, y que la mayor parte o toda la lógica de negocio este en la aplicación web, y así aprovechar su extensibilidad (nuevas funcionalidades o mejoras de los componentes) y mantenimiento (detección y reparación de errores), hace que la arquitectura base del sistema global sea la que utiliza Drupal. Antes de proceder a su descripción, es necesario comprender su funcionamiento, por lo que hay que conocer como

fluyen los datos dentro del sistema. El secreto de Drupal para conseguir su reconocida flexibilidad y facilidad en la creación de aplicaciones web, es la abstracción y organización en capas de su contenido. Hay cinco capas en el sistema, organizadas por distintos componentes anteriormente descritos.

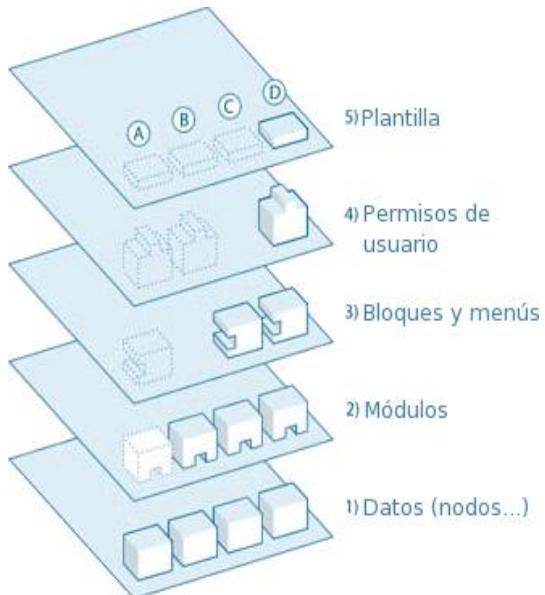


Ilustración 71: Estructura por capas, en Drupal

Drupal está estructurado en una arquitectura por capas. A nivel genérico, guarda mucha relación con algunas arquitecturas conocidas, y tal como se ha diseñado el sistema, se pueden identificar tres capas, formando así una arquitectura en tres capas:

- Los datos, incluyendo la capa de abstracción de la base de datos, así como algunos módulos contribuidos como *Views*.
- La lógica, basada principalmente en el código de los módulos.
- Presentación, representada por los temas, compuestos por plantillas.

Es obvio, que cuando pensamos en Drupal, pensamos en algunos conceptos y principios fundamentales, familiares al desarrollo web, y la arquitectura en tres capas es muy popular, por el mero hecho de ser un buen modelo y fácilmente aplicable. Este enfoque puede convertirse fácilmente en un modelo *MVC*, que muchos otros *frameworks* adoptan como arquitectura principal.

Así que, es posible describir Drupal como arquitectura en tres capas, aunque haya dos partes importantes que no logra contemplar:

- **PAC (o jerarquización de MVC):** Drupal no solo tiene pinceladas del patrón *MVC*; tiene un patrón *MVC* jerárquico más similar al patrón *PAC*. En este modelo, los distintos componentes tienen sus propias tríadas de Presentación-Abstracción-Control, y éstas

forman parte de un todo. Esto contrasta con el patrón *MVC*, que consiste en un controlador que manipula un modelo con el fin de crear algún tipo de estructura de datos que luego es entregada a la capa de presentación para su representación.

A continuación podemos ver un diagrama, de como un contenido, pasa por diferentes componentes hasta ser mostrado. Como ejemplo se puede considerar un tipo de contenido llamado **Página**, que contiene algunos bloques. Cada **bloque** se representa por separado, con su propio modelo (datos), vista (tema) y controlador (utilizando la función *hook_block* en un módulo). El bloque devuelve una estructura de datos que la vista³⁴ debe transformar en *HTML*.

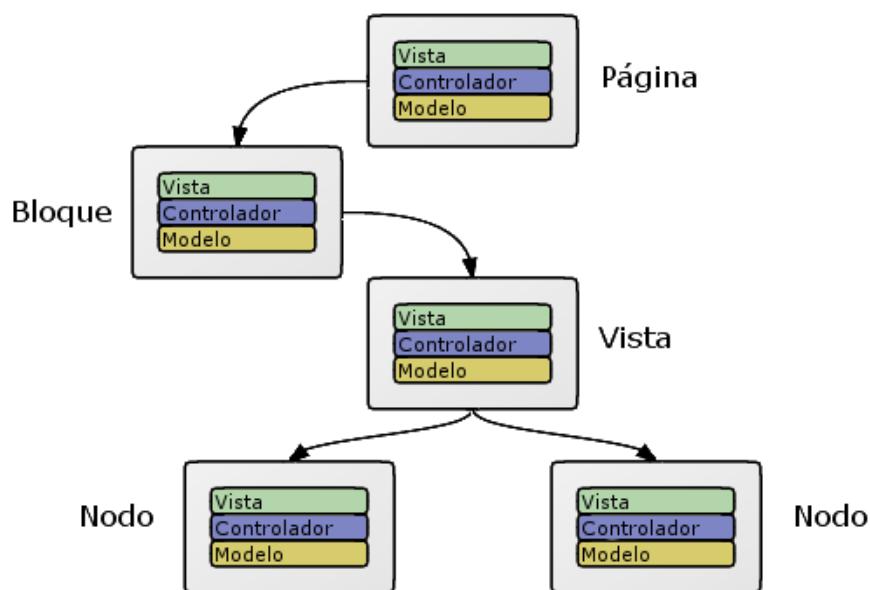


Ilustración 72: Esquema de contenido de ejemplo en Drupal como patrón de diseño PAC

Así, de la misma forma en que hay tres capas en cada uno de los procesos, tenemos una serie de tríadas *MVC*.

- **Estructura Vertical, no Horizontal:** En Drupal, un módulo representa, por llamarlo de alguna forma, una “tira” vertical del patrón de diseño en tres capas. Un módulo puede definir un esquema de datos, proveer de la lógica de negocio necesaria y definir y aplicar las funciones necesarias para presentar los datos al usuario, todo ello mediante los famosos *hooks* de los que hemos hablado anteriormente, aunque para ello sea necesario el uso de buenas prácticas de código (no es recomendable, por ejemplo,

³⁴ Las vistas, más conocidas como **Views** en Drupal, es un módulo que permite hacer consultas con una enorme flexibilidad. Es posiblemente el módulo desarrollado por terceros más potente e indispensable para Drupal.

actualizar la tabla de nodos mediante consultas *SQL* cuando existen funciones como *node_save()* para ello).

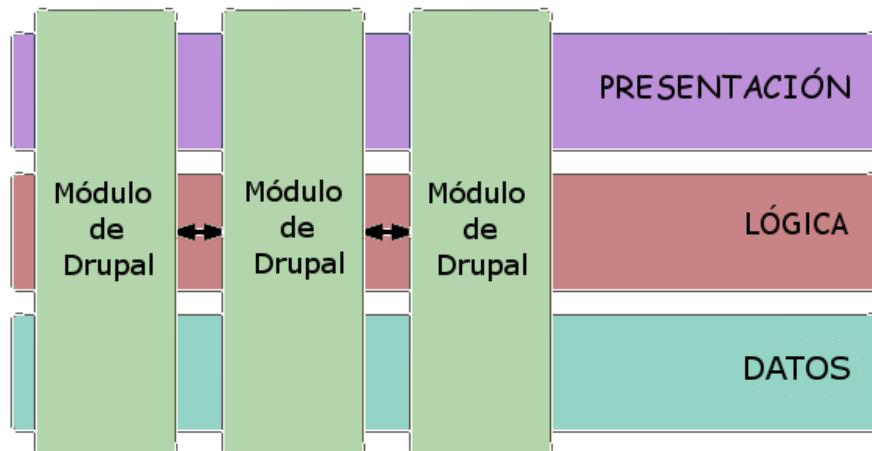


Ilustración 73: Esquema horizontal que podemos encontrar en Drupal

En resumen, se podría decir que Drupal tiene su propia arquitectura en capas, pero ninguno de los conceptos explicados podrían definir que siguiera una arquitectura en tres capas. Más bien se ha realizado una descripción como *framework*. Pese a todo, el diseño se ha hecho guardando una estrecha relación a tres capas, realizando toda la lógica de negocio en un módulo desarrollado explícitamente para ello, y utilizando plantillas para mostrar los datos a los usuarios, por lo que el diseño se puede describir en tres partes diferenciadas, siendo éstas la **interfaz, lógica de negocio y gestión de datos**.

Patrones de diseño utilizados por Drupal

Drupal tiene un diseño que utiliza algunos patrones de diseño [18] comunes y conocidos en la ingeniería del *software*, algunos de ellos descritos a continuación:

- **Singleton:** Si asociamos los módulos y temas a objetos, entonces siguen el patrón *singleton*. En general en estos objetos no se encapsulan datos, lo que separa a un módulo de otro, es el conjunto de funciones que contiene, por lo que debe ser considerado como una clase con una instancia *singleton*.
- **Decorador:** Drupal hace un uso extenso del patrón decorador. Existen *hooks* que permiten extender el comportamiento a ciertos componentes de Drupal, pero sobre todo a los nodos.
- **Observador:** El patrón Observador existe de forma generalizada en todo Drupal, ya que muchos *hooks* permiten a los módulos registrarse como observadores de los objetos presentes, para que de este modo, si se realiza algún cambio en alguno de ellos

por el uso de un *hook* determinado, se llamará a todos los módulos que lo implementen.

También contiene otros patrones como *Bridge*, *Chain of Responsibility* y *Command*, aunque los que se han descrito se perciben a simple vista cuando se empieza a desarrollar una aplicación con Drupal.

A continuación se describen cada una de las partes en que se ha dividido el diseño.

4.3.1. Interfaz gráfica

La interfaz gráfica, que es la parte que el sistema presenta al usuario, le comunica información y captura información del usuario en un mínimo proceso. La interfaz es muy conveniente que tenga características amigables para el usuario final.

Podemos dividir el diseño en dos partes, el diseño realizado para la aplicación web y para la aplicación móvil. Además, cada una de ellas se presentará mediante diagramas de navegación genéricos y por usuarios, y las debidas capturas de pantalla que representen el diseño gráfico realizado.

Los mapas de navegación permiten saber de forma rápida cual será la forma de navegar a través del *software*, o dicho de otra forma, pasar de unas pantallas de la aplicación a otras, formadas por diferentes vistas que van mostrando información de diferentes maneras.

A continuación se describe el diseño para cada una de las aplicaciones.

Diseño de la aplicación móvil

- Mapa de navegación genérico: En el mapa de navegación presentado a continuación, se contempla las diferentes pantallas a las que puede acceder un usuario organizadas por funcionalidades. Se ha creado el menú de navegación teniendo en cuenta los patrones de diseño específicos para Android [19]. Desde la pantalla de inicio de sesión, se puede acceder a la pantalla principal donde se cargarán las etiquetas propias del usuario, o se puede acceder a la pantalla informativa. Después mediante funcionalidades de contexto y propias del menú, se puede navegar entre ellas.

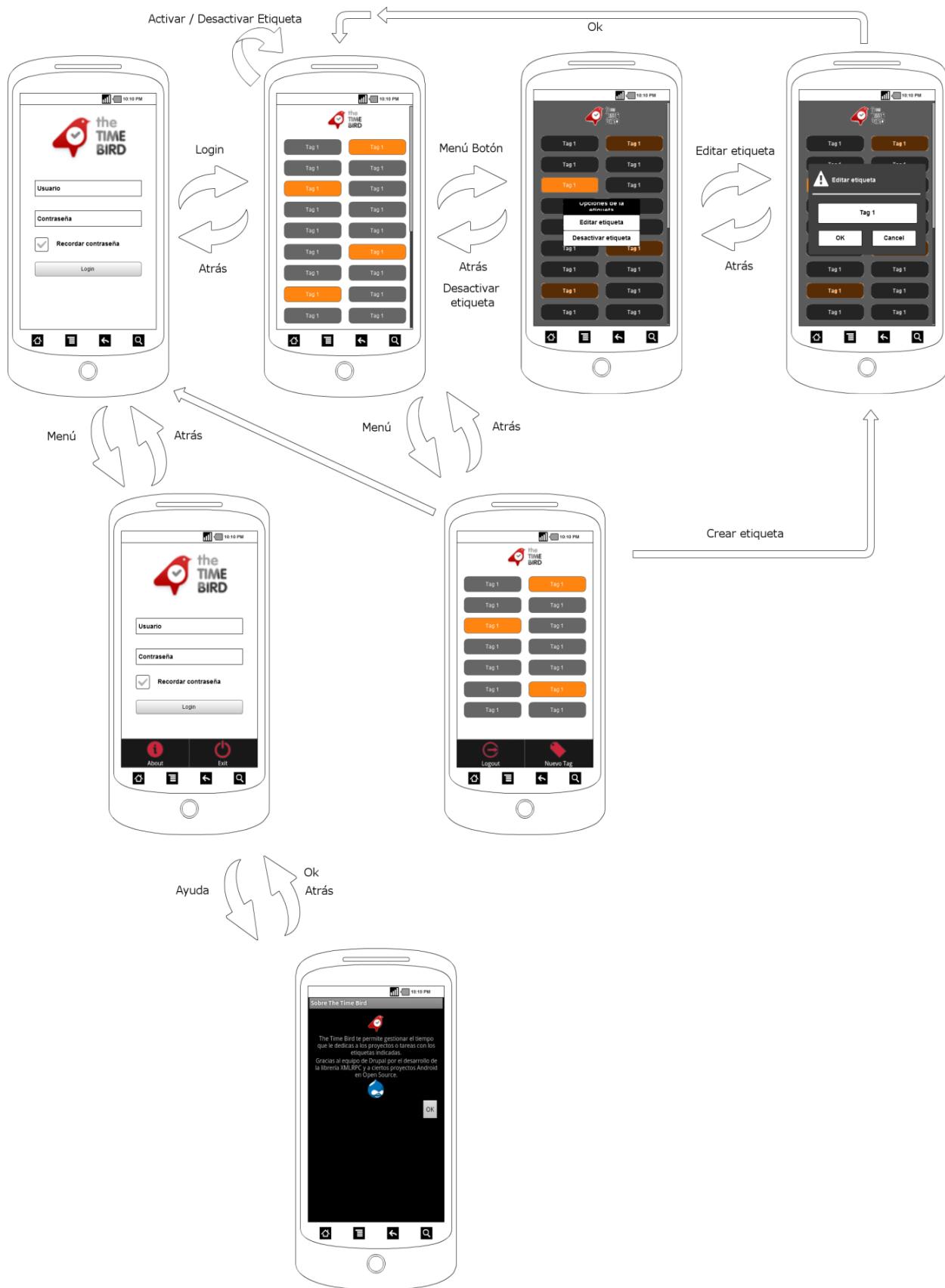


Ilustración 74: Mapa de navegación de la aplicación móvil

- Mapa de navegación por usuario: Como se puede ver a continuación, un usuario anónimo solo puede acceder a la pantalla de inicio de sesión, y mediante el menú de Android, puede acceder a la pantalla de información sobre la aplicación, mientras que un usuario registrado puede acceder a la página principal donde podrá realizar la gestión de todas sus etiquetas.

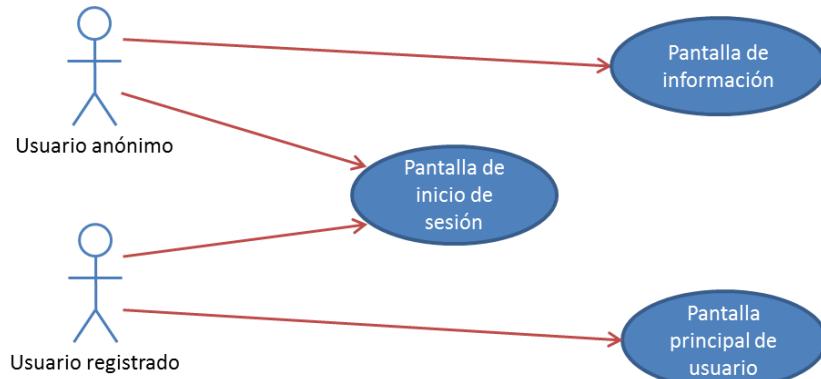


Ilustración 75: Mapa de navegación por usuarios de la aplicación móvil

- Capturas de pantalla de las pantallas o Actividades principales en la aplicación móvil: En la primera se puede observar claramente que se dispone de dos campos de texto correspondientes al nombre de usuario y contraseña para poder acceder a las funcionalidades básicas que ofrece **The Time Bird**. En la segunda imagen, se puede observar una lista de etiquetas de usuario, que se han decidido colocar en dos columnas por usabilidad. La tercera imagen corresponde a la pantalla puramente de información sobre la aplicación, a la que pueden acceder usuarios anónimos.

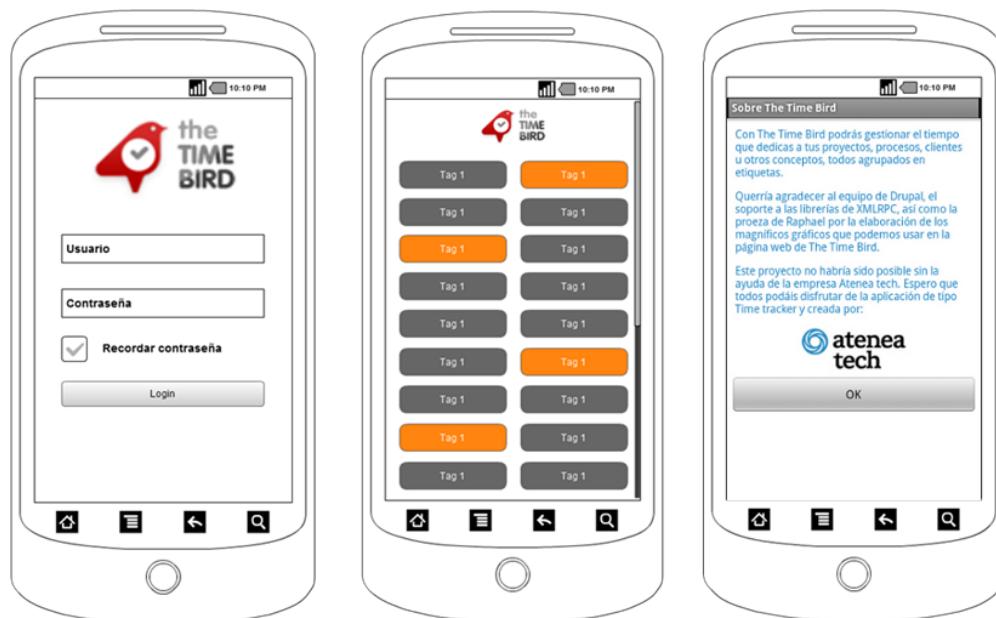


Ilustración 76: Conjunto de Actividades en la aplicación móvil

Diseño de la aplicación web

- Mapa de navegación genérico: En el mapa de navegación presentado a continuación, se muestran las diferentes páginas visibles en la aplicación web. Como se puede observar, una vez realizado el inicio de sesión de usuario, se utilizan dos menús (uno en la barra superior, y otro en la barra lateral derecha), para poder acceder directamente a la mayoría de páginas excepto a la de gráficos de ejemplo. Por lo que, una vez el usuario haya entrado, tiene un acceso rápido y simple a todas las funcionalidades de la aplicación web.

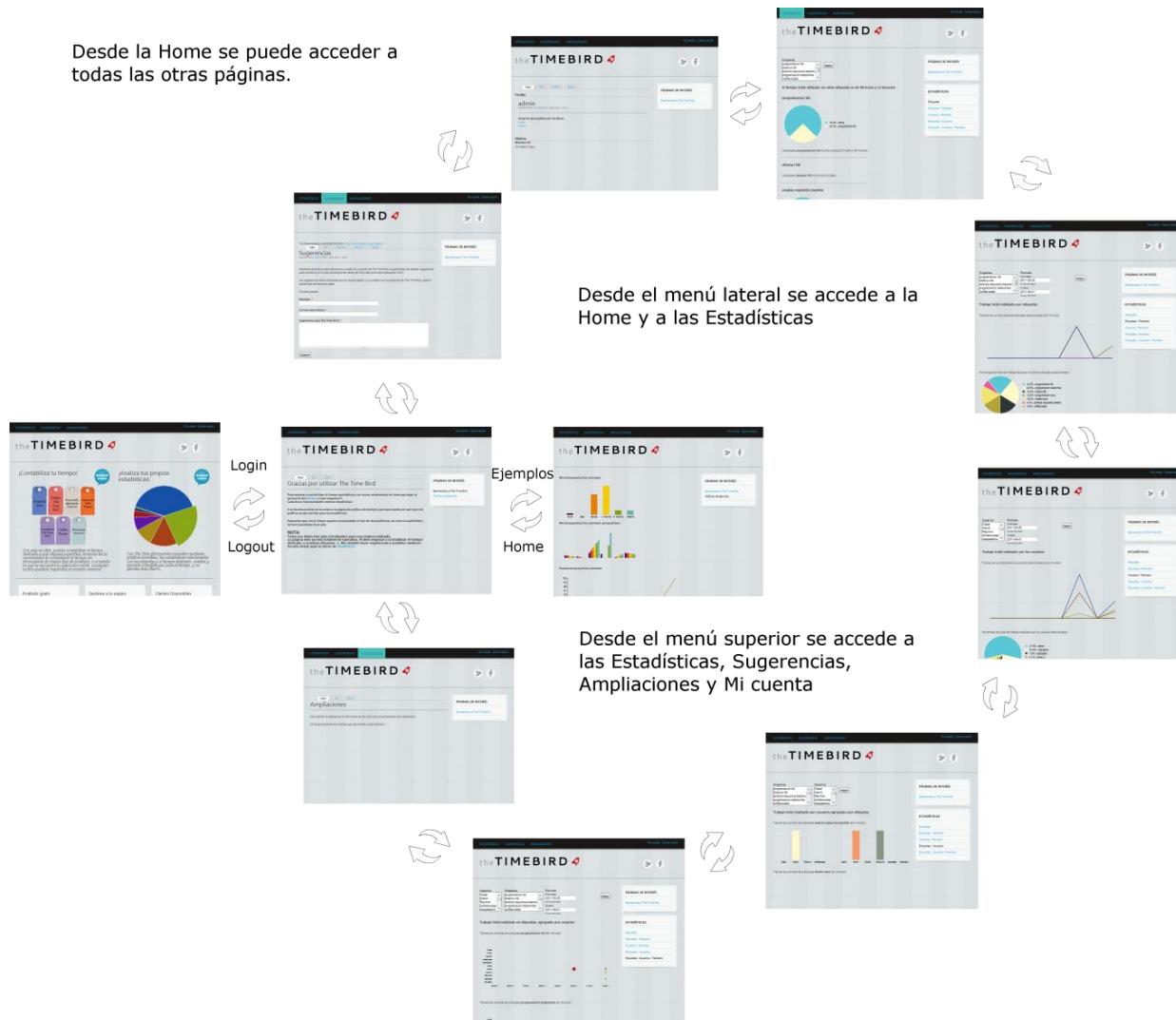


Ilustración 77: Mapa de navegación de la aplicación móvil

- Mapa de navegación por usuario: El mapa de navegación por usuario, se diferencia del anterior en qué se realiza por roles y funcionalidades, no por páginas. El usuario anónimo solo tiene acceso a la página de inicio, mientras que el usuario registrado tiene acceso a todas las páginas que corresponden a las funcionalidades más importantes.

Cabe destacar que el usuario administrador, tendrá acceso al resto de funcionalidades de administración de la aplicación web, pero no lo hemos incluido para simplificar el mapa de navegación.

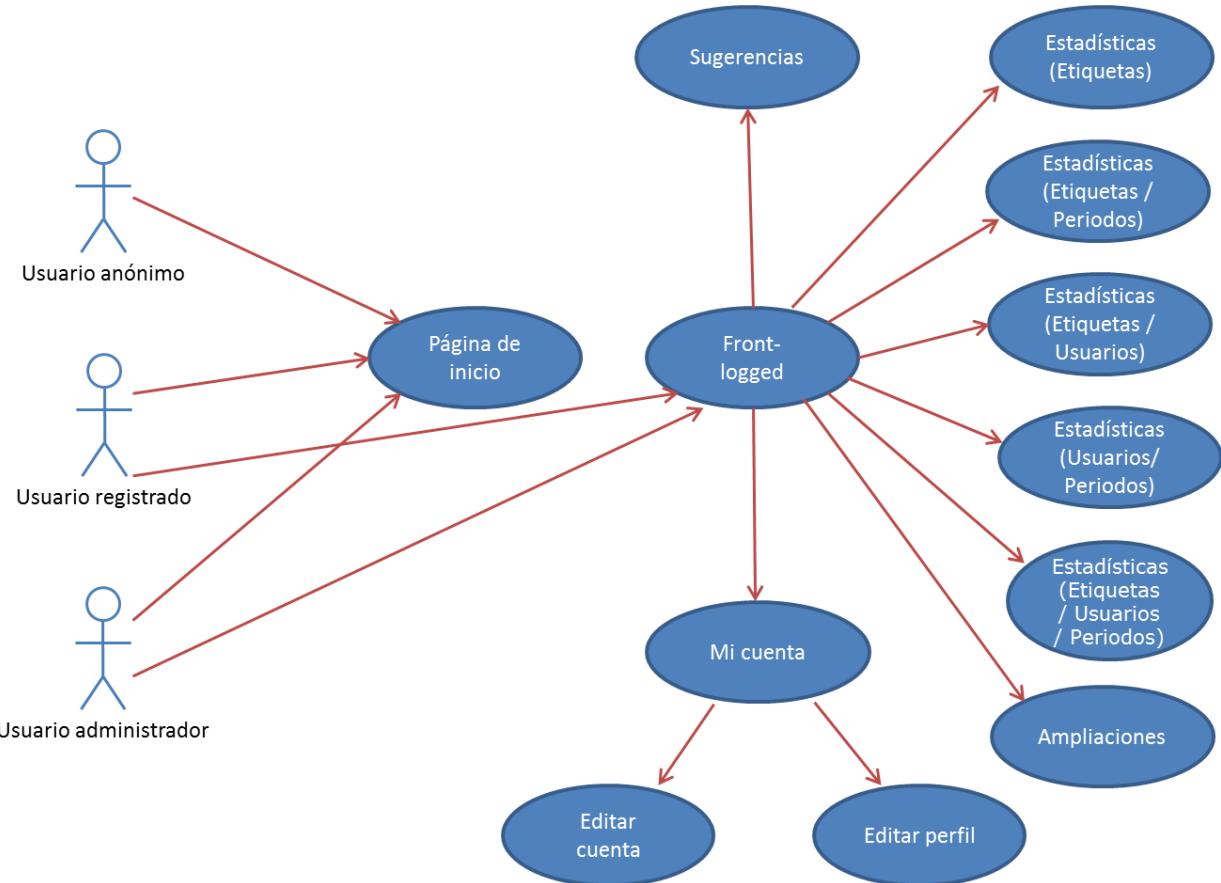


Ilustración 78: Mapa de navegación por usuarios de la aplicación web

- Capturas de pantalla: Las imágenes que se muestran a continuación se han clasificado por diseño aplicado. La página de presentación del producto, tiene un diseño propio. Lo mismo pasa con todas las páginas de texto creadas página de inicio de usuario, ampliaciones, sugerencias, etc.). Las páginas de estadísticas tienen un mismo estilo, y los gráficos dibujados son del mismo tamaño. Por último, diferenciar el estilo utilizado en la página de cuenta de usuario.

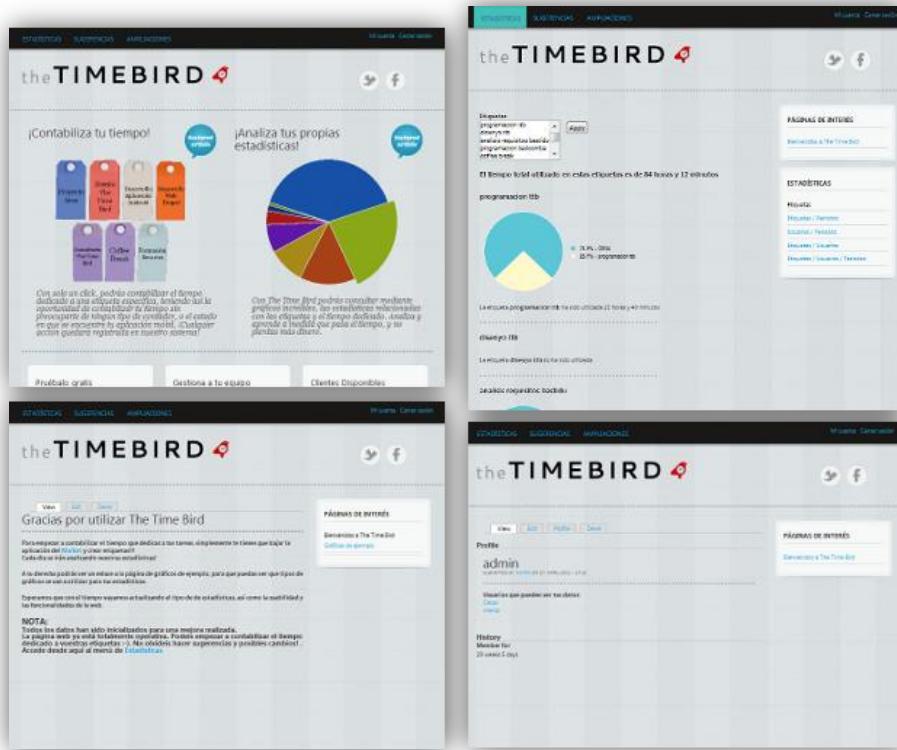


Ilustración 79: Diseño por páginas en la aplicación web

Módulos y temas necesarios para el diseño gráfico de la aplicación web

Para poder proceder con el diseño de la aplicación web, se van a utilizar los siguientes módulos:

- **CSS Injector:** Permite a los administradores insertar código CSS en las páginas deseadas mediante reglas de configuración. Es muy útil para añadir pequeños fragmentos de CSS sin tener que modificar el tema utilizado en Drupal.
- **Menu block:** Permite introducir elementos del menú de segundo nivel en los bloques, dando así la posibilidad al usuario de tener una mejor experiencia en la navegación web. El principal objetivo es que una vez se ha enlazado la página de estadísticas, todos los tipos de estadísticas sean enlazables desde un bloque en la parte lateral derecha de la página.
- **Raphael:** Nos ofrece las librerías en *PHP* y *Javascript* originales de los gráficos de Raphael, necesarias para la elaboración de los diferentes tipos de gráficos que él mismo ha desarrollado.
- **Journalcrunch:** Tema atractivo de forma visual, que destaca por su sencillez y que presenta unos rasgos que se adaptan perfectamente a las necesidades de ***The Time Bird***.

4.3.2. Lógica de negocio

La lógica de negocio es la encargada de que las funcionalidades especificadas hagan lo que tengan que hacer, ya sea para preparar datos para presentarlos al usuario, como preparar los datos para ser tratados por Drupal y ser guardados en la base de datos. Para ello ejecutará los métodos que sean necesarios, ya sean del propio Drupal o desarrollados explícitamente para realizar las funciones especificadas.

En el caso de **The Time Bird**, la mayoría de estas funcionalidades están desarrolladas en un módulo personalizado, que permite enviar y recibir peticiones de la aplicación móvil, mediante el uso de diferentes métodos que son llamados desde el módulo *Services* cuando se realiza una petición desde la aplicación móvil.

Se van a utilizar diagramas de secuencia para mostrar la interacción entre el conjunto de objetos de las diferentes aplicaciones, para cada una de las funcionalidades. Además, en cada uno de los diagramas, se podrá observar claramente que en los objetos pertenecientes a la aplicación web se encuentra la lógica de negocio, y los objetos pertenecientes a la aplicación móvil son de transmisión de datos. Además, están diferenciados los métodos propios y los *hooks*, que son funciones de forma `hook_nombre` del módulo.

Los objetos que van a intervenir en los diagramas son:

- **Cliente externo:** Cliente externo, en nuestro caso la aplicación **The Time Bird** instalada en un dispositivo móvil con Android.
- **Cliente XMLRPC:** Cliente XMLRPC, en nuestro caso, formado por las librerías de Apache y las librerías XMLRPC desarrolladas específicamente en la aplicación móvil para poder realizar peticiones a la aplicación web.
- **Módulo Services:** Módulo que gestiona todas las peticiones entrantes. Depende de la llamada que se haya realizado, utilizará los métodos que el propio módulo incorpora, o delegará esta función al módulo donde se encuentren desarrollados estos métodos.
- **Módulo TTB:** Módulo desarrollado específicamente para realizar la mayor parte de la lógica de negocio necesaria en **The Time Bird**. En este módulo encontramos todos los métodos específicamente diseñados para la comunicación entre la aplicación móvil y web. Es el módulo *Services* el que le delegará las peticiones que así lo requieran.
- **Núcleo de Drupal y/o otros módulos:** Tanto el propio Drupal como algunos módulos de terceros (*Views* por ejemplo), contienen una *API* que permite a módulos externos interactuar con el sistema y realizar los cambios necesarios en la base de datos.

Ya que el proceso seguido es muy parecido en muchas de las funcionalidades, solo se van a describir algunos de los diagramas más importantes.

Inicio de sesión

El siguiente diagrama muestra el proceso seguido al iniciar una sesión de usuario en la aplicación móvil, hasta que la aplicación web realiza la validación del usuario correspondiente y guarda la información de sesión correspondiente.

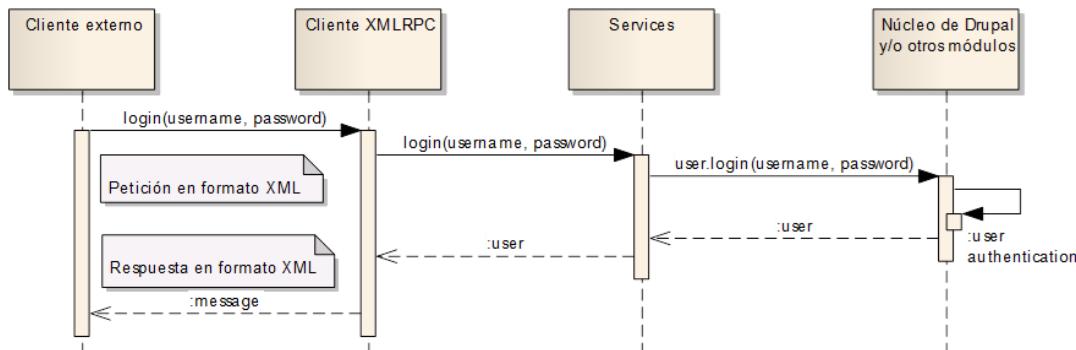


Ilustración 80: Diagrama de secuencia ampliado, de Inicio de sesión

Activar etiqueta

El siguiente diagrama muestra el proceso seguido al activar una etiqueta, desde el envío de datos en la aplicación móvil hasta que la aplicación web guarda la información en la base de datos.

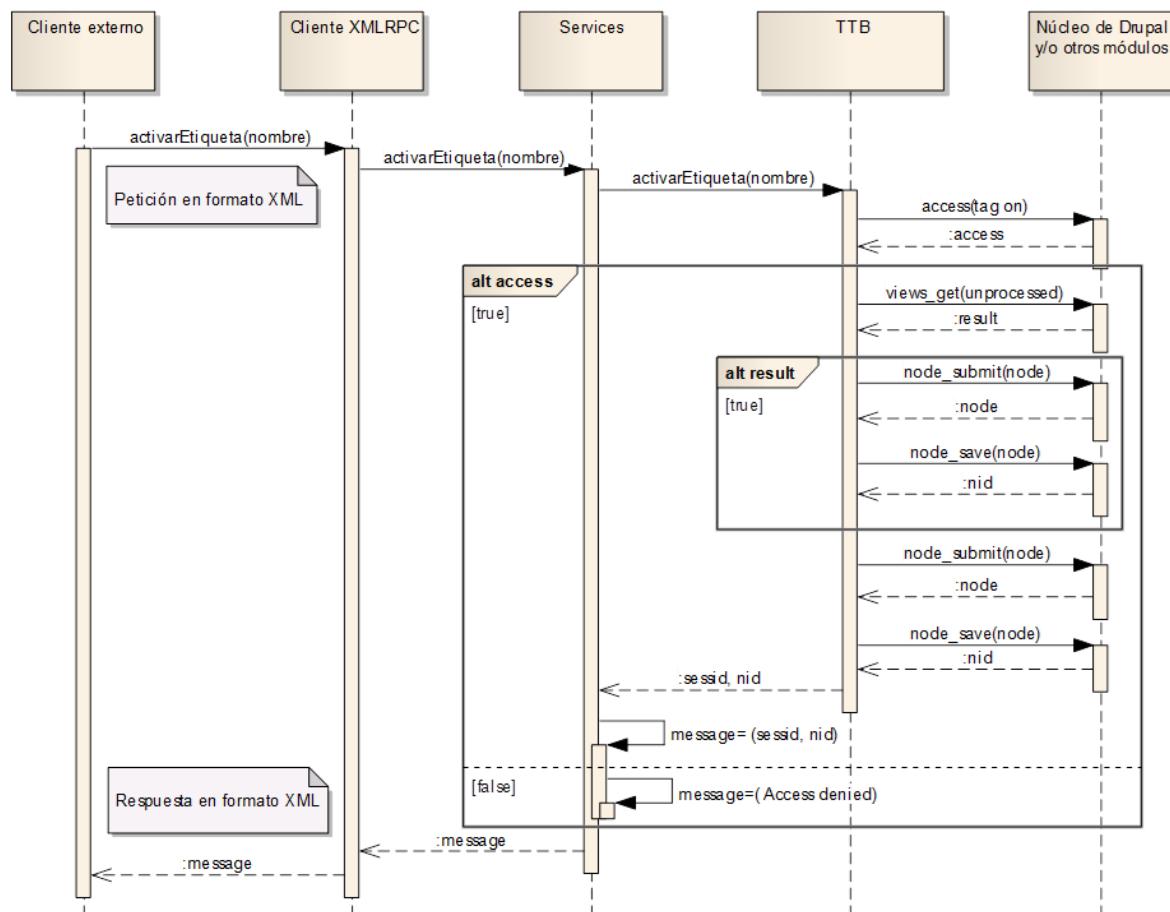


Ilustración 81: Diagrama de secuencia ampliado, de inicio de Activar etiqueta

Desactivar etiqueta

El siguiente diagrama muestra el proceso seguido al desactivar una etiqueta, desde el envío de datos de la aplicación móvil hasta que la aplicación web guarda la información en la base de datos.

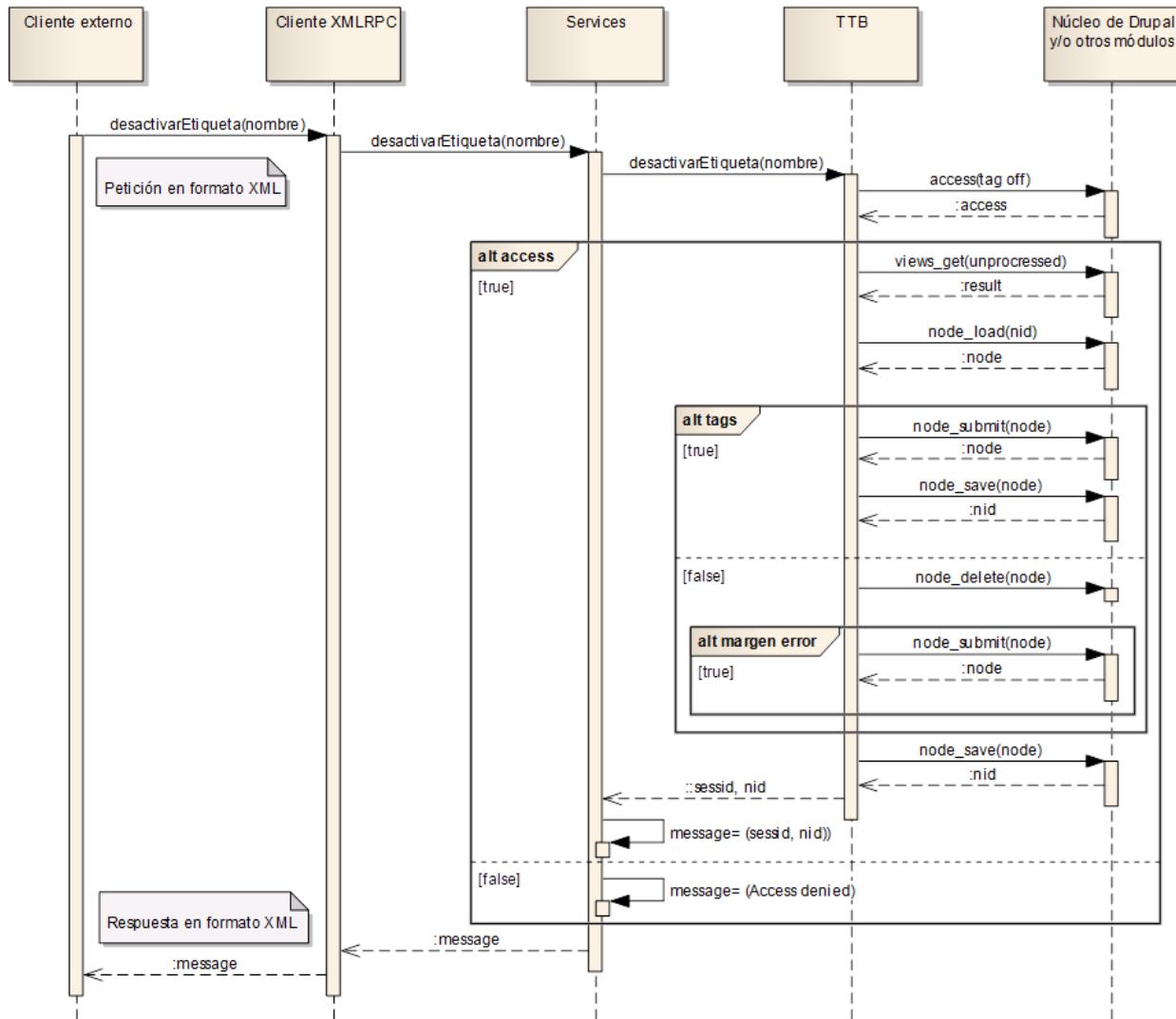


Ilustración 82: Diagrama de secuencia ampliado, de inicio de Desactivar etiqueta

Crear etiqueta

El siguiente diagrama muestra el proceso seguido al crear una etiqueta, desde el envío de datos en la aplicación móvil hasta que la aplicación web guarda la información en la base de datos.

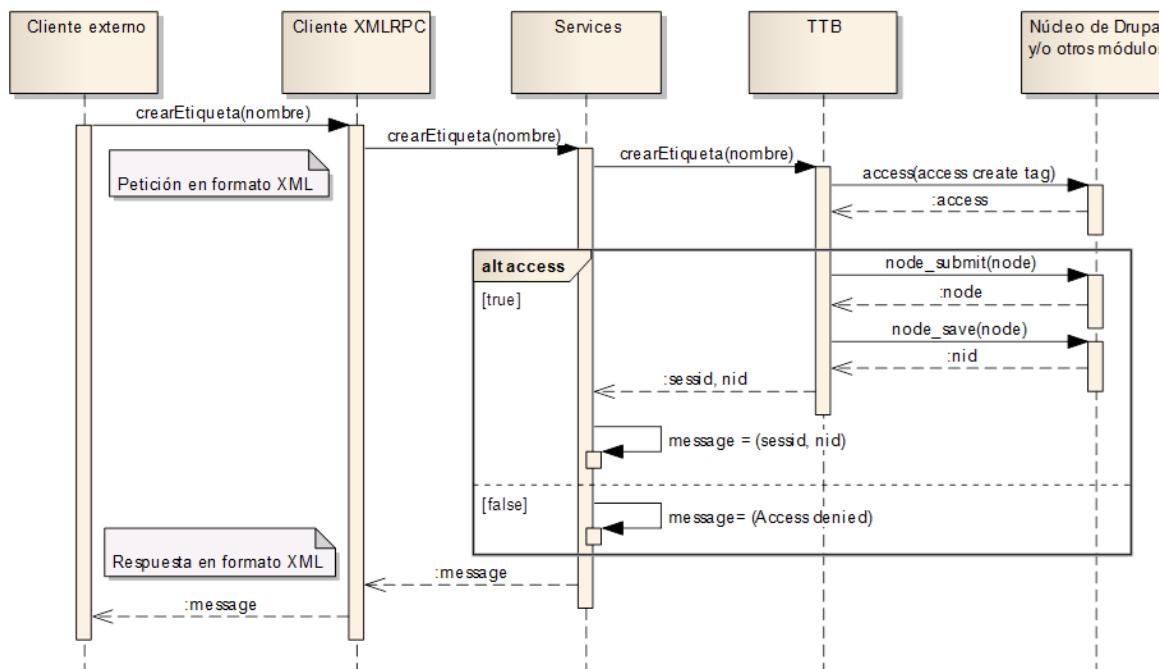


Ilustración 83: Diagrama de secuencia ampliado, de Crear etiqueta

Editar etiqueta

El siguiente diagrama muestra el proceso seguido al editar una etiqueta, desde el envío de datos en la aplicación móvil hasta que la aplicación web guarda la información en la base de datos.

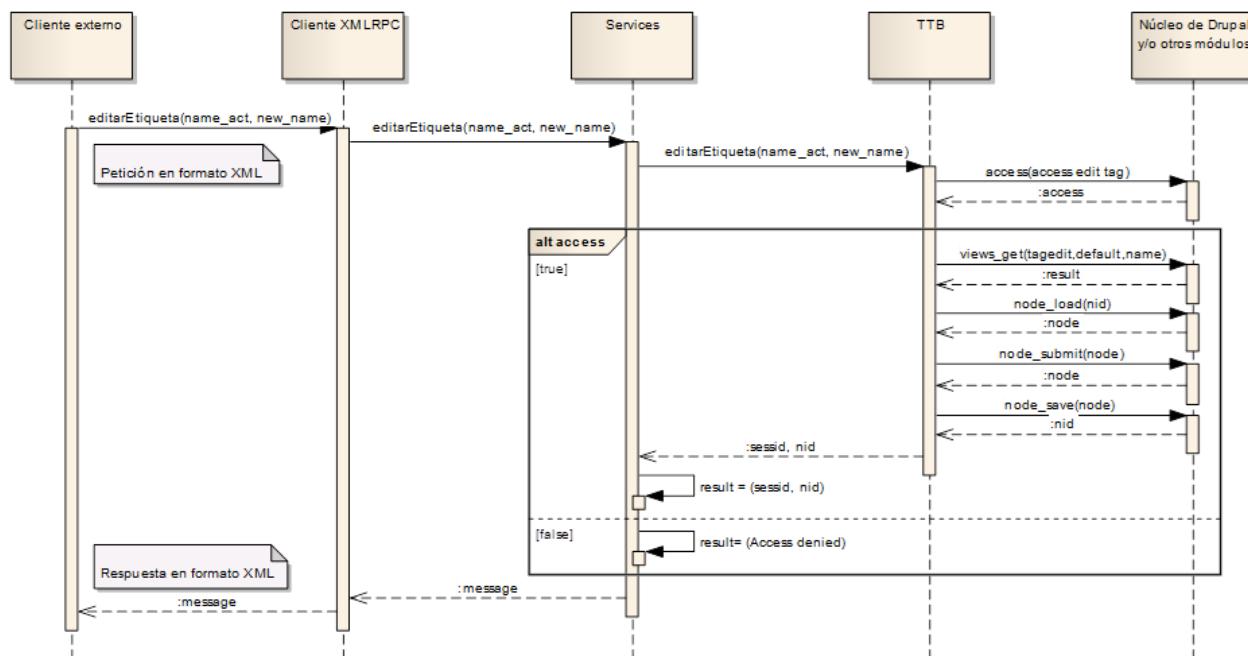


Ilustración 84: Diagrama de secuencia ampliado, de Editar etiqueta

Proceso de transformación de tareas (mediante *hook_cron*)

El siguiente diagrama muestra el proceso seguido al realizar el proceso de transformación de datos y así poder realizar una mejor lectura de datos, a la hora de realizar ciertas estadísticas.

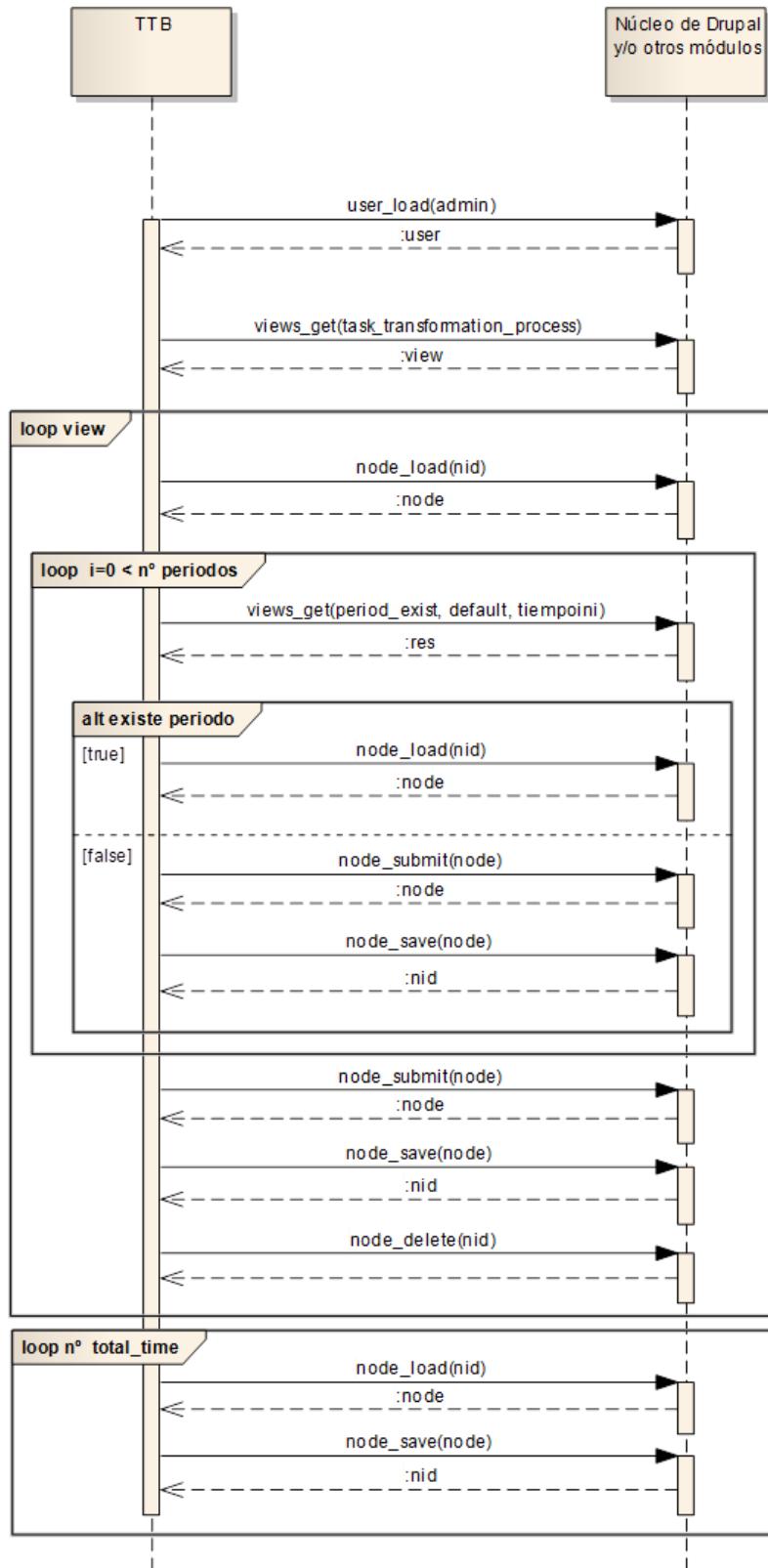


Ilustración 85: Diagrama de secuencia ampliado, del proceso de transformación en Tareas procesadas

Módulos necesarios para incluir las funcionalidades anteriores

Para poder proceder a la creación de todo el contenido mencionado, es necesaria la instalación y configuración de ciertos módulos de Drupal, que nos proporcionan parte de las funcionalidades deseadas.

- **Services:** Este módulo, provee de una solución estandarizada para la integración de aplicaciones externas con Drupal. En nuestro caso, hacemos uso de su *API* para la elaboración de todas las funcionalidades que necesiten una transferencia de datos entre la aplicación móvil y la aplicación web.
- **Views:** Es uno de los módulos más utilizados en Drupal, ya que nos permite hacer consultas de complejidad elevada mediante una interfaz fácil de usar. Además el uso de este módulo para consultas a la base de datos, es una forma de reutilizar código para ciertas consultas similares, por lo que aporta mucha flexibilidad.
- **TTB:** El módulo que incluirá las funcionalidades más importantes para ***The Time Bird***. Es un módulo desarrollado específicamente para ello.

4.3.3. Gestión de datos

La gestión de datos se encarga de la persistencia de datos, haciendo referencia al almacenamiento y obtención de datos usando un sistema de almacenamiento permanente.

En Drupal, los datos tratados en el modelo conceptual se forman de diferentes tablas y atributos. Drupal solo soporta *SGBD* relacionales (*MySQL* y *PostgreSQL*) por lo que no se permiten almacenar objetos, solo tipos simples. Cada uno de los objetos del modelo conceptual, estará dividido en tantas tablas como atributos sea necesario. Para ello se necesitarán de ciertos módulos desarrollados por terceros y que aportarán a la base de datos las tablas que sean necesarias para aportar ciertos campos de un objeto.

Drupal proporciona una capa de abstracción de la base de datos [20], que permite el acceso a diferentes servidores de bases de datos, usando el mismo código, proporcionando así a los desarrolladores la capacidad de soportar diferentes servidores de bases de datos fácilmente. Su intención es la de preservar la sintaxis y el potencial de *SQL* tanto como sea posible, dejándole el control a Drupal de las partes de consultas que necesitan ser escritas de forma diferente para los diferentes servidores así como proveer controles básicos de seguridad.

La mayoría de las consultas de bases de datos, se realizan mediante la llamada a `db_query()` o `db_query_range()`. También existen módulos de terceros que facilitan la gestión de la base de datos, gracias a la *API* que provee Drupal para ello.

En **The Time Bird**, los datos más importantes con los que se va a trabajar, son los diferentes tipos de contenido necesarios para cada una de las funcionalidades, y todos ellos están almacenados en la aplicación web.

Etiquetas

Las etiquetas, son uno de los tipos de contenido de más importancia en **The Time Bird**. El concepto que una etiqueta representa, es más bien abstracto, siendo así los procesos, clientes, proyectos u otras entidades que puedan ser de utilidad para los usuarios.

Los campos más importantes de una etiqueta son:

- Título: Nombre representativo para la etiqueta.
- Estado: Estado en el que se encuentra la etiqueta. El estado está representado por un entero, que por defecto será 1, significando que está habilitada, y podrá cambiar a 0, significando que está deshabilitada.
- Tiempo dedicado: Campo, con valor inicial 0, que permite saber el tiempo dedicado a la etiqueta en cuestión.
- Usuario: Campo que referencia al usuario que ha creado la etiqueta.



Ilustración 86: Etiqueta

Tareas no procesadas

Las tareas no procesadas, son tareas creadas con el propósito de que los datos enviados por la aplicación web, se almacenen de la forma más rápida posible, creando así un sistema transaccional rápido y eficaz.

Los campos más importantes de una tarea no procesada son:

- Tiempo inicial: Tiempo en formato *timestamp*³⁵ de inicio de la tarea.

³⁵ El *timestamp* es una secuencia de caracteres que representa una fecha y/u hora específicas. Es el tiempo en que un evento es guardado por una computadora, representando el número de segundos desde el dia 1 de Enero de 1970.

- Tiempo final: Tiempo en formato *timestamp* de finalización de la tarea.
- Etiquetas: Conjunto de identificadores de las etiquetas utilizadas en dicha tarea.
- Usuario: Campo que referencia al usuario que ha creado la etiqueta.



Ilustración 87: Tarea no procesada

Periodos

Cada uno de los periodos representa una unidad de tiempo trabajada en una tarea. Se ha decidido utilizar periodos en fracciones de minuto para que la contabilidad de las etiquetas sea lo más exacta posible.

Los campos más importantes de un periodo son:

- Tiempo inicial: Tiempo en formato *date* (fecha) de la fracción de tiempo que representa, siempre en porción de minutos.



Ilustración 88: Periodo

Tareas procesadas

Las tareas procesadas, son el tipo de contenido de más importancia en ***The Time Bird***. Estas tareas representan los datos finales utilizados para mostrar ciertas estadísticas mediante gráficos, de la información que las tareas almacenan.

Los campos más importantes de una tarea procesada son:

- Periodos: Campos que referencian a cada uno de los periodos utilizados por la tarea. Por ejemplo, si una tarea se ha realizado en 60 minutos. Habrá 60 referenciadas a cada uno de los periodos que representan cada uno de esos minutos.
- Etiquetas: Campos que referencian a cada una de las etiquetas utilizadas en la tarea.
- Usuarios: Campo que referencia al usuario que ha creado la etiqueta.

2011,05,30	
SUBMITTED BY ADMIN ON 30. MAY 2011 - 17:00	

periodos:	
1306744020	
1306744080	
1306744140	
1306744200	
1306744260	
1306744320	
1306744380	
1306744440	
1306744500	
1306744560	
1306744620	
1306744680	
1306744740	
1306752660	
1306752720	
1306752780	
1306752840	
1306752900	
1306752960	
Etiquetas:	
edukame	
Usuarios:	
carlos	

Ilustración 89: Tarea procesada

La elaboración de un tipo de contenido de estas características, nos da flexibilidad a la hora de añadir nuevos datos a la tarea, y así poder añadir nuevos tipos de estadísticas. Además, nos permitirá hacer una lectura de datos mucho más rápida que si tuviéramos que estudiar los datos recogidos de otra forma.

Perfiles

Los perfiles van asociados directamente a los usuarios. Este tipo de contenido es necesario ya que permite a cada uno de los usuarios, tener una lista de usuarios a los que permite ver sus datos en estadísticas representadas en gráficos.

Los campos más importantes de un perfil son:

- Lista de usuarios: Campo que referencia a todos los usuarios a los que se les da permiso para poder obtener sus datos mediante gráficos.

Raymon
SUBMITTED BY RAYMON ON 27. APRIL 2011 - 17:27

Usuarios que pueden ver tus datos:

- shatir
- danielazo
- menor

Ilustración 90: Perfil

Sugerencias

Un aspecto importante en **The Time Bird**, es que permitimos a todos los usuarios envíos de sugerencias sobre posibles mejoras o ampliaciones del producto.

Los campos más importantes de una sugerencia son:

- Nombre: Nombre del usuario, como identificador.
- Correo electrónico: Correo electrónico del usuario, para así poder contestarle en caso que sea necesario.
- Sugerencia: Campo de texto para que el usuario pueda introducir la sugerencia que desea realizar.

Sugerencias
SUBMITTED BY ADMIN ON 7. APRIL 2011 - 15:25

Mediante este formulario ofrecemos a todos los usuarios de The Time Bird, la posibilidad de realizar sugerencias para cambios y/o futuras ampliaciones tanto del sitio web como de la aplicación móvil.

Las sugerencias serán evaluadas por los responsables, y si cumplen con los objetivos de The Time Bird, cabe la posibilidad de llevarse a cabo.

Muchas gracias.

Nombre: *

Correo electrónico: *

Sugerencia para The Time Bird: *

Ilustración 91: Sugerencia

Páginas e Historias

Aunque no sea un tipo de contenido esencial en **The Time Bird**, es necesaria la creación de páginas e historias, o contenido informativo para el usuario. Las páginas son creadas para ofrecer ciertas funcionalidades de carácter informativo, y lo mismo pasa con las historias, aunque estas tienen la opción de presentarse en la página principal. Es la única característica que las diferencia.

Los campos más importantes de una página y una historia son:

- Título: Título para la página, o cabeza principal de la información.
- Cuerpo: Texto informativo sobre la página o historia correspondiente.

Gracias por utilizar The Time Bird

Para empezar a contabilizar el tiempo que dedicas a tus tareas, simplemente te tienes que bajar la aplicación del [Market](#) y crear etiquetas!!!
Cada día se irán analizando vuestras estadísticas!

A tu derecha podrás ver un enlace a la página de gráficos de ejemplo, para que puedas ver que tipos de gráficos se van a utilizar para tus estadísticas.

Esperamos que con el tiempo vayamos actualizando el tipo de de estadísticas, así como la usabilidad y las funcionalidades de la web.

NOTA:

Todos los datos han sido inicializados para una mejora realizada.
La página web ya está totalmente operativa. Podeis empezar a contabilizar el tiempo dedicado a vuestras etiquetas :-). No olideis hacer sugerencias y posibles cambios! . Accede desde aquí al menú de [Estadísticas](#)

Ilustración 92: Página



Ilustración 93: Historia

Módulos necesarios para la creación de contenido

Para poder proceder a la creación de todo el contenido mencionado, es necesaria la instalación y configuración de ciertos módulos de Drupal, que nos proporcionan parte de las funcionalidades deseadas.

- **CCK (Construction Content Kit):** Permite la creación de campos personalizados para los diferentes tipos de contenido disponibles. Drupal por defecto solo ofrece un título y cuerpo, como campos propios de contenido. Este módulo proporciona sub-módulos como **Number** y **Text** que tienen que ser activados para poder tener campos de tipo numérico y textual en **Tareas no procesadas**.

También proporciona los sub-módulos **Node Reference** y **User Reference** necesarios para tener en las **Tareas procesadas** referencias del usuario que ha realizado la tarea, y las **Etiquetas** utilizadas en la misma, así como la lista de usuarios a los que dejamos ver nuestros datos, campo utilizado en los **Profiles** de cada usuario **autenticado**.

- **Date:** Una de las principales características del tipo de contenido **Periodo**, es que ha de contener un campo con el tiempo inicial del mismo. Para ello es necesario guardar el tiempo en formato *Date* para que así Drupal pueda transformar datos guardados como *timestamp* en el formato de fecha deseado.
- **Back Reference:** Proporciona una relación 1 a 1 entre nodos que contengan campos de tipo **Node Reference**, por lo que requiere su previa activación. Este módulo será muy útil para poder acceder a las **Tareas procesadas** desde las **Etiquetas** que se utilizan, así como todos sus **Periodos**.
- **Content Profile:** Específicamente diseñado para permitir la creación de **Profiles** de forma que todo el contenido creado como este tipo de contenido, tenga la lógica necesaria para que el **Perfil** esté asociado a la cuenta del propio usuario y se puedan realizar las acciones necesarias pertinentes. En este caso, el **Perfil** de cada usuario contendrá una lista de otros usuarios registrados en la aplicación web.
- **Webform:** Específicamente diseñado para permitir la creación de formularios, presentando toda la lógica de negocio para el envío y recepción de los formularios llenados. También proporciona mecanismos de validación de campos del tipo correspondiente.
- **Auto Node Title:** Da la posibilidad de crear títulos automáticos para el contenido creado, basándose en diferentes patrones del propio contenido. Esto permite que cada vez que se crea una tarea, no sea necesario que el usuario le ponga nombre, dejando así que de esta tarea se ocupe el sistema.

5. Implementación

Una vez visto el diseño del producto, se va a proceder a describir de forma detallada las tareas realizadas durante el desarrollo de las dos aplicaciones.

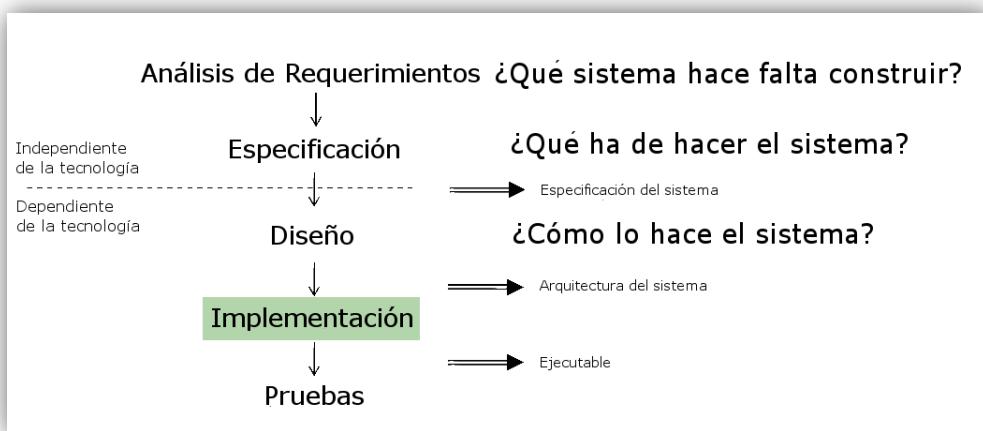


Ilustración 94: Etapas de desarrollo software (Implementación)

Este capítulo va a estar dividido en cuatro grandes partes. Primeramente se describe el trabajo realizado una vez instalado Drupal y configurado el entorno de desarrollo para la aplicación móvil (procesos descritos en el anexo) para más adelante hacer posible el desarrollo de las funcionalidades principales. A continuación se procede con el desarrollo de las librerías de comunicación de datos, seguido de las funcionalidades en cada una de las aplicaciones. Finalmente se describen las interfaces desarrolladas tanto en la aplicación móvil como web, y se realizan las configuraciones necesarias para que nuestros clientes puedan disfrutar de **The Time Bird**.

Se justifica este método para describir el desarrollo, con la finalidad de seguir de la forma más fiel posible, la arquitectura de Drupal en capas, vista en el capítulo de diseño.

5.1. Configuración inicial de la aplicación web

Una vez instalado Drupal en el servidor de desarrollo se ha realizado con la configuración inicial. Para ello, se procede a la creación de los tipos de contenido con los que se trabajará en todo momento, seguidamente de la gestión de permisos, para dar accesos a ciertos lugares de la aplicación web a los usuarios (no daremos la opción de activar o desactivar etiquetas desde la aplicación web, por ejemplo) y de la configuración del módulo Services que permite el intercambio de datos entre las aplicaciones.

Módulos necesarios para administración y desarrollo

Para poder administrar la aplicación web de forma fácil y rápida, y desarrollar funcionalidades de forma cómoda, existen dos módulos casi obligatorios para todo desarrollador.

- **Admin:** Proporciona una mejor gestión de las opciones de administración de la aplicación. El módulo proporciona una barra lateral con un menú de navegación con las diferentes opciones que tiene el usuario sobre el sitio.
- **Devel:** Módulo muy útil para desarrolladores, ya que proporciona una *API* que se puede utilizar en todo momento en el desarrollo de módulos propios. Además nos permite estudiar la estructura interna de todo el contenido creado, así como la posibilidad de crear contenido de prueba.

5.1.1. Creación de contenido

Una vez realizada la configuración inicial de la aplicación web, se procede a crear todo el contenido necesario y así poder proceder con el desarrollo de todas las funcionalidades. Pero antes de crear todos los tipos de contenido necesarios, descritos en el capítulo anterior, es necesaria la instalación de los módulos **Date**, **Webform**, **CCK**, **Back Reference** y **Content Profile**, que nos proporcionarán ciertos campos y tipos de contenido que Drupal no incorpora por defecto.

Una vez hecho esto, ya tendremos todos los datos necesarios para empezar a trabajar con las funcionalidades de las dos aplicaciones.

5.1.2. Permisos de usuario

Los permisos de usuario, se tendrán que ir activando/desactivando a medida que se desarrollan nuevas funcionalidades en la aplicación web, o se instalan nuevos módulos. Es de vital importancia que los usuarios tengan los permisos adecuados de acceso a los datos, ya que de lo contrario se podría producir el robo de información privada de los usuarios registrados en la aplicación web.

El único permiso que tiene que estar activado inicialmente tanto para los usuarios registrados como para los anónimos, es el acceso a todo el contenido de la aplicación.

PERMISSION	ANONYMOUS USER	AUTHENTICATED USER	IN CHARGE
node module			
access content	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>

Ilustración 95: Página de permisos de Drupal

5.1.3. Instalación y configuración del módulo *Services*

Al estar toda la lógica de negocio concentrada en la aplicación web, es muy importante tener configurado este módulo que permitirá la creación de las funcionalidades necesarias para el intercambio de datos entre las dos aplicaciones.

Las características principales que ofrece el módulo son:

- *Service API* le permite a los módulos crear otros servicios, incluyendo control de acceso.
- *Server API* que le permite a otros módulos crear otros servidores, como *REST* o *SOAP*.
- Navegador y página de testeo.
- Administrar API Keys.
- Integración con funcionalidades del núcleo de Drupal, como archivos, nodos, taxonomías, usuarios y más.

Instalando Services

Una vez instalado el módulo, éste contiene tres sub-módulos importantes:

- Autenticación: Incluye el módulo *Key Authentication* que permite añadir seguridad a la conexión entre la aplicación externa y la aplicación web.
- Servers: Incluye el módulo *XMLRPC Server* que permite tener un servidor XMLRPC para la realización de conexiones. Este módulo es imprescindible.
- Services: Incluye varios módulos que permiten obtener y enviar cierto tipo de información a la aplicación. Los más importantes y que han de ser activados son:
 - System Service: Módulo que permite inicializar la conexión con el servidor XMLRPC. Este módulo es imprescindible para poder iniciar sesiones de usuario en la aplicación web.
 - User Service: Módulo que permite administrar usuarios, ya sea crearlos o eliminarlos, hasta iniciar y cerrar sesión, por lo que también es imprescindible para entrar en la aplicación.

API KEY

Services ofrece una serie de opciones de seguridad. Por defecto, el acceso a la aplicación web se realiza sin ningún tipo de seguridad, es decir que cualquiera de los métodos activos podría ser utilizado para obtener o introducir datos en la aplicación web. Por ello es estrictamente necesario habilitar la autenticación de la clave API que requiere un conjunto de credenciales pre-creados, o de lo contrario la solicitud será rechazada.

También se puede habilitar la autenticación basada en sesiones, que requieren una llamada para comprobar que sigues siendo el mismo consumidor que había iniciado sesión previamente.

Por último, cada método del servicio, puede proveer de su propia comprobación de acceso, por lo que por cada método desarrollado se puede definir un acceso para un rol en particular. En nuestro caso será necesario dar permiso a los usuarios registrados para cada uno de los métodos creados en el módulo a desarrollar.

Se pueden crear tantas claves como sean necesarias, normalmente cada una de ellas tendrá asociados ciertos permisos, por lo que con una nos basta. Esta clave se tendrá que utilizar en la aplicación móvil, y en cada método propio del servicio web, por lo que se tendrá que pasar como un parámetro más.

KEY	TITLE	DOMAIN	OPERATIONS
34b2f0...e38e55	The Time Bird	thetimebird.atenealabs.com	edit delete

Ilustración 96: *API KEY* para ***The Time Bird***

5.2. Desarrollo librerías de comunicación

Una vez configurado el módulo **Services**, se procede al desarrollo de las librerías necesarias para las dos plataformas, posiblemente una de las tareas más complicadas de toda la fase de implementación.

5.2.1. Android

Para la elaboración de las librerías necesarias en Android, se han utilizado librerías de XMLRPC de Apache en Java, lo que nos ha facilitado el trabajo. Por otro lado, algunos de los métodos que proporciona el módulo *Services* en la aplicación web, requieren que los

datos lleguen de una forma determinada desde el cliente exterior, por ello, métodos como `system.connect` y `system.login` necesitarán una estructura de datos especial, algo que *HashMap*³⁶ ha podido suplir. Hay que destacar que la elaboración de estas librerías no ha sido una tarea fácil, al no saber en ciertas ocasiones si algunos errores encontrados (como la denegación de acceso) los devolvía la aplicación web o la aplicación móvil.

La elaboración de las librerías, como se ha comentado, ha supuesto ciertos problemas de diseño que finalmente han sido atajados. Para hacer las librerías lo más modulares posible, se han creado dos clases (initialmente fueron tres, aunque una de ellas fue descartada).

- **DrupalXmlRpcService:** Esta clase es la más importante. Contiene las funciones que inicializan los datos básicos para la conexión con Drupal, es decir, todos los parámetros necesarios para poder utilizar cualquiera de los métodos que se hayan desarrollado.
- **DrupalServiceCommands:** Esta clase está desarrollada con el propósito de tener presente los métodos que se están utilizando actualmente. Para ello se ha utilizado un tipo enumerado³⁷ que llamamos cada vez que llamamos al cliente XMLRPC desde la clase **DrupalXmlRpcService**.

Excepto para los métodos `system.connect` y `user.login`, el proceso seguido para cada uno de los métodos es el mismo. El primero de ellos, solo será necesario que se utilice durante el inicio de sesión, y sirve para obtener la identificación de un usuario como anónimo. Con este identificador, se puede utilizar el segundo método, permitiendo así autenticar al usuario que quiera utilizar el resto de métodos disponibles, ya que podríamos definir el resultado de estos dos métodos como un identificador de sesión, necesario para poder entrar en el sistema.

Por temas de seguridad, y para evitar que la sesión de un usuario haya caducado, se seguirá el mismo proceso cada vez que se quiera utilizar uno de los métodos a desarrollar. Primero se realizará un inicio de sesión. Después se usará al método deseado, y finalmente se cerrará la sesión. A continuación podemos encontrar un esquema que ilustra perfectamente el funcionamiento de conexión con la aplicación web.

³⁶ Matriz multidimensional utilizada en Java, que consta de claves y valores. La principal ventaja de esta estructura es la posibilidad de acceder rápidamente a sus elementos.

³⁷ Los tipos enumerados sirven para restringir el contenido de una variable a una serie de valores predefinidos. Esto suele ayudar a reducir los errores en nuestro código.

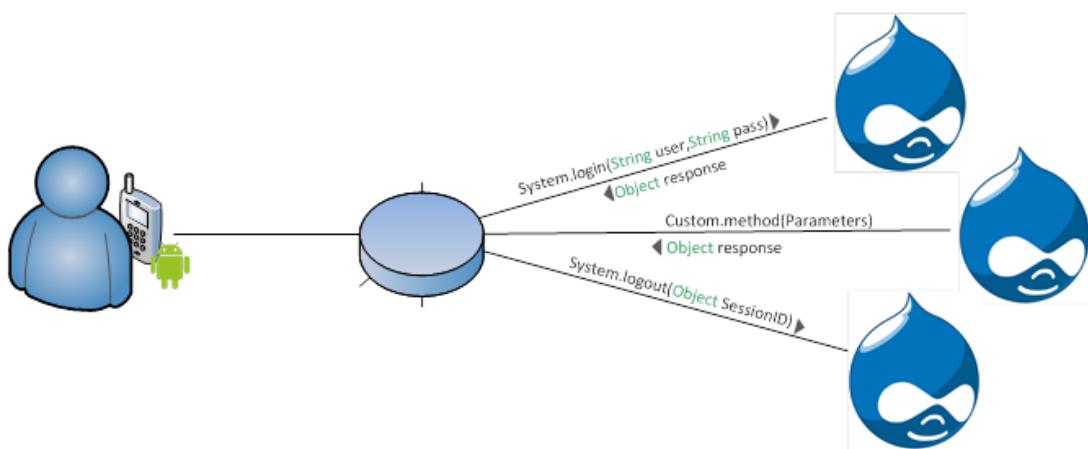


Ilustración 97: Proceso seguido a la hora de hacer una llamada a la aplicación web

Un aspecto importante en el diseño de los métodos que permiten realizar las funcionalidades descritas en la especificación, es la utilización de simples cadenas de caracteres como parámetros de entrada. De esta forma evitamos tener que enviar una estructura compleja de datos a tratar por la aplicación web, por lo que se reduce el tamaño de los paquetes de datos que se envían.

Además de los parámetros propios de cada funcionalidad, en cada una de las llamadas a la aplicación web, se necesitarán preparar otros cinco parámetros obligatorios.

- `hash`: Variable formada por una tira de bytes computados por un algoritmo de encriptación, combinando la *API KEY*, el `timestamp` y `nonce`. Es utilizado como validador, a la hora de realizar cualquier llamada a la aplicación web, mediante *Services*.
- `serviceDomain`: Dominio del sitio web al que se quiere acceder, en este caso `thetimebird.com`.
- `timestamp`: Parámetro que contiene el tiempo actual en segundos.
- `nonce`: parámetro formado por una tira de caracteres aleatorio y que sirve como representación del tiempo actual, en milisegundos. Este parámetro, combinado con algunos otros, sirve para la generación del parámetro `hash`.
- `sessionID`: Parámetro con la sesión del usuario, necesario para realizar cualquier llamada a la aplicación web. La obtención de este parámetro podrá variar en función del rol que tenga el usuario. Si es un usuario anónimo, se obtendrá mediante la llamada a `system.connect`, resultado que sirve para realizar el inicio de sesión mediante el método `user.login`, que nos dará un nuevo valor, necesario para poder trabajar con la aplicación web como usuario registrado.

5.3. Desarrollo de funcionalidades principales

Una vez con el esqueleto de las librerías necesarias para la comunicación de datos entre aplicaciones, se puede empezar a desarrollar todas las funcionalidades que tendrá el sistema.

A modo de destacar las tareas que han requerido un mayor esfuerzo, se va a proceder a describir el trabajo llevado a cabo para implementar las principales funcionales en ambas plataformas. Funcionalidades secundarias, como la creación de bloques y páginas de información para el usuario, o información de usuario implícita en Drupal, no se han considerado. Tampoco se ha descrito la edición de etiquetas, ya que es una funcionalidad simple y el proceso seguido es muy parecido al de crear una etiqueta.

Un aspecto en común de la mayoría de funcionalidades descritas, es el uso de una estructura que haga de cola, para así procesar las peticiones en orden de llegada. Estas peticiones se irán encolando en un `ArrayList` en un hilo de ejecución, de esta forma se pueden seguir realizando acciones en el dispositivo, mientras éste vaya enviando datos a la aplicación web en *background*.

5.3.1. Inicio de sesión

El inicio de sesión, es una funcionalidad que nos permite diferenciar a los usuarios autenticados de los anónimos en ***The Time Bird***, restringiendo de este modo las funcionalidades que se puedan llevar a cabo.

Para el desarrollo de esta funcionalidad, es necesario un método que permita a la aplicación móvil, enviar el usuario y contraseña a la aplicación web, que es la que contiene toda la lógica de negocio, incluyendo la gestión de datos. Para ello, se describe el trabajo realizado en las dos aplicaciones.

Aplicación web

El módulo *Services* por defecto provee de ciertos métodos básicos para la comunicación de aplicaciones externas con Drupal, y una de ellas es el inicio de sesión, por lo que será necesaria su activación. De esta forma, en la aplicación web ya tendremos la lógica necesaria para realizar el inicio de sesión de usuario una vez realizada la llamada desde una aplicación externa.

Call method		
NAME	REQUIRED	VALUE
Hash	required	Gets generated after form submission
Domain name	required	thetimebird.atenealabs.com
Timestamp	required	Gets generated after form submission
Nonce	required	EECg58QH8R
Session ID	required	9741b88ae325c841df4acb0b468cd7c9
username	required	
password	required	

Ilustración 98: Parámetros necesarios para iniciar la sesión de usuario

Aplicación móvil

Lo único que tenemos que pasarle a Drupal, mediante el Cliente *XMLRPC*, son los campos de texto de usuario y contraseña que el cliente introduzca en la aplicación Android:

```
public boolean login(String username, String password)
```

Esta función devolverá un booleano indicando si se ha podido realizar el inicio de sesión en la aplicación correctamente.

Hay que recordar que el inicio de sesión se tendrá que realizar siempre que se quiera utilizar cualquier otra funcionalidad desde la aplicación móvil, como se ha dicho anteriormente, por seguridad y para prevenir posibles caducidades de sesiones.

5.3.2. Listar etiquetas

La funcionalidad de listar etiquetas, será implícita para el usuario, ya que se realiza junto al inicio de sesión del mismo. El proceso de listar las etiquetas del usuario, se puede dividir en la parte del trabajo realizado en la aplicación móvil y el realizado en la aplicación web.

Aplicación móvil

Una vez validado el usuario que acaba de iniciar sesión en la aplicación web, se llama al método de la librería **DrupalXmRpcService**, que permite listar las etiquetas del usuario conectado:

```
public ArrayList<String> getTags()
```

Una vez devuelta la lista de etiquetas, se cambia de Actividad, iniciando así la pantalla principal de etiquetas con la función `createView()` que realiza un bucle para cada una de

ellas, pintándolas en una tabla de botones divididos en dos columnas. Al tener en la lista constancia de las etiquetas que están activas, si se encuentra alguna en dicho estado, el botón se pintará de color naranja, y en caso contrario de color gris.



Ilustración 99: Lista de etiquetas en la aplicación móvil

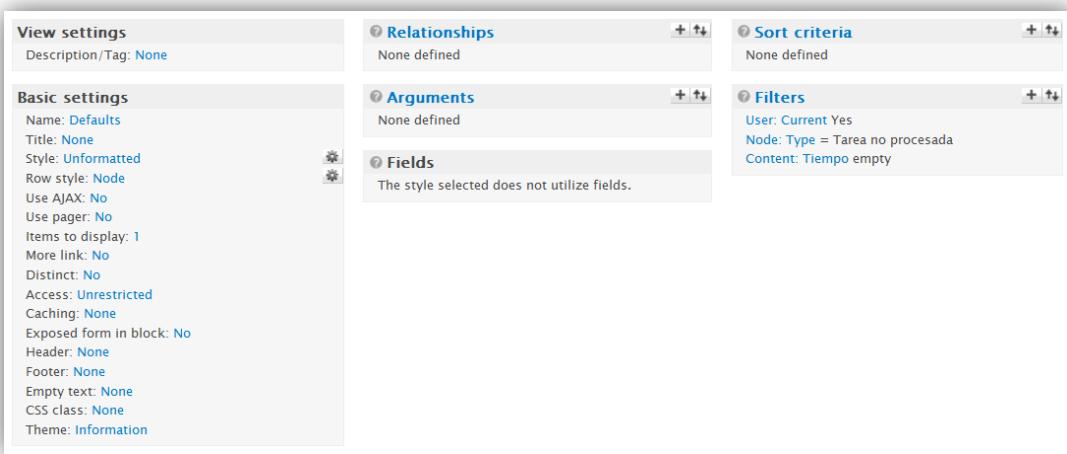
Aplicación web

Cuando Drupal recibe los datos, se llama al método `TTB_service` del módulo TTB, que se encarga de redirigir la petición a la función `tag_list()`. Esta función se encarga de buscar todas las etiquetas habilitadas por el usuario, mediante la vista **`tag_list`**.

Ilustración 100: Vista `tag_list`

Como se puede observar, la consulta filtra por tipo de contenido **Etiqueta** y permite que se pasen ciertos argumentos, como el identificador de usuario o el estado de la etiqueta, por lo que si se reciben estos parámetros, como en este caso, utilizando el identificador del usuario que hace la petición, y el estado 1 (habilitada), también se añadirán al filtro.

También necesitamos saber el estado de las etiquetas que estén activadas en este momento. Para ello necesitamos el uso de otra vista llamada **`last_unprocessed`**, que nos indique si existe alguna **Tarea no procesada** actualmente, y si es así poder obtener de ella las etiquetas activadas.

Ilustración 101: Vista *last_unprocessed*

En esta consulta a la base de datos, se filtran los nodos de tipo **Tarea no procesada**, el usuario actual y que el campo de tiempo final de la tarea, esté vacío, lo que indicará que es la única en la que el usuario pueda estar trabajando en ese preciso momento.

Una vez con la lista de etiquetas activas obtenidas de la tarea y la lista de etiquetas obtenidas de la vista **tag_list**, se añaden a una misma estructura y se retorna una lista que contiene las etiquetas obtenidas de la primera vista, y las obtenidas de la segunda.

5.3.3. Crear etiqueta

Una de las principales bazas de **The Time Bird**, es la posibilidad de trabajar simultáneamente con ciertas etiquetas. Éstas, representan diversos conceptos, como clientes, proyectos o procesos en los que el usuario esté trabajando.

El proceso de creación de una etiqueta se puede dividir en el trabajo realizado por la aplicación móvil y la aplicación web.

Aplicación móvil

Una vez el usuario ha seleccionado crear una nueva etiqueta en el menú, se abre una ventana emergente para que pueda introducir el nombre deseado. Una vez introducido, se comprueba que no haya ningún proceso en marcha en la cola de peticiones y se encola el proceso de creación de etiquetas. Una vez activo, se llama al método correspondiente de la librería **DrupalXmlRpcService**, que permita enviar los datos necesarios a la aplicación web.

```
public int createTag(String name)
```

Esta función devolverá un entero, con el que sabremos qué tipo de acción se ha realizado en la aplicación web. En el caso que se haya creado correctamente la etiqueta, se mostrará el nuevo botón por pantalla, llamando a la función `createView()` explicada anteriormente, o mostrará mensajes de advertencia correspondientes al proceso seguido en la aplicación web.

Aplicación web

Para que no haya una carga masiva de etiquetas en la pantalla de la aplicación móvil, al usuario se le permite deshabilitar las etiquetas creadas con anterioridad, y así poder trabajar de una forma mucho más fluida. Hay que controlar que el usuario no cree etiquetas existentes y activas, o que estén deshabilitadas. Por ello, cuando Drupal recibe los datos, se llama al método `TTB_service` del módulo TTB, que se encarga de redirigir la petición a la función `create_tag()`, que se sigue el siguiente proceso.

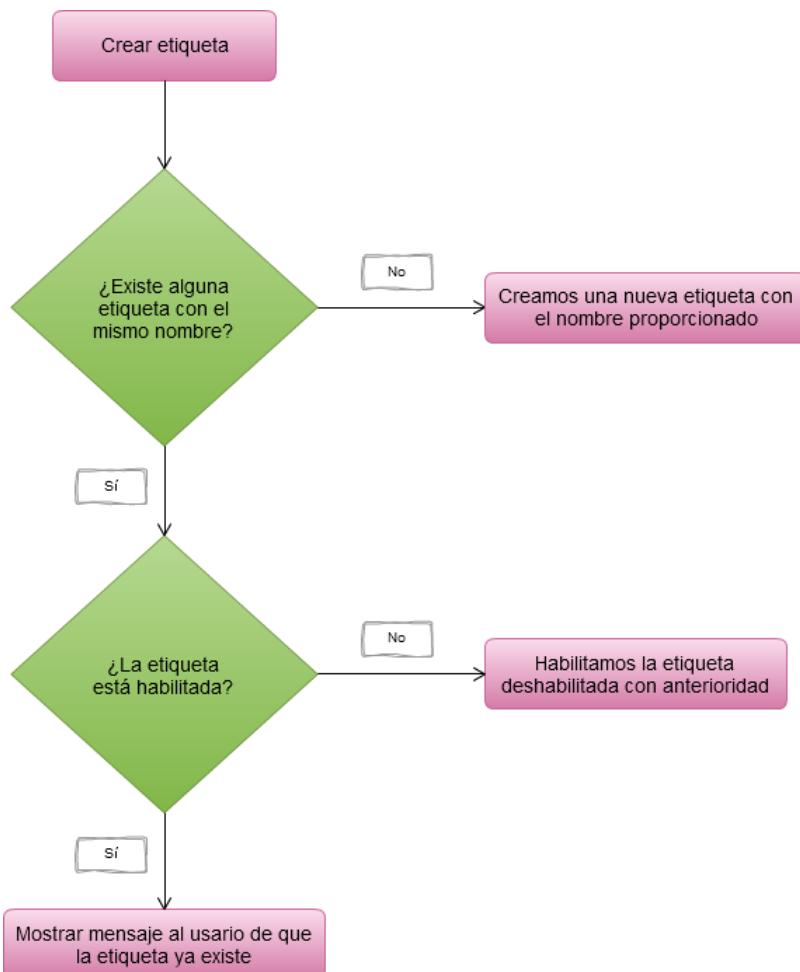


Ilustración 102: Proceso de creación de una etiqueta en la aplicación web

Para el proceso descrito, es necesario el uso de ciertas vistas. La primera de ellas de nombre **tag**, sirve para comprobar si existe en la base de datos una etiqueta con el mismo nombre que el parámetro de entrada.

The screenshot shows the configuration interface for a Drupal view named 'tag'. On the left, under 'Basic settings', various options like 'Title', 'Style', and 'Items to display' are set to their default values. In the center, the 'Arguments' section is configured to accept a 'Node: Title' argument. On the right, the 'Filters' section includes a condition where the node type must be 'Etiqueta'. This setup ensures that the view only retrieves nodes of type 'Etiqueta'.

Ilustración 103: Vista tag

Como se puede observar, se le pasa el nombre de la etiqueta como argumento, que actúa como filtro, por lo que, si existe una etiqueta en la base de datos con el nombre pasado como parámetro, devolverá el nodo correspondiente a la etiqueta.

Una vez con los resultados, sabiendo que podamos tener una etiqueta existente, se continúa el proceso seguido en el diagrama de la ilustración 102. Finalmente se devolverá un número entero, siendo 0 en el caso que se haya intentado crear una etiqueta ya creada y habilitada, 1 en el caso que se intente crear una etiqueta deshabilitada, y el número al nodo correspondiente en el caso que se haya creado la etiqueta correctamente.

5.3.4. Activar etiqueta

Una vez se ha accedido a la pantalla principal de la aplicación y se han listado las etiquetas, todas ellas pueden ser activadas y desactivadas, siempre dependiendo del estado en el que se dejaron todas ellas la última vez que se realizaron tareas.

El proceso de activación de una etiqueta se divide en el trabajo realizado por la aplicación móvil y la aplicación web.

Aplicación móvil

Una vez el usuario ha apretado el botón correspondiente a una etiqueta desactivada, la aplicación móvil comprueba que no haya ningún proceso en marcha en la cola de

peticiones y se encola el proceso de activación de etiquetas. Una vez activo, se llama al método correspondiente de la librería **DrupalXmRpcService**, que permita enviar los datos necesarios a la aplicación web.

```
public boolean tagOn(String name)
```

Esta función devolverá un booleano indicando si en la aplicación web se han realizado los cambios correctamente. En caso afirmativo se mostrará un mensaje por pantalla indicando que la etiqueta se ha activado correctamente, y en caso contrario se mostraría otro aviso indicando que ha habido un problema (se indica cuál) y se volvería a pintar la etiqueta del color del estado anterior correspondiente.

Aplicación web

Cuando Drupal recibe datos, se llama al método `TTB_service` del módulo TTB, que se encarga de redirigir la petición a la función `tag_on()`, en la que se sigue el siguiente proceso.

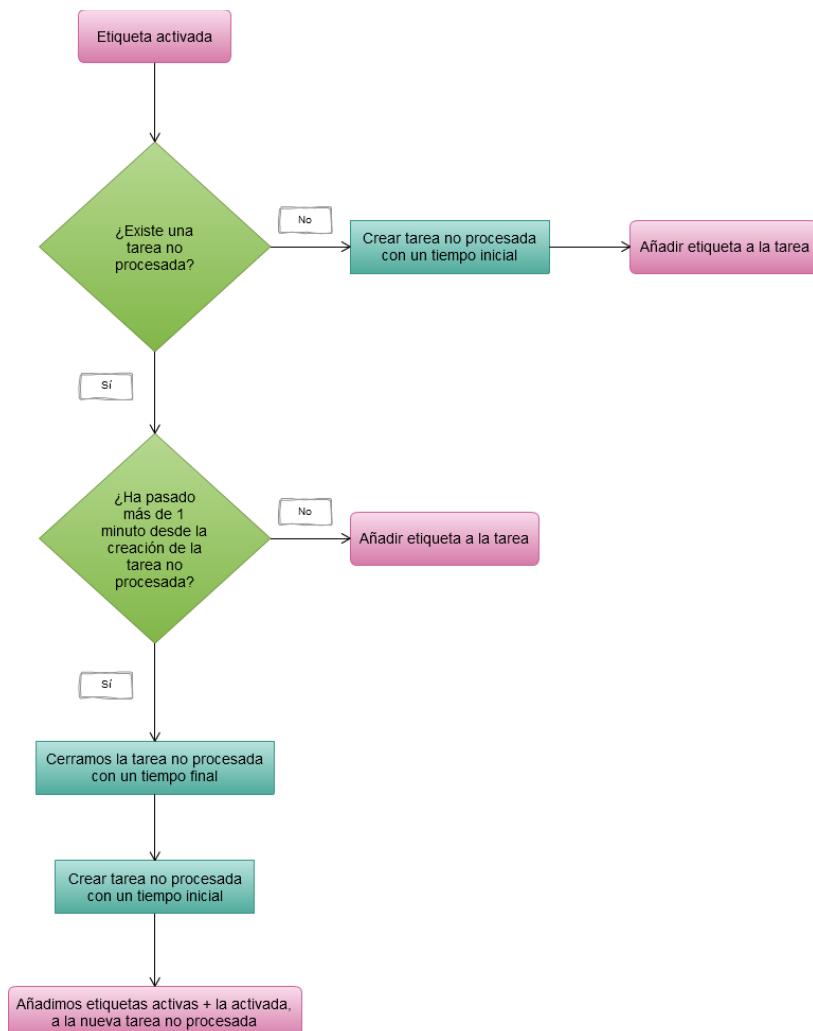


Ilustración 104: Proceso de activación de una etiqueta en la aplicación web

Para el proceso descrito, es necesario el uso de ciertas vistas. La primera de ellas sirve para buscar la etiqueta a partir del nombre recibido, en la base de datos. Para ello se utiliza la vista **tag**, descrita con anterioridad. También se utiliza la vista **last_unprocessed**, que permite saber si existe alguna **Tarea no procesada** y realizar los cambios correspondientes descritos en el proceso de la ilustración anterior.

En cuanto al proceso, cabe detallar que, al realizar cambios de tarea por fracciones de minuto, una vez activada una etiqueta, cabe la posibilidad de que no exista ninguna **Tarea no procesada** actual, lo que indica que no hay ninguna **Etiqueta** activada actualmente. Si fuera así se crea una nueva tarea y se añade la etiqueta correspondiente. En caso contrario, se tiene que comprobar si ha pasado más de un minuto desde que se creó la tarea, y de no ser así se podrán seguir realizando cambios en la misma **Tarea no procesada**, pero en caso contrario, se cerrará la tarea con su correspondiente tiempo final y se creará una nueva con las etiquetas activadas anteriormente junto con la que se acaba de añadir.

Finalmente se devuelve el identificador del nodo de la tarea que se acaba de crear o modificar.

5.3.5. Desactivar etiqueta

Esta funcionalidad funciona de forma parecida a la activación de etiquetas, aunque sigue un proceso algo diferente. También es necesario dividir el trabajo a realizar entre la aplicación móvil y la aplicación web.

Aplicación móvil

Una vez el usuario ha apretado el botón correspondiente a una etiqueta activada, la aplicación móvil comprueba que no haya ningún proceso en marcha en la cola de peticiones y se encola el proceso de desactivación de etiquetas. Una vez activo, se llama al método correspondiente de la librería **DrupalXmlRpcService**, que permita enviar los datos necesarios a la aplicación web.

```
public boolean tagOff(String name)
```

Esta función devolverá un booleano indicando si en la aplicación web se han realizado los cambios correctamente. En caso afirmativo se mostrará un mensaje por pantalla indicando que la etiqueta se ha desactivado correctamente.

Aplicación web

Una vez Drupal recibe los datos, se llama al método `TTB_service` del módulo TTB, que se encarga de redirigir la petición a la función `tag_off()`, en la que se sigue el siguiente proceso.

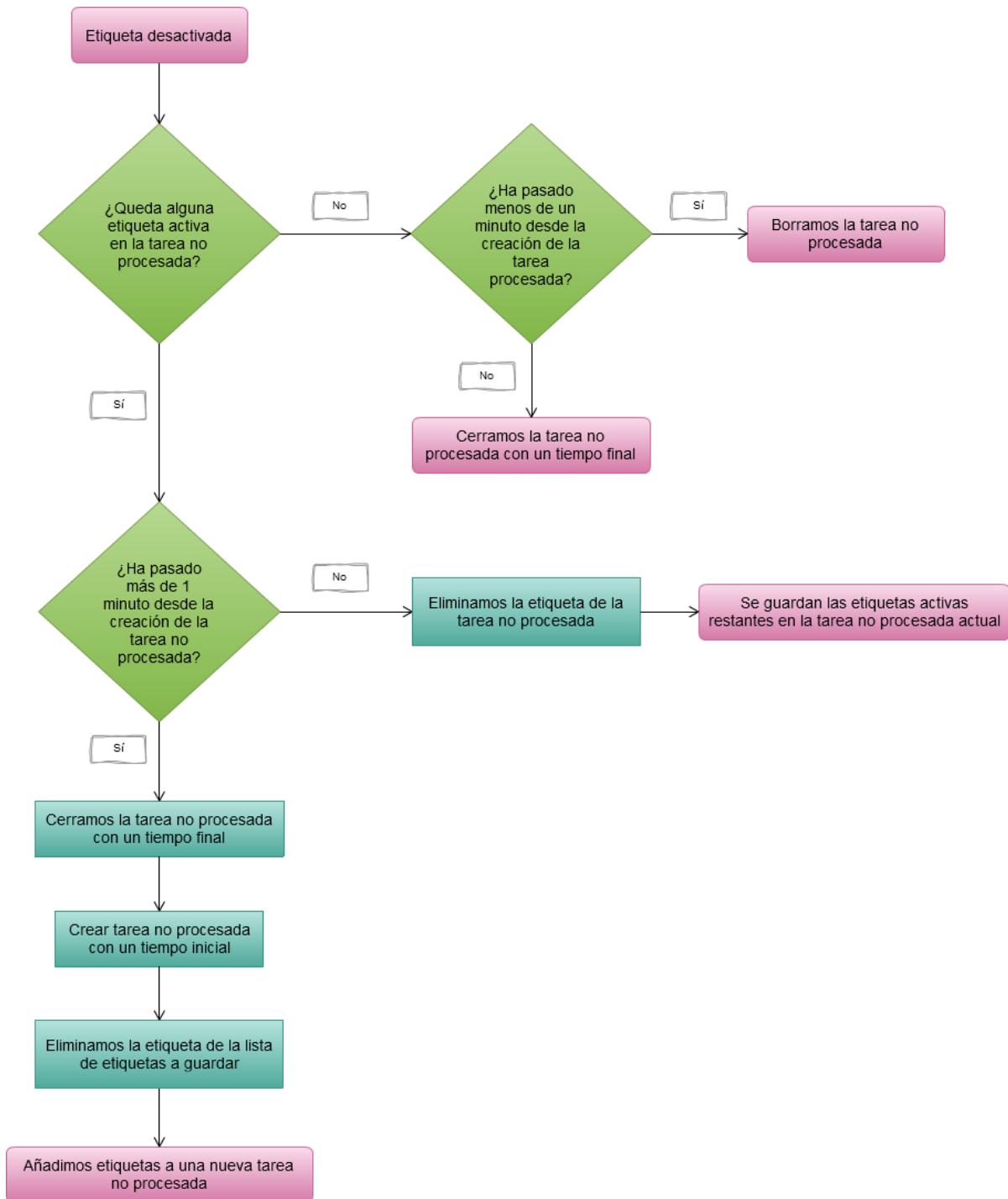


Ilustración 105: Proceso de desactivación de una etiqueta en la aplicación web

Del mismo modo que a la hora de activar una etiqueta, durante este proceso es necesario el uso de las vistas **tag** y **last_unprocessed** para la obtención de la **Etiqueta** a desactivar y la tarea actual, respectivamente.

En cuanto al proceso seguido, cabe detallar que es un poco más complejo que el seguido a la hora de activar etiquetas. Lo primero que se comprueba es que no sea la última **Etiqueta** activa en una **Tarea no procesada**. En caso de no serlo, si ha pasado menos de un minuto desde su creación, al dejarle al usuario un margen de error de un minuto, siendo el tiempo inicial el tiempo de creación de la tarea, ésta sería borrada , pero si ha pasado un minuto se cerrará la tarea con un tiempo final.

En caso de no ser la última **Etiqueta** activa en la **Tarea no procesada** y ha pasado más de un minuto desde su creación, se elimina de la lista de etiquetas activas en la tarea y se guardan los cambios realizados. En caso contrario, se cierra la tarea actual con un tiempo final correspondiente a la hora actual, y se crea una nueva **Tarea no procesada**, con una nueva lista de etiquetas activadas, habiendo excluido la actual.

5.3.6. Deshabilitar etiqueta

La opción de deshabilitar ciertas etiquetas, se ha dado con el fin de no cargar con muchos botones la pantalla principal. Hay que pensar que si un usuario lleva trabajando un tiempo con la aplicación, pueda tener más de 20 o 30 etiquetas, por lo que es una buena opción de ir deshabilitando las etiquetas antiguas a medida que no se vayan utilizando.

Cabe mencionar que se ha descartado la opción de eliminar las etiquetas creadas, ya que de esta forma siempre se podrá acceder a estadísticas de etiquetas usadas con anterioridad. Si eliminásemos la etiqueta, no sería posible su acceso.

Su proceso, al igual que las funcionalidades anteriores, está dividido en el trabajo realizado por la aplicación móvil y la aplicación web.

Aplicación móvil

Una vez el usuario ha seleccionado deshabilitar una **Etiqueta**, mediante el menú contextual de un botón correspondiente a la misma, se comprueba que no haya ningún proceso en marcha en la cola de peticiones y se encola el proceso de deshabilitar la **Etiqueta**. Una vez activo, se llama al método correspondiente de la librería **DrupalXmI RpcService**, que permita enviar los datos necesarios a la aplicación web.

```
public int disableTag(String name)
```

Esta función devolverá un entero, representando al identificador del nodo de la **Etiqueta**

deshabilitada. En el caso que no se hayan realizado los cambios en la aplicación web, de forma correcta, mostraremos un mensaje por pantalla indicándoselo al usuario.

Aplicación web

Una vez Drupal recibe los datos, se llama al método `TTB_service` del módulo TTB, que se encarga de redirigir la petición a la función `disable_tag()`, en la que se sigue el siguiente proceso.

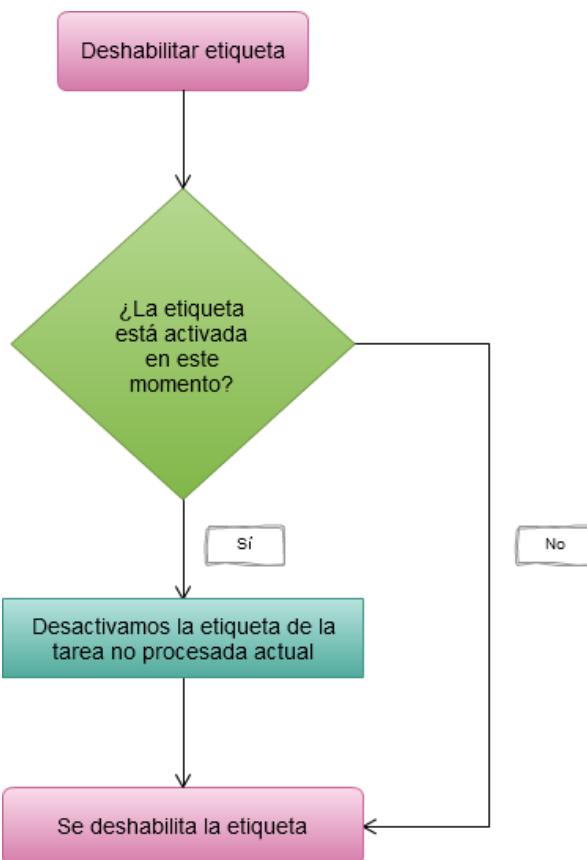


Ilustración 106: Proceso de deshabilitación de una etiqueta en la aplicación web

Para el proceso seguido, es necesario el uso de las vistas **tag** y **last_unprocessed**.

Durante el proceso, se comprueba si la **Etiqueta** recibida se encuentra activada. Si es así, hay que desactivarla antes de deshabilitarla, por lo que se realizará una llamada al método correspondiente de desactivación de etiquetas `tag_off()`. Seguidamente se cambia el campo correspondiente al estado de la etiqueta a 0 y se guardan los cambios realizados. En caso que no estuviera activada, solo se realizaría el cambio de estado de etiqueta y se guardarían los cambios realizados en la propia etiqueta.

5.3.7. Proceso de análisis mediante Cron

Como se ha comentado en la descripción de los elementos más importantes de Drupal, el Cron es una herramienta que se ejecuta periódicamente con la finalidad de realizar diferentes tipos de tareas básicas de mantenimiento del sitio web. En el caso de **The Time Bird**, el Cron se utiliza por los siguientes motivos:

- El proceso de análisis, en el que se dividirán los datos recogidos por el usuario para facilitar y mejorar la rapidez de su posterior lectura durante la creación de los gráficos de las estadísticas seleccionadas.
- Permite realizar tareas cada cierto tiempo deseado, por lo que se podrá realizar todo el análisis estadístico con la frecuencia en que los administradores de la aplicación vean necesaria.

El tiempo establecido para la ejecución del proceso correspondiente al cron, es de **5** minutos por dos motivos principales:

- Es una ventaja para los usuarios que en apenas cinco minutos después de acabar sus tareas, ya puedan contemplar las estadísticas correspondientes a todos sus datos. Existen otros servicios como el de analíticas de las web de Google, o las del mercado de Android, que solo permiten su obtención durante el día posterior.
- Si se realizara este proceso en un periodo superior a cinco minutos, habría el riesgo de que muchos usuarios acaban sus tareas y podría haber un consumo de memoria elevado en el servidor, ya que en muchos casos, es necesaria la creación de muchos periodos y otro tipo contenido que puede llegar a sobrecargar el sistema.

Para la realización de todo el proceso, hace falta llamar a la función *hook_cron*, que es la función que Drupal llamará cada periodo de tiempo definido por el administrador (5 minutos en nuestro caso).

El proceso seguido consta en la transformación de la información guardada en las tareas no procesadas, en tareas procesadas. A continuación se puede observar el proceso seguido de forma gráfica.

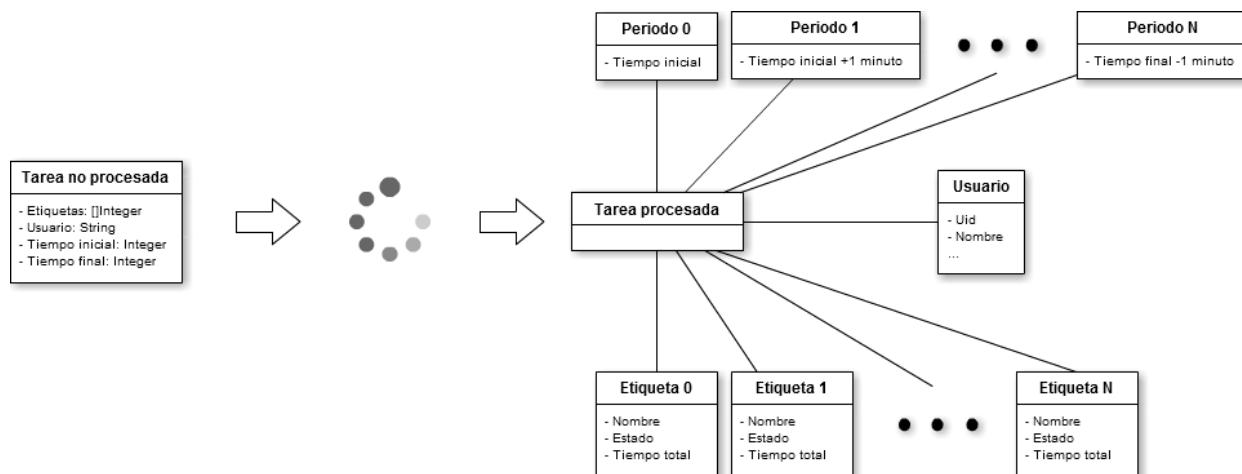


Ilustración 107: Proceso de transformación de tareas mediante Cron

Como se puede observar, el resultado de todo el proceso es la creación de tareas procesadas que guardan las referencias al usuario que ha creado la tarea, a las etiquetas usadas y los períodos utilizados en dicha tarea.

El proceso de análisis, siempre se ejecuta de la misma forma. Se recogen todas las tareas no procesadas existentes en el sistema, mediante la vista ***task_transformation_process***.

View settings Description/Tag: None	Relationships None defined	Sort criteria None defined
Basic settings Name: Defaults Title: None Style: Unformatted Row style: Node Use AJAX: No Use pager: No Items to display: Unlimited More link: No Distinct: No Access: Unrestricted Caching: None Exposed form in block: No Header: None Footer: None Empty text: None CSS class: None Theme: Information	Arguments None defined	Filters Node: Type = Tarea no procesada Content: Tiempo not empty

Ilustración 108: Vista ***task_transformation_process***

Para cada una de las tareas, se realizan los cálculos necesarios para obtener tantos períodos como minutos dedicados a la elaboración de esa tarea. Cuando más de un usuario está realizando una tarea, no hará falta crear los períodos correspondientes más de una vez. Se utilizará la vista ***period_exist*** para comprobar en la base de datos si existe un periodo con el nombre que se le pasa como parámetro, de tal forma que si así

fueras, simplemente se referenciaría al periodo ya existente y no haría falta la creación de un periodo igual. De esta forma no se llena tanto la base de datos.

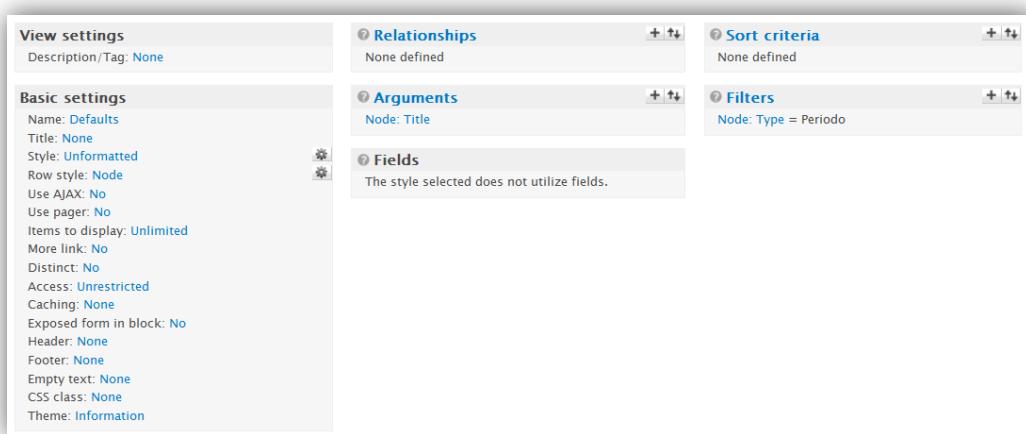


Ilustración 109: Vista *period_exist*

Una vez referenciados los periodos, cabe referenciar también las etiquetas utilizadas, las cuales se van obteniendo y a la vez actualizando con el tiempo total dedicado a cada una ellas, mediante una variable que ha sumado un minuto por cada uno de los periodos a enlazar, así siempre se puede obtener el total de tiempo dedicado a una **Etiqueta**.

Una vez enlazadas etiquetas y periodos, se crea la tarea procesada y se guarda también una referencia del usuario que ha creado la tarea.

Realizado todo el proceso, se procede a borrar la **Tarea no procesada**. Todas las tareas procesadas, no vuelven a ser analizadas, ya que no tendría ningún sentido.

5.3.8. Estadísticas

Las estadísticas en forma de gráficos sobre los datos recogidos de los usuarios es una de sus bazas principales de **The Time Bird**. Las estadísticas, serán combinaciones de datos mediante tres variables; **tiempo, usuarios y etiquetas**.

Se ofrecen cinco tipo de combinaciones estadísticas, donde cada una mostrará gráficos de diferentes tipos

Las estadísticas, estarán formadas por la combinación de las variables nombradas, y así formar cinco funcionalidades mediante las que se puede obtener diferente información respecto a ciertos usuarios y/o etiquetas.

Cabe tener en cuenta que siempre trabajaremos con nodos de tipo **Tarea no procesada**, ya que de esta forma se conseguirán todos los demás tipos de contenido referenciados por la tarea.

Antes de proceder a la explicación de cada una de las estadísticas, cabe destacar un proceso llevado a cabo a la hora de filtrar datos que se ha realizado por la limitación que ofrece el módulo *Views*.

Este proceso consta de dos partes, siendo ciertas modificaciones aplicadas a los filtros expuestos que ofrece el módulo *Views* para la posterior consulta a la base de datos.

- Filtro de usuarios: **The Time Bird** da la posibilidad de que usuarios vean datos de otros usuarios a los que le han concedido el permiso para obtener sus datos, y así sucesivamente creándose una jerarquización de usuarios. El proceso de obtención de todos estos usuarios no es sencillo. Para ello se necesita un algoritmo que recorra en profundidad cada usuario que nos ha dado permiso para ver sus datos, y así hasta llegar al máximo número de usuarios registrados o se hayan explorado todos los caminos posibles. El número de usuarios del primer sobre el que se empezará la búsqueda, viene dado por los resultados de una vista creada para ello o la selección de una cantidad de usuarios mediante el filtro expuesto para ello. A continuación se describe el proceso realizado.

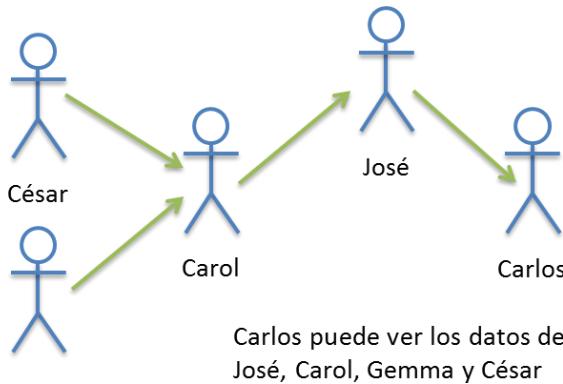


Ilustración 110: Relación de permisos para obtención de datos, entre usuarios

Para tratar con los formularios expuestos, será necesario trabajar con el *hook TTB_form_views_exposed_form_alter*, que podemos encontrar en el módulo TTB, y que permite modificar los datos presentados en los formularios (en nuestro caso los filtros expuestos de la vista). También será necesario crear una vista, que nos permita obtener los perfiles de los usuarios que nos dejan obtener sus datos.

Ilustración 111: Vista *users_perm*

Una vez con los usuarios de primer nivel, hay que encontrar todos los de los demás niveles, y para ello se ha creado una función llamada `backtracking()` que precisamente, realice una búsqueda en profundidad de todos los usuarios de los que podemos sacar información. La lista de usuarios devuelta por la función, serán remplazados por los que había principalmente en el filtro, por lo que el usuario final no se puede percatar de los cambios introducidos.

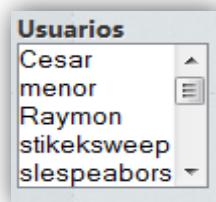


Ilustración 112: Filtro de usuarios

- **Filtro de etiquetas:** Dado que el filtro expuesto de etiquetas, lista todas las etiquetas de la aplicación web, es necesario solo listar las del usuario que lo esté utilizando, o en el caso de estar junto al filtro de usuarios, filtrar también todas las etiquetas de los usuarios seleccionados. Para ello, es necesario trabajar también con el hook `TTB_form_views_exposed_form_alter`, mediante el que podemos modificar los resultados de los campos expuestos. En este caso, mediante la vista ***tag_list***, podemos obtener la lista de etiquetas de la lista de identificadores que se le pase como parámetro. Si solo queremos nuestras etiquetas, se le pasa nuestro identificador. En el

caso que también tengamos un filtro de usuarios, la función `backtracking()` proporciona la estructura necesaria para obtener los identificadores de cada uno de ellos, y así poder obtener una nueva lista de etiquetas de todos los usuarios filtrados, que será sustituida por la lista total de etiquetas filtrada por defecto.

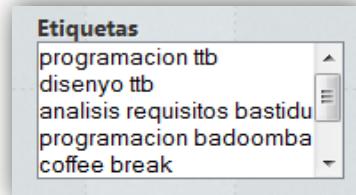


Ilustración 113: Filtro de etiquetas

Una vez visto los cambios que se tendrán que realizar en cada uno de los filtros mencionados, se puede proceder a describir cada una de las estadísticas. Para la realización de cada una de ellas, se ha utilizado una vista y una plantilla de tema diferente.

- **Etiquetas:** El usuario puede seleccionar cualquiera de sus etiquetas y puede obtener el tiempo total dedicado a cada una de ellas.

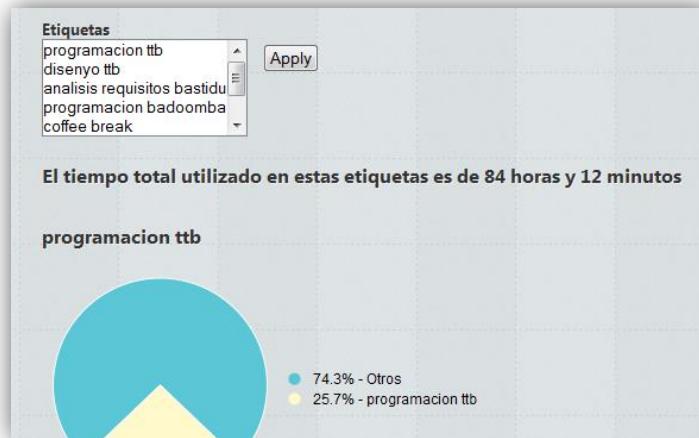


Ilustración 114: Resultados ya procesados de las estadísticas

El proceso seguido para crear las estadísticas, comienza por el uso de la vista ***statistics_tags*** que nos prepare los datos que serán enviados a la plantilla de tema `views-view-unformatted—statistics-tags.tpl.php` correspondiente a esta vista. En este

caso, solo hace falta filtrar nodos de tipo **Tarea procesada** que contengan las etiquetas seleccionadas del filtro expuesto, si se da el caso.

Ilustración 115: Vista *statistics_tags*

En la plantilla, tratamos cada una de las tareas filtradas, obteniendo el nombre de las etiquetas utilizadas y el tiempo total que se le ha dedicado a cada una de ellas. Una vez con estos datos se estructuran de tal forma que la *API* pueda generar correctamente el tipo de gráfico utilizado.

- **Etiquetas/Periodos:** El usuario puede seleccionar las etiquetas y el margen de tiempo deseado, hasta un máximo de 2 años en los filtros expuestos, y así poder obtener una gráfica con el progreso de tiempo dedicado para las etiquetas seleccionadas y un gráfico comparativo del tiempo dedicado entre las etiquetas seleccionadas.

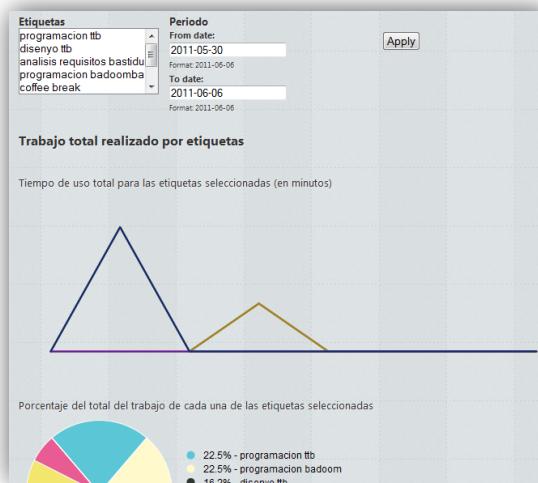
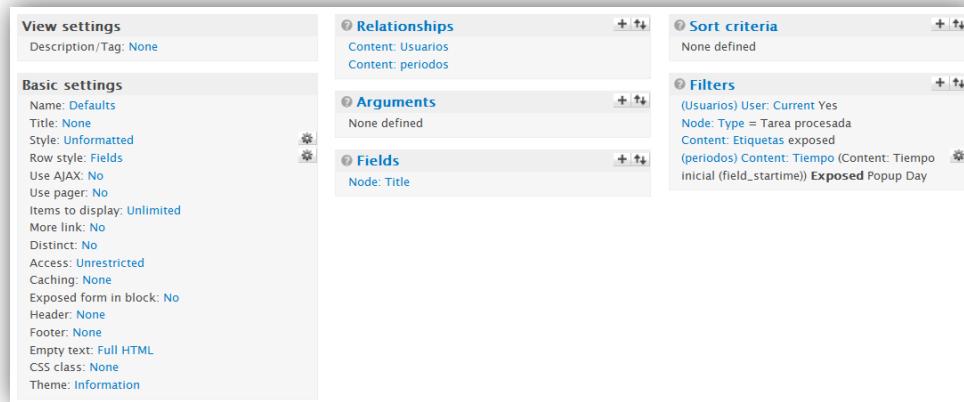


Ilustración 116: Resultados ya procesados de la estadística de etiquetas por períodos

El proceso seguido para crear las estadísticas es el mismo. Se requiere del uso de la vista ***statistics_tags_periods*** que nos prepare los datos que serán enviados a la plantilla de tema *views-view-unformatted—statistics-tags-periods.tpl.php* correspondiente a la vista nombrada. En este caso, se utilizan los filtros de tiempo y de etiquetas para obtener nodos de tipo **Tarea procesada**, por lo que los resultados variarán en función del filtro que se haya seleccionado.



The screenshot shows the configuration interface for a Drupal view named 'statistics_tags_periods'. The 'Basic settings' section includes fields like Name (Defaults), Title (None), Style (Unformatted), Row style (Fields), and various visibility and caching options. The 'Filters' section is expanded, showing a complex query: '(Usuarios) User: Current Yes AND Node: Type = Tarea procesada AND Content: Etiquetas exposed AND (periodos) Content: Tiempo (Content: Tiempo inicial (field_startime)) Exposed Popup Day'. Other sections like 'Relationships' and 'Sort criteria' are also visible.

Ilustración 117: Vista *statistics_tags_periods*

En la plantilla, tratamos cada una de las tareas filtradas, obteniendo por cada uno de los días filtrados, el número de minutos dedicados a las etiquetas de la tarea. Una vez con estos datos se estructuran de tal forma que la *API* pueda generar correctamente los tipos de gráfico utilizados.

- **Etiquetas/Usuarios:** El usuario puede seleccionar las etiquetas y los usuarios por los que desea filtrar los datos, y así poder obtener gráficas del tiempo total dedicado por cada usuario seleccionado, a cada una de las etiquetas filtradas.



Ilustración 118: Resultados ya procesados de la estadística de etiquetas por usuarios

El proceso seguido a la hora de crear estadísticas comienza por el uso de la vista **statistics_tags_users** que nos prepare los datos que serán enviados a la plantilla del tema *views-view-unformatted—statistics-tags-users.tpl.php* correspondiente a la vista nombrada. En este caso, se pueden utilizar los filtros de etiquetas y usuarios. Habrá tantos gráficos como etiquetas seleccionadas, y cada uno de ellos estará agrupado por todos los usuarios seleccionados.

The screenshot displays the configuration interface for the 'statistics_tags_users' view. On the left, under 'Basic settings', various options are listed such as Name: Defaults, Title: None, Style: Unformatted, Row style: Fields, and so on. In the center, there are several tabs: 'Relationships' (Content: Etiquetas), 'Arguments' (None defined), 'Fields' (listing '(Etiquetas) Node: Title', '(Etiquetas) Content: Tiempo 9999', and 'Content: Usuarios Default'), 'Sort criteria' (None defined), and 'Filters' (Node: Type = Tarea procesada, Content: Etiquetas exposed, Content: Usuarios exposed). Each tab has a '+' icon to add more items.

Ilustración 119: Vista *statistics_tags_users*

En la plantilla, tratamos cada una de las tareas filtradas, obteniendo por cada una de las etiquetas, y el tiempo total que le ha dedicado cada uno de los usuarios. Una vez con estos datos se estructuran de tal forma que la *API* pueda generar correctamente el tipo de gráfico utilizado.

- **Usuarios/Periodos:** El usuario puede seleccionar los usuarios y el margen de tiempo, hasta un máximo de dos años, y así obtener una gráfica con el progreso del tiempo total dedicado a todas sus etiquetas, así como un gráfico comparativo indicando cuáles de entre los usuarios seleccionados, ha trabajado una mayor porción de tiempo.

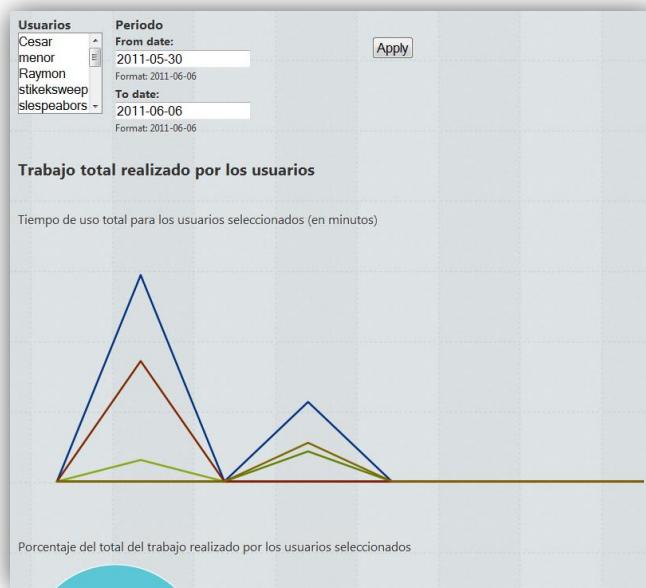


Ilustración 120: Resultados ya procesados de la estadística de usuarios por períodos

El proceso seguido para crear las estadísticas, empieza por el uso de la vista ***statistics_users_periods*** que nos prepare los datos que serán enviados a la plantilla de tema *views-view-unformatted—statistics-users-periods.tpl.php* correspondiente a la vista nombrada. En este caso, se pueden utilizar los filtros de usuarios y períodos, por lo que se generarán dos gráficos agrupando el tiempo total dedicado a todas las etiquetas de cada uno de los usuarios.

The screenshot shows the configuration interface for the 'statistics_users_periods' view. It includes sections for 'View settings' (Description/Tag: None), 'Basic settings' (Name: Defaults, Title: None, Style: Unformatted, Row style: Fields, Use AJAX: No, Use pager: No, Items to display: Unlimited, More link: No, Distinct: No, Access: Unrestricted, Caching: None, Exposed form in block: No, Header: None, Footer: None, Empty text: Full HTML, CSS class: None, Theme: Information), 'Relationships' (Content: períodos), 'Arguments' (None defined), 'Fields' (Node: Nid, Content: Usuarios Default), 'Sort criteria' (None defined), and 'Filters' (Node: Type = Tarea procesada, Content: Usuarios exposed (períodos) Content: Tiempo (Content: Tiempo inicial (field_starttime)) Exposed Popup Day).

Ilustración 121: Vista *statistics_users_periods*

En la plantilla, tratamos cada una de las tareas filtradas, obteniendo por cada uno de

los usuarios, el tiempo dedicado a sus etiquetas en los días seleccionados, pudiendo seguir así el progreso del tiempo que ha dedicado el usuario a realizar las tareas día a día. Una vez con estos datos se estructuran de tal forma que la *API* pueda generar correctamente el tipo de gráfico utilizado.

- **Etiquetas/Usuarios/Periodos:** El usuario puede seleccionar cualquiera de sus etiquetas, usuarios y el margen de tiempo, deseado hasta un máximo de 2 años, y así obtener gráficas con el progreso del tiempo dedicado a cada una de las etiquetas por cada usuario seleccionado.



Ilustración 122: Resultados ya procesados de la estadística de etiquetas por usuarios por períodos

El proceso seguido para crear las estadísticas, empieza por el uso de la vista **statistics_tags_users_periods** que nos prepare los datos que serán enviados a la plantilla *views-view-unformatted—statistics-tags-users-periods.tpl.php* correspondiente a la vista nombrada. En este caso, se pueden utilizar los filtros de etiquetas, usuarios y períodos. Habrá tantos gráficos como etiquetas seleccionadas, y cada uno de ellos estará agrupado por todos los usuarios seleccionados.

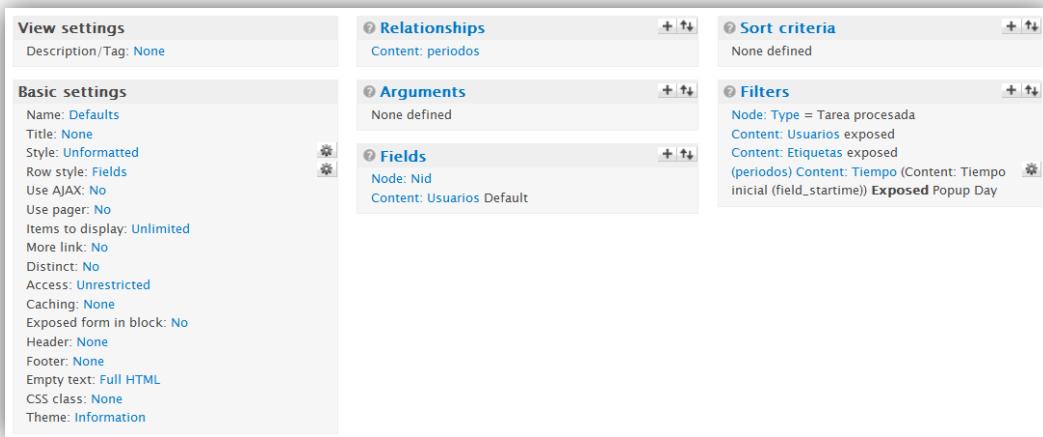


Ilustración 123: Vista *statistics_tags_users_periods*

En la plantilla, tratamos cada una de las tareas filtradas, obteniendo por cada una de las etiquetas, el tiempo dedicado por los usuarios seleccionados en los días seleccionados, pudiendo así seguir el progreso de tiempo que ha dedicado cada usuario a cada una de las etiquetas. Una vez con estos datos se estructuran de tal forma que la API pueda generar correctamente el tipo de gráfico utilizado.

5.4. Diseño gráfico

El diseño gráfico utilizado, se puede dividir en dos partes, siendo éstas el desarrollo y la configuración de las interfaces de las dos aplicaciones. A continuación se comentará para ambas, los aspectos más importantes que se han tenido en cuenta.

5.4.1. Aplicación móvil

Gracias al diseño de la aplicación móvil realizado por el diseñador de **Atenea tech**, se ha conseguido un muy buen aspecto final en las dos pantallas en las que se trabaja con **The Time Bird**.

El diseño seguido ha sido fiel a la estética del logotipo, así como de su simplicidad. Con **The Time Bird** no se quiere dar una interfaz cargada, y así poder realizar las funciones básicas rápidamente, por ello, la aplicación móvil cuenta con una pantalla de inicio de sesión con dos campos de texto siguiendo el estilo de Android, así como un botón de recordatorio de los campos introducidos, y un botón con diseño de la interfaz por defecto de Android. Puede que lo más trabajado sea el Logotipo principal y la pantalla de etiquetas.

Ilustración 124: Logotipo de ***The Time Bird***

En la pantalla de etiquetas, por usabilidad, se utiliza una tabla de dos columnas de botones, ya que se ha creído conveniente utilizar dos y no más botones en una misma, al no poder percibir el texto correspondiente a las etiquetas, si este supera cierto número de caracteres determinado.

En cuanto a los iconos de los menús, se han utilizado iconos de **gRaphael** [21], que juegan un papel perfecto para la interfaz amigable de la aplicación.



Ilustración 125: Botones de menú de la aplicación móvil

5.4.2. Aplicación web

El diseño de una web, normalmente es más tedioso que el diseño de una pantalla para plataformas móviles, y esta no es una excepción. Al igual que con la aplicación móvil, uno de los objetivos del proyecto, es que la aplicación web fuera bonita, por lo que se ha trabajado su interfaz, dividiendo el trabajo en dos partes, el diseño de la web y el diseño de los gráficos de estadísticas.

- **Diseño web:** Drupal, al igual que otros gestores de contenido de Código abierto, posee una cantidad de temas [22] desarrollados por miembros de la comunidad. Uno de ellos, llamado ***journalcrunch*** [23], ya mencionado en el capítulo de diseño, ha sido escogido por su bonita y simple interfaz gráfica, aparte de proporcionar ciertas funcionalidades

adaptables perfectamente al proyecto, como la posibilidad de ofrecer una completa personalización de la interfaz para mostrar información sobre la aplicación al usuario y un pie de página dividido en cuatro columnas, que nos permite dar más información sobre los autores de la aplicación y Drupal.

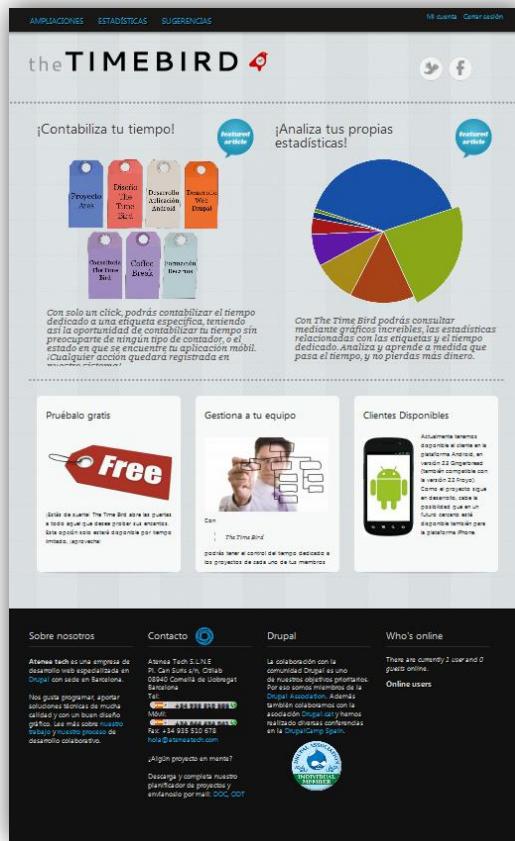


Ilustración 126: Página principal de ***The Time Bird***

La instalación y configuración del tema es relativamente sencilla. Una vez introducido en la carpeta **Themes**, se puede instalar desde el menú de administración de Drupal. Este tema permite publicar en la página principal contenido de tipo **Página** o **Historia** con un estilo propio del tema claramente diferenciable entre el contenido de la mitad superior (para páginas) e inferior (para historias) de la página. Una vez creado el contenido necesario, solo se han tenido que hacer unos pequeños retoques de las imágenes introducidas, en la hoja de estilo del tema TTB. También se han realizado ciertos retoques sobre el tema de **journalcrunch**, siendo algunos de ellos situar un bloque de inicio de sesión en la barra superior, o la eliminación de elementos que no eran necesarios para el producto (como el servicio de RSS). Todos ellos realizados gracias al módulo **css_injector**, que ha permitido realizar cambios sin tener que tocar la hoja de estilos del propio tema.

- **Diseño para los gráficos de estadísticas:** Viendo las diferentes posibilidades para integrar librerías de generación de gráficos en la aplicación web, se encontró una gran cantidad de *APIs* en fases poco maduras, por lo que la usabilidad o la variedad de a la hora de elaborar gráficos en la mayoría de ellas era un problema. También se estudió algunos módulos para Drupal que proporcionan *APIs* para poder trabajar con gráficos en *PHP*, alternativamente a *Flash*³⁸, se encontró una librería interesante, y a pesar de estar en una fase no muy madura, ofrece unos gráficos de muy buena calidad y con una gran personalización, así como unas buenas expectativas de futuro y la posibilidad de usarlas mediante un módulo para Drupal, llamado **raphael** , que cumple con el objetivo estipulado bonita visibilidad del contenido de la aplicación, y que ofrece gráficos como los mostrados a continuación.

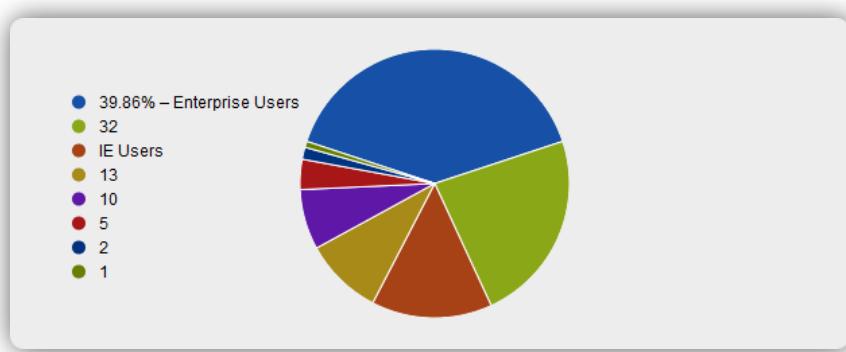


Ilustración 127: Gráfico de ejemplo de Raphael, de tipo *pie*

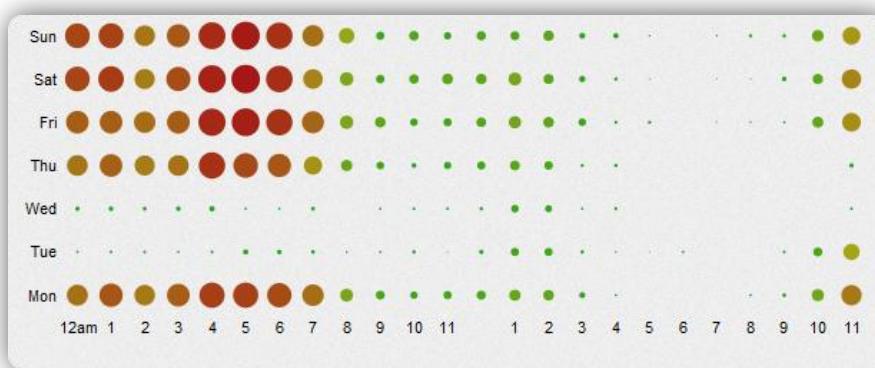


Ilustración 128: Gráfico de ejemplo de Raphael, de tipo *progresivo*

El proceso requerido para el uso de la *API* de Raphael para trabajar con estos gráficos, ha sido bastante simple. Solo hay que bajar e instalar el módulo *Raphael*, y una vez

³⁸ Aplicación de creación y manipulación de gráficos vectoriales con posibilidad de manejo de código mediante el lenguaje *ActionScript* en forma de estudio de animación que trabaja sobre “fotogramas” y está destinado a la producción y entrega de contenido interactivo para las diferentes audiencias alrededor del mundo, sin importar la plataforma

hecho esto, es necesario añadir una línea de código al archivo de información de nuestro propio tema, llamado TTB, donde están situados cada una de las plantillas para cada una de las vistas utilizadas.

```
Dependencies[] = graphael
```

Esta línea permite incluir las librerías *PHP* y *Javascript* necesarias para poder utilizar cualquiera de los gráficos disponibles, y cada uno de estos, como se ha visto en el capítulo de implementación, necesita de una estructura propia de datos para poder ser pintado correctamente.

6. Pruebas

La fase de pruebas, llevada a cabo conjuntamente al desarrollo de todas las funcionalidades del proyecto, es una parte muy importante, ya que, además de hacer constar que las funcionalidades funcionan correctamente, permite descubrir posibles vulnerabilidades de las aplicaciones que no se habían tenido en cuenta, y que pueden ser muy importantes para la seguridad de los datos con los que se trabaja, pudiendo influir en las sensaciones de nuestros clientes sobre la calidad de ***The Time Bird***. Es por ello, por lo que se seguirá un proceso de pruebas para cada una de las aplicaciones, y funcionalidades.

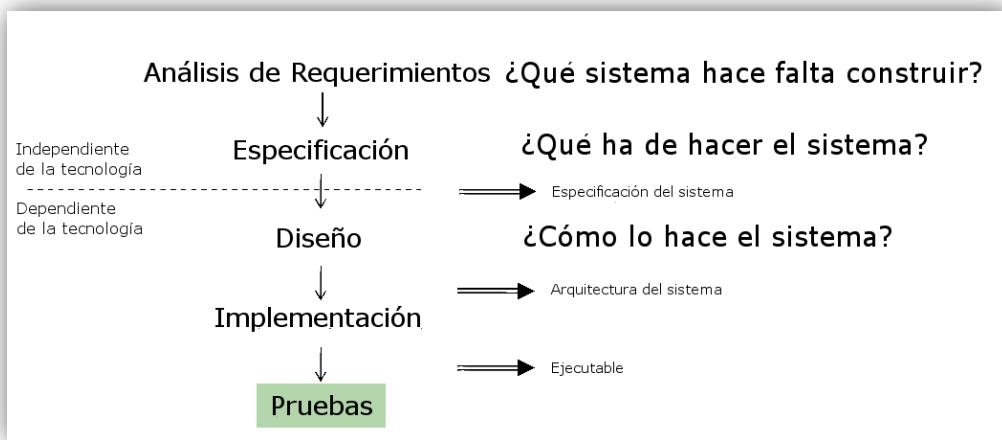


Ilustración 129: Etapas de desarrollo software (Pruebas)

Solo se describen las pruebas realizadas posteriormente a la comprobación de cada funcionalidad, ya que se considera innecesario describir las comprobaciones de las funcionalidades desarrolladas, al suponer que han de funcionar. Para su descripción se utilizarán los símbolos (✓) y (✗) que sirven para anotar si la comprobación ha sido realizada y tratada, o no se ha podido encontrar una solución al respecto.

6.1. Aplicación móvil

A continuación se describen brevemente las comprobaciones realizadas durante la fase de implementación para cada una de las funcionalidades de la aplicación móvil.

6.1.1. Inicio de sesión

Campos vacíos de usuario y/o contraseña	✓
Conectividad de datos (Internet)	✓
Autenticación de usuario	✓
Otras excepciones en el uso de las librerías de comunicación	✗

Tabla 48: Pruebas referentes al inicio de sesión

6.1.2. Cierre de sesión

Conectividad de datos (Internet)	✓
Otras excepciones en el uso de las librerías de comunicación	✗

Tabla 49: Pruebas referentes al cierre de sesión

6.1.3. Creación de etiquetas

Creación de etiquetas ya existentes y activadas	✓
Creación de etiquetas ya existentes y deshabilitadas	✓
Conectividad de datos (Internet)	✓
Otras excepciones en el uso de las librerías de comunicación	✗

Tabla 50: Pruebas referentes a la creación de etiquetas

6.1.4. Edición de etiquetas

Conectividad de datos (Internet)	✓
Otras excepciones en el uso de las librerías de comunicación	✗

Tabla 51: Pruebas referentes a la edición de etiquetas

6.1.5. Activación y desactivación de etiquetas

Conectividad de datos (Internet)	✓
Otras excepciones en el uso de las librerías de comunicación	✗

Tabla 52: Pruebas referentes a la activación y desactivación de etiquetas

6.1.6. Deshabilitación de etiquetas

Conectividad de datos (Internet)	✓
Otras excepciones en el uso de las librerías de comunicación	✗

Tabla 53: Pruebas referentes a la deshabilitación de etiquetas

6.1.7. Compatibilidad

Compatibilidad de la versión 1.6 hasta la versión 2.3 de Android	✓
Compatibilidad con dispositivos de diferentes tamaños de pantalla	✓

Tabla 54: Pruebas referentes a la compatibilidad de dispositivos

6.2. Aplicación web

Aunque el número de elementos a controlar en la aplicación web no sea mayor que en la aplicación móvil, el análisis tendrá que ser más exhaustivo, ya que al estar concentrada la lógica de negocio y gestión de datos en la aplicación web, no podemos permitir que se puedan entrar en zonas restringidas.

A continuación se describen brevemente las comprobaciones realizadas durante la fase de implementación para cada una de las funcionalidades de la aplicación web, comenzando por las funcionalidades más importantes y seguido por comprobaciones de carácter más genérico realizadas en cualquier aplicación web desarrollada con Drupal.

6.2.1. Estadísticas

Filtros expuestos para cada una de las estadísticas	✓
Casos base, sin resultados	✓
Casos extremos, en que hay miles de datos y usuarios	✗

Tabla 55: Pruebas referentes a la elaboración de las distintas estadísticas

6.2.2. Sugerencias

Envío del formulario a la dirección correcta	✓
Envío de correo de confirmación por parte del sistema	✓

Tabla 56: Pruebas referentes a la elaboración de sugerencias

6.2.3. Interfaz

Enlaces del sitio actualizados (incluidos menús)	✓
--	---

Tabla 57: Pruebas referentes a comprobaciones en la interfaz de usuario

6.2.4. Seguridad y permisos de usuario

Vulnerabilidades conocidas de Drupal	✓
Acceso a zonas restringidas para usuarios no registrados (navegación)	✓
Acceso y modificación al contenido permitido	✓

Tabla 58: Pruebas referentes a comprobaciones de seguridad y permisos de usuario

6.2.5. Compatibilidad

Compatibilidad con los navegadores IE 6.0+, Google Chrome 11.0, Firefox 3.0+	✓
Acceso y modificación al contenido permitido	✓

Tabla 59: Pruebas referentes a compatibilidad de la aplicación web

6.2.6. Rendimiento

Tiempo de respuesta de la aplicación web	✗
Carga máxima de usuarios	✗

Tabla 60: Pruebas referentes a rendimiento de la aplicación

7. Conclusiones, ampliaciones y mejoras

Este capítulo, pretende revisar y opinar sobre todo el trabajo realizado.

Primero se realizará una revisión de los objetivos iniciales marcados, seguida de las desviaciones que se han producido respecto a la planificación inicial realizada. También se realizará un pequeño análisis con las conclusiones finales y valoración personal del proyecto y finalmente se describirán algunas de las muchas ampliaciones que se pueden realizar a **The Time Bird**.

7.1. Revisión de los objetivos

Es importante comprobar si se han cumplido con los objetivos iniciales del proyecto, una vez acabado el desarrollo, por lo que se volverán a nombrar cada uno de ellos junto a una breve explicación sobre el resultado final obtenido.

● De la **aplicación móvil**, con Sistema Operativo Android 2.3:

- Inicio de sesión para cada uno de nuestros clientes, lo que permitirá gestión de etiquetas personalizada.

Se ha desarrollado una aplicación con restricción de acceso a usuarios registrados en la aplicación web, lo que permite a cada usuario obtener los datos almacenados en la misma

- Pantalla de gestión de diferentes conceptos como proyectos, procesos y clientes, materializados en etiquetas.

Se ha creado una pantalla que permite al usuario manejar las etiquetas creadas muy fácilmente y de forma intuitiva.

- Opciones secundarias y personalización de la aplicación.

Se han desarrollado algunas opciones secundarias meramente informativas de cada al usuario, y se ha personalizado la aplicación al estilo fiel a The Time Bird.

- Desarrollo de librerías en Java necesarias para proveer a la aplicación móvil de los métodos necesarios para la transacción de datos entre las aplicaciones.

La aplicación móvil permite enviar y recibir datos de la aplicación web, y permite la gestión de errores durante el proceso.

- Compatibilidad con todas las versiones posibles del SO (incluyendo como mínimo las más utilizadas; versión 2.2 (*Froyo*) y versión 2.3 (*Gingerbread*)).

Se ha comprobado que la aplicación desarrollada sea compatible con las últimas versiones (y más utilizadas) de Android.

● De la **aplicación web**, con Drupal:

- Creación del contenido necesario para almacenar los datos de los usuarios.
La aplicación web permite trabajar con todo el contenido que permite a los usuarios contabilizar el tiempo invertido en ciertas etiquetas.
- Instalación y configuración de los módulos necesarios que harán posible la gestión de ciertos tipos de contenido.
Se han instalado y configurado todos los módulos necesarios para que *The Time Bird* realice las funcionalidades indicadas en el capítulo de especificación. También se han deshabilitado los módulos que no eran necesario y que Drupal traía por defecto.
- Configuración del módulo Services que hará posible el desarrollo de librerías para el envío y recepción de datos para permitir la sincronización de datos con el móvil.
Dicho módulo ha sido configurado correctamente añadiendo todas las opciones disponibles de seguridad para la aplicación web.
- Desarrollo de librerías en PHP necesarias para proveer a la aplicación web de los métodos necesarios para la transacción de datos entre aplicaciones.
La aplicación web permite enviar y recibir datos de la aplicación móvil.
- Automatización en *background* del proceso de análisis de las tareas realizadas hasta el momento (proceso a realizar cada 5 minutos).
El proceso de análisis es llevado a cabo de forma correcta cada vez que es ejecutado el Cron de Drupal. No se pueden contemplar las estadísticas de las etiquetas utilizadas hasta no haber realizado este proceso, realizado cada 5 minutos.
- Gestión de los diferentes tipos de estadísticas de los datos enviados por el usuario.
La aplicación permite visualizar una serie de estadísticas relacionadas con los usuarios, etiquetas y tiempo dedicado de cada una de ellas.
- Aplicaciones **simples, bonitas, usables y útiles**.
Aunque siempre se podría mejorar la usabilidad, la sencillez e interfaces de las aplicaciones, en ambas se ha realizado un trabajo satisfactorio en estos aspectos.
- Análisis del potencial del Sistema Operativo Android y el CMS Drupal.
Se ha realizado un estudio analizando las posibilidades de cada sistema, y se ha justificado su uso.
- Ambas aplicaciones tendrán asociada una licencia libre **GPL**.
Las dos aplicaciones, serán distribuidas con una licencia GPL.
- Realizar toda la documentación necesaria para que nuestros clientes puedan entender el funcionamiento de la aplicación perfectamente.
Se ha elaborado toda la documentación necesaria como guía para los usuarios en el uso de cada una de las aplicaciones.

En resumen, y como se puede observar, se han cumplido con todos los objetivos propuestos.

7.2. Desviaciones de la planificación inicial y presupuesto

En el análisis inicial de proyecto, se realizó una planificación orientativa aunque ya muy aproximada del tiempo que costaría realizar todo el proyecto, habiendo dividido las tareas en base a los objetivos y el alcance de proyecto. Como en algunas materias no se tenía mucha experiencia, se han encontrado algunos problemas difíciles de solucionar, que han llevado a una pequeña desviación de la planificación inicial, con su correspondiente coste económico añadido.

En este apartado se presentarán los principales problemas que hicieron que el proyecto se retrasara, seguido de la planificación y presupuesto final realizados.

7.2.1. Problemas encontrados

Los principales problemas encontrados fueron en su mayoría, sobre el diseño de ciertas funcionalidades, como la elaboración de estadísticas, u otros problemas relacionados con la implementación de las librerías necesarias para la comunicación de datos.

A continuación se presenta una lista con las principales dificultades encontradas:

- **Configuración y testeo del módulo Services:** Aunque se trate de un módulo bastante maduro, hubo bastantes problemas para adaptarlo con los métodos desarrollados con Android que permitían el inicio de sesión en las aplicaciones. El problema fue que al realizar las comunicaciones, Drupal daba errores de acceso sin ningún tipo de información complementaria, por lo que el error podía estar tanto en los datos enviados por la aplicación móvil, como por una mala configuración en la aplicación web. Al final se descubrió que el problema estaba en la aplicación web, al tener que dar permisos a la *API KEY* creada, para cada método que se quería utilizar.
- **Liberías de comunicación de datos:** En algunos de los métodos que ofrece el módulo *Services*, es necesario recibir una estructura de tipo objeto, por lo que se creó una clase en Java, llamada ***DrupalNode*** que representaba tal estructura, con los respectivos atributos obligatorios representando un tipo de contenido en la aplicación web. El problema surgió cuando se necesitaba enviar información a Drupal, con un tipo de estructura peculiar, y que era imposible realizar en Java, por lo que se desechó la idea de utilizar esta clase, y se decidió utilizar métodos propios que permitían enviar datos simples como cadenas de caracteres a la aplicación web, y que de esta

forma, fuera la encargada de realizar todo el proceso para almacenar las estructuras en la base de datos.

Cabe decir que esta opción de diseño nos daría una flexibilidad importante a la hora de ampliar el número de clientes a otras plataformas móviles, ya que solo tendríamos que desarrollar librerías que permitan el envío de datos tan simple como caracteres, ocupando la aplicación web de toda la lógica de negocio.

- **Funcionalidades secundarias:** Una de las funcionalidades secundarias que se quiso realizar, a modo de mejora del diseño de la interfaz de la aplicación móvil, fue el desarrollo de una imagen propia para la barra de progreso (más conocido como *Spinner*) utilizada a la hora de cargar cierta información. Su desarrollo llevo más tiempo de lo esperado, ya que se encontraban errores que fueron difíciles de solucionar, al no estar muy familiarizados con la plataforma Android.
- **Librerías para gráficos:** En una primera elaboración del diseño que tendría ***The Time Bird***, no se había contemplado el formato que tenían que tener los gráficos en la aplicación web, si tenían que ser estáticos, dinámicos, desarrollados en *Flash*, o *Javascript*, etc. Se dedicó una cantidad de horas considerable a la búsqueda de librerías y alternativas para Drupal, y al final se encontró un módulo llamado ***Open Flash Chart API*** [24], muy buena opción para mostrar gráficos dinámicos. Una vez realizadas algunas estadísticas con este módulo que te proporcionaba ciertas librerías en *PHP*, preferimos buscar alternativas que permitieran poder ver los gráficos en todos los navegadores posibles, por lo que el lenguaje *Flash* no era una opción. Finalmente se encontró una buena alternativa, pero el tiempo estimado para el desarrollo de los gráficos para las diferentes estadísticas, se tuvo que ampliar.

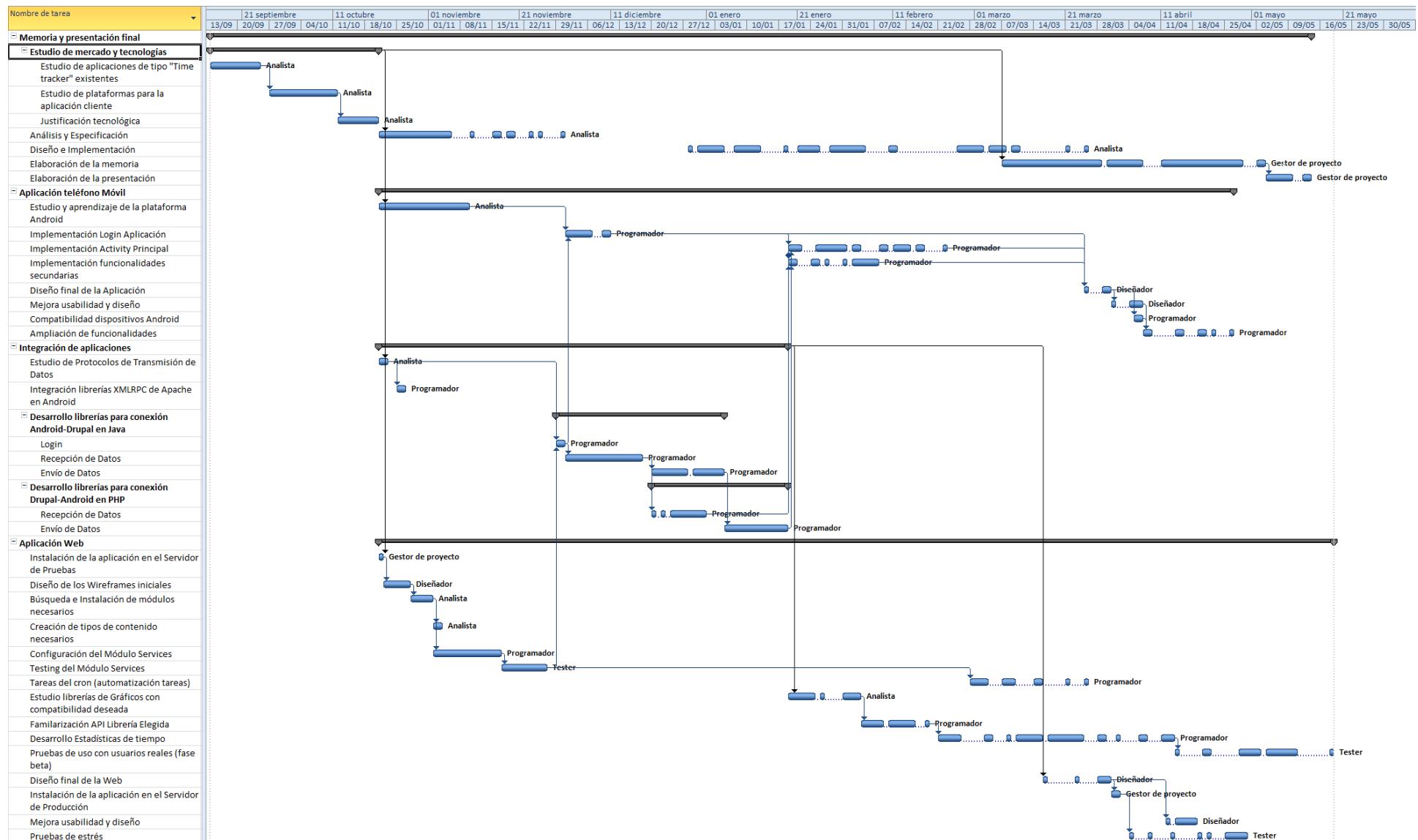
Todos estos problemas causaron un retraso más o menos importante en la planificación realizada inicialmente, por lo que se tuvo que ajustar en la medida que fue necesaria.

7.2.2. Planificación final

La planificación final, pretende mostrar de forma muy aproximada, la carga de trabajo durante todo el tiempo que se ha estado desarrollando el proyecto. Como se ha visto en el anterior apartado, algunos problemas han hecho que se tuvieran que hacer ajustes en la planificación inicial, aunque debido a las dependencias entre actividades, no se ha alargado de forma considerable el día de final de proyecto y se ha aumentado la carga de trabajo en la medida necesaria.

A continuación se muestra el diagrama de la planificación final.

Conclusiones, ampliaciones y mejoras



7.2.3. Presupuesto final

Cabe destacar que debido a la desviación producida en la planificación inicial, ha habido un aumento en los costes de los recursos humanos, al haber aumentado la carga de trabajo.

Entonces, para realizar el cómputo total, solo hace falta recalcular los costes de los recursos humanos, y sumarle los costes de los recursos físicos, que son exactamente iguales que en análisis económico inicial.

Las horas de trabajo finales, son las calculadas a partir de la dedicación de recursos humanos en la última planificación de proyecto finalizada. La siguiente tabla refleja las horas totales dedicadas por cada recurso, así como el presupuesto o coste final de proyecto.

Rol	Horas de trabajo	€/hora	Coste total en €
Gestor de proyecto ³⁹	149	40	5960
Analista	330	30	9900
Programador	454	20	9080
Diseñador	48	20	960
Tester	79	20	1580
Total recursos			27480

Tabla 61: Costes totales finales de recursos humanos

Entonces, el coste total de proyecto serían 27.480€ de recursos humanos, más 1.210€ de recursos físicos, hacen un total de **28.690€**.

A continuación se puede ver a modo de detalle, el tiempo final aproximado dedicado por cada uno de los recursos disponibles.

³⁹ Se ha incluido el tiempo utilizado en la elaboración de la documentación del proyecto.

Conclusiones, ampliaciones y mejoras

Nombre del recurso	Trabajo	Detalles		'10		27 sep '10		18 oct '10		08 nov '10		29 nov '10		20 dic '10		10 ene '11		31 ene '11		21 feb '11		14 mar '11		04 abr '11		25 abr '11		16 may '11								
		12	22	02	12	22	01	11	21	01	11	21	31	10	20	30	09	19	01	11	21	31	10	20	30	10	20									
- Gestor de proyecto	149 horas	Trabajo					2h																		12h	24h	22h	17h	18h	26h	20h	8h				
Elaboración de la memoria	123 horas	Trabajo																								12h	24h	22h	13h	18h	26h	8h				
Elaboración de la presentación	20 horas	Trabajo																													12h	8h				
Instalación de la aplicación en el Servidor de Pruebas	2 horas	Trabajo						2h																												
Instalación de la aplicación en el Servidor de Producción	4 horas	Trabajo																													4h					
- Analista	330 horas	Trabajo	36h	48h	36h	48h	26h	32h	6h	4h							4h	5h	7h	17h	28h	6h	8h	14h	0h	5h										
Estudio de aplicaciones de tipo "Time tracker" existentes	54 horas	Trabajo	36h	18h																																
Estudio de plataformas para la aplicación cliente	66 horas	Trabajo		30h	36h																															
Justificación tecnológica	42 horas	Trabajo				42h																														
Análisis y Especificación	29 horas	Trabajo					2h	10h	7h	6h	4h						4h	5h	6h	5h	20h	6h	8h	14h	0h	5h										
Diseño e Implementación	73 horas	Trabajo																																		
Estudio y aprendizaje de la plataforma Android	31 horas	Trabajo						2h	10h	19h																										
Estudio de Protocolos de Transmisión de Datos	4 horas	Trabajo						2h	2h																											
Búsqueda e Instalación de módulos necesarios	6 horas	Trabajo							4h	2h																										
Creación de tipos de contenido necesarios	4 horas	Trabajo								4h																										
Estudio librerías de Gráficos con compatibilidad deseada	21 horas	Trabajo																			1h	12h	8h													
- Programador	454 horas	Trabajo						4h	16h	16h	9h	38h	24h	38h	27h	29h	24h	32h	34h	28h	32h	20h	24h	28h	19h	12h										
Implementación Login Aplicación	9 horas	Trabajo									9h																									
Implementación Activity Principal	52 horas	Trabajo																			3h	17h	12h	18h	2h											
Implementación funcionalidades secundarias	22 horas	Trabajo																			2h	7h	13h													
Compatibilidad dispositivos Android	8 horas	Trabajo																												8h						
Ampliación de funcionalidades	29 horas	Trabajo																												10h	7h	12h				
Integración librerías XMLRPC de Apache en Android	4 horas	Trabajo					4h																													
Login	9 horas	Trabajo									9h																									
Recepción de Datos	49 horas	Trabajo										29h	20h																							
Envío de Datos	40 horas	Trabajo											2h	24h	14h	1h																				
Recepción de Datos	17 horas	Trabajo											2h	14h	1h	.h																				
Envío de Datos	36 horas	Trabajo												12h	24	24	24h																			
Configuración del Módulo Services	32 horas	Trabajo							16h	16h																										
Tareas del cron (automatización tareas)	30 horas	Trabajo																												4h	18h	4h	4h			
Familiarización API Librería Elegida	23 horas	Trabajo																			7h	16h														
Desarrollo Estadísticas de tiempo	94 horas	Trabajo																					22h	14h	16h	20h	10h	12h								
- Diseñador	48 horas	Trabajo						6h																								2h	17h	9h	14h	
Diseño final de la Aplicación	6 horas	Trabajo																																		
Mejora usabilidad y diseño	9 horas	Trabajo																																		
Diseño de los Wireframes iniciales	6 horas	Trabajo							6h																											
Diseño final de la Web	13 horas	Trabajo																																		
Mejora usabilidad y diseño	14 horas	Trabajo																																		
- Tester	79 horas	Trabajo										10h	19h																							
Testing del Módulo Services	29 horas	Trabajo										10h	19h																							
Pruebas de uso con usuarios reales (fase beta)	28 horas	Trabajo																																		
Pruebas de estrés	22 horas	Trabajo																																		

7.3. Valoración

La experiencia vivida durante toda la elaboración del proyecto, ha sido muy satisfactoria. He tenido la oportunidad de realizar un proyecto realmente interesante para pequeñas empresas que disponen de pequeños grupos de trabajo, y para autónomos, que normalmente no suelen tener una estimación real de las horas que le dedican a su trabajo diario. **The Time Bird** ya ha recibido sus primeras aceptaciones, y ya está en fase de pruebas, con más de cuarenta usuarios registrados en la aplicación web, lo cual demuestra que la gente que va enterándose de la existencia de la aplicación, muestre curiosidad por ella y la pruebe.

Además, se tiene una buena expectativa de futuro puesta en **The Time Bird**, ya que las ampliaciones que va a percibir son muy interesantes para los usuarios, y sin perder uno de los objetivos principales, que es la sencillez de uso, se van a poder realizar ampliaciones de las funcionalidades y mejoras en la interfaz, que hagan cada vez más interesante su uso.

En cuanto a experiencia personal, puedo decir que he aprendido muchísimo tanto de Drupal como Android en este proyecto, dos tecnologías muy utilizadas en la actualidad, y las dos en auge, por lo que no me cabe la menor duda que serán muy útiles en mi vida laboral, y me permitirá mejorar mi habilidad en el desarrollo de aplicaciones. También cabe decir que todo el trabajo realizado me ha permitido recordar mucha teoría vista a lo largo de la carrera, aplicada a casos reales en el mundo laboral, lo cual, es una muy buena experiencia porque te da una visión alternativa a muchos de los conceptos aprendidos.

El mero hecho de haber podido cumplir con todos los objetivos, y de haber podido trabajar con personas geniales, la vida laboral durante todos estos meses se me ha hecho muy amena, y ha resultado ser un trabajo bonito y grato, y estoy feliz por ello.

7.4. Ampliaciones y mejoras

A continuación se procede a describir las ampliaciones y mejoras que se podrían realizar sobre las dos aplicaciones, haciendo así a **The Time Bird** un producto mucho más atractivo y funcional.

Todos los puntos descritos no se han podido llevar a cabo por la limitación que ha supuesto el alcance el proyecto. Además algunas de las ampliaciones nombradas son algo costosas, por lo que se tendrían que llevar a cabo en futuras versiones de la aplicación.

7.4.1. Ampliaciones

Aplicación móvil

- Compartir etiquetas entre usuarios.
- Creación de grupos para etiquetas (clientes, procesos, proyectos...) diferenciados por colores.
- Geolocalización de las tareas realizadas.
- Permitir la transacción de datos en nuevos clientes, al menos en plataforma web e iPhone.
- Avisos temporales en la barra de estado, sobre el tiempo que lleva el usuario procesando ciertas etiquetas activas en el momento.
- Guardar temporalmente los datos en los clientes móviles hasta que haya Internet y se puedan enviar los datos.
- Entrada manual de tiempo.

Aplicación web

- Soporte a Geolocalización de tareas y etiquetas, mediante el uso de **Open Layer⁴⁰**.
- Asociación estadística a etiquetas y otros tipos de contenido.
- Apartado de búsqueda de etiquetas relacionadas.
- Permitir peticiones para obtener los datos de otros usuarios.

7.4.2. Mejoras

Aplicación móvil

- Mejoras genéricas de usabilidad.
- Mejoras en la interfaz de la aplicación.
- Si un usuario lleva mucho tiempo con una tarea activa (15h) enviarle un correo electrónico de recordatorio. Si lleva más de 24h borrar la tarea.

⁴⁰ **OpenLayer** es un recurso sencillo para agregar un mapa interactivo en nuestro sitio web. Existe un módulo en Drupal, llamado **openlayers** [25] para permitir a los desarrolladores, trabajar con mapas.

Aplicación web

- Mejoras en la rapidez del proceso de análisis y muestra de gráficos.
- Creación de una interfaz completamente nueva mediante la ayuda de un diseñador gráfico.
- Elaboración de más tipos de gráficos estadísticos.

8. Glosario

El glosario, pretende recoger todas las siglas utilizadas a lo largo del documento de forma que siempre que se desee, se puedan consultar sus definiciones.

Las siglas están ordenadas de forma alfabética.

API	<i>Application Programming Interface.</i> La Interfaz de Programación de Aplicaciones, es un grupo de rutinas que definen cómo invocar desde un programa, un servicio que éstos prestan. En otras palabras, una <i>API</i> representa una interfaz de comunicación entre componentes <i>software</i>
API KEY	Es un identificador que sirve como autenticador cada vez que se utilizan los servicios de un sitio determinado.
CAPTCHA	<i>Completely Automated Public Turing test to tell Computers and Humans Apart.</i> Se trata de una prueba desafío-respuesta utilizada en computación para determinar cuando el usuario es o no humano.
CMS	<i>Content Management System.</i> Un Sistema gestor de contenidos, se trata de una aplicación para crear, editar, gestionar y publicar contenido de forma consistente y organizada.
CRM	<i>Customer Relationship Management.</i> Es un modelo de gestión de una organización orientada a la gestión de la relación con sus clientes.
CSS	<i>Cascading Style Sheets.</i> Las hojas de estilo en cascada, es un lenguaje usado para definir la presentación de un documento estructurado escrito en <i>HTML</i> o <i>XML</i> . El <i>World Wide Web Consortium</i> es el encargado de formular la especificación de las hojas de estilo que servirán de estándar para los navegadores.
GNU GPL	<i>GNU General Public License.</i> La Licencia Pública General de GNU, es una licencia creada por la <i>Free Software Foundation</i> en 1989 (la primera versión), y está orientada principalmente a proteger la libre distribución, modificación y uso de <i>software</i> . Su propósito es declarar que el <i>software</i> cubierto por esta licencia es libre y protegerlo de intentos de apropiación que restrinjan esas libertades a los usuarios.

GPS	<i>Global Positioning System.</i> Un sistema de posicionamiento global es un sistema que permite determinar en todo el mundo la posición de un objeto con una precisión de pocos metros.
HTML	<i>HyperText Markup Language.</i> Es el lenguaje predominante para el desarrollo de páginas web. <i>HTML</i> se escribe en forma de etiquetas rodeadas por corchetes angulares, que sirven para definir cierto contenido.
IDE	<i>Integrated Development Environment.</i> Es un entorno de programación que ha sido empaquetado como un programa de aplicación, es decir, consiste en un editor de código, un compilador, un depurador y un constructor de interfaz gráfica.
MVC	El Modelo Vista Controlador es un patrón de arquitectura del <i>software</i> que separa los datos de una aplicación, la interfaz de usuario y la lógica de control en tres componentes distintos.
JSON	<i>JavaScript Object Notation.</i> La Notación de Objetos en <i>JavaScript</i> es un formato ligero para el intercambio de datos.
PAC	<i>Presentation Abstraction Control.</i> Es un patrón de arquitectura de <i>software</i> para sistemas interactivos basada en una jerarquía de agentes cooperantes similar a Modelo Vista Controlador.
PC	<i>Personal Computer.</i> Un ordenador personal, es una computadora diseñada para ser usada por una persona a la vez
PHP	Lenguaje de programación diseñado originalmente para la creación de páginas web dinámicas. Generalmente se usa en el lado del servidor.
REST	<i>Representational State Transfer.</i> Es una técnica de arquitectura de <i>software</i> para sistemas hipermedia distribuidos como la <i>World Wide Web</i> . Usa HTTP como protocolo de transmisión de mensajes.
RSS	<i>Really Simple Syndication.</i> Formato <i>XML</i> para compartir contenido en la web. Se utiliza para difundir información actualizada frecuentemente a usuarios que se han suscrito a la fuente de contenidos.
SDK	<i>Software Development Kit.</i> Un kit de desarrollo de <i>software</i> es un conjunto de herramientas de desarrollo que permite a un programador

	crear aplicaciones para un sistema concreto.
SGBD	Los Sistemas de Gestión de Bases de Datos son un tipo de <i>software</i> muy específico, dedicado a servir de interfaz entre la base de datos, el usuario y las aplicaciones que la utilizan.
SO	Un Sistema Operativo es un programa o conjunto de programas que efectúan la gestión de los procesos básicos de un sistema informático, y permite la normal ejecución del resto de las operaciones.
SOAP	<i>Simple Object Access Protocol.</i> Es un protocolo estándar que define como dos objetos en diferentes procesos pueden comunicarse por medio de intercambio de datos XML. Este protocolo deriva del protocolo <i>XML-RPC</i> .
SQL	<i>Structured Query Language.</i> Es un lenguaje utilizado para el acceso a bases de datos relacionales que permite especificar diversos tipo de operaciones en éstas.
SSL	<i>Secure Sockets Layer.</i> El protocolo de capa de conexión segura proporciona autenticación y privacidad de la información entre extremos sobre Internet, mediante el uso de criptografía. Normalmente sólo el servidor es autenticado, mientras que el cliente se mantiene sin autenticar.
TTB	Módulo desarrollado en <i>PHP</i> para poder llevar a cabo la mayor parte de funcionalidades de The Time Bird en la aplicación web.
URL	<i>Uniform Resource Locator.</i> El localizador uniforme de recursos es una secuencia de caracteres, de acuerdo a un formato modélico y estándar, que se usa para nombrar recursos en Internet para su localización o identificación.
XML	<i>eXtensible Markup Language.</i> El lenguaje de marcas extensible es un metalenguaje extensible de etiquetas, desarrollado por el <i>World Wide Web Consortium</i> y permite definir una gramática de lenguajes específicos. Tiene un papel muy importante en la actualidad ya que permite la compatibilidad entre sistemas para compartir la información de forma segura, fiable y fácil.
XTHML	<i>eXtensible Hypertext Markup Language.</i> El lenguaje extensible de

marcado de hipertexto es el lenguaje pensado para sustituir *HTML* como estándar para las páginas web. Su objetivo es avanzar en el proyecto del *World Wide Web Consortium*^k de lograr una web semántica, donde la información y la forma de presentarla estén claramente separadas.

- XMLRPC** Primer mecanismo que surgió para invocar procedimientos remotos vía XML. Ofrece de forma sencilla, invocar operaciones en sistema heterogéneos a través de una estructura simple y usa *HTTP* como protocolo de transmisión de mensajes.

9. Bibliografía

En este capítulo, se mencionan las fuentes y referencias que han sido utilizadas para la elaboración del proyecto. Encontraremos la bibliografía dividida en dos partes. La primera, la bibliografía referenciada en la memoria, ordenada por orden de aparición. La segunda, hace referencia a otro material utilizado y no referenciado.

Bibliografía referenciada

- [1] DOOLPHY. (2011) *Project Management & Time Tracking, A Perfect Marriage.*
<http://blog.doolphy.com/2011/02/17/project-management-time-tracking-a-perfect-marriage/>
- [2] Traducción al castellano de la licencia *GPL* de GNU.
<http://www.viti.es/gnu/licenses/gpl.html>
- [3] Metodologías de desarrollo *software* genéricas.
http://es.wikipedia.org/wiki/Metodolog%C3%ADA_de_desarrollo_de_software#Enfoques_de_desarrollo_de_software
- [4] Desarrollo de *software* con metodologías ágiles.
http://es.wikipedia.org/wiki/Desarrollo_%C3%A1gil_de_software
- [5] XATAKA. (2011) *Nokia adopta Windows Phone 7. Todas las claves de la alianza entre Microsoft y Nokia.* <http://www.xataka.com/moviles/nokia-adopt-a-windows-phone-7-todas-las-claves-de-la-alianza-entre-microsoft-y-nokia>
- [6] Comparativa de sistemas operativos móviles.
<http://diarioandroid.com/2009/05/06/comparativa-entre-sistemas-operativos-moviles>
- [7] Página oficial de *Nielsen*. <http://es.nielsen.com/site/index.shtml>
- [8] Página oficial de *Tap Tap*. <http://www.taptapnetworks.com/>
- [9] Página oficial de *Lookout*. <https://www.mylookout.com/>
- [10] Estudio AppGenome sobre las tiendas de aplicaciones de Google y Apple.
<https://www.mylookout.com/appgenome>
- [11] Estudio realizado por Gartner sobre la venta de *smartphones* en Estados Unidos en 2010. <http://www.gartner.com/it/page.jsp?id=1543014>
- [12] Página de información sobre el equipo de seguridad de Drupal.
<http://drupal.org/security-team>
- [13] Sitio web de comparativas entre gestores de contenido. <http://cmsmatrix.org>
- [14] Blog de tendencias de CMS de *buildwith*. <http://trends.builtwith.com/cms>
- [15] Buenas prácticas de código con Drupal. <http://drupal.org/node/318>

- [16] Información sobre hooks en Drupal. <http://api.drupal.org/api/drupal/6>
- [17] Página de información para desarrolladores, sobre las plataformas más utilizadas en Android. <http://developer.android.com/resources/dashboard/platform-versions.html>
- [18] Página de descarga de la implementación XMLRPC de Apache.
<http://ws.apache.org/xmlrpc>
- [19] Patrones de diseño utilizados en Drupal. <http://drupal.org/node/547518>
- [20] Patrones de diseño para la navegación en aplicaciones Android.
http://www.androidpatterns.com/uap_category/navigation
- [21] Capa de abstracción de la base de datos en Drupal.
<http://api.drupal.org/api/drupal/includes--database.inc/group/database/6>
- [22] Iconos de *Raphael*. <http://raphaeljs.com/icons>
- [23] Página para la descarga de Temas de Drupal. <http://drupal.org/project/Themes>
- [24] Módulo *journalcrunch* de Drupal. <http://drupal.org/project/journalcrunch>
- [25] Módulo *Open Flash Chart* para Drupal.
http://drupal.org/project/open_flash_chart_api
- [26] Módulo *OpenLayers* de Drupal. <http://drupal.org/project/openlayers>

Bibliografía no referenciada

- [1] DOLORS COSTAL, XAVIER FRANCH, M. RIBERA SANCHO, ERNEST TENIENTE. (Edicions UPC 2000) *Enginyeria del Software: Especificació*.
- [2] JOHN K. VANDYK, MATT WESTGATE. (USA Apress 2007) *Pro Drupal Development*.
- [3] *Android Development community tutorials*. <http://www.anddev.org>
- [4] *Wikipedia*. <http://es.wikipedia.org>

10. Anexos

10.1. Lista de elementos que aparecen en la memoria

Índice de Ilustraciones

Definición gráfica para los métodos de desarrollo ágiles.....	12
Descripción gráfica del modelo en cascada.....	14
Visualización de ejemplo de <i>tick</i>	15
Visualización de ejemplo de <i>timr</i>	17
Visualización de ejemplo de <i>Task Coach</i>	18
Visualización de ejemplo de <i>Officetime</i>	19
Visualización de ejemplo de <i>FreshBooks</i>	21
Visualización de ejemplo (1) de <i>toggI</i>	22
Visualización de ejemplo (2) de <i>toggI</i>	23
Visualización de ejemplo de <i>Klok</i>	24
Visualización de ejemplo de <i>Cashboard</i>	26
Visualización de ejemplo de <i>Harvest</i>	27
Visualización de ejemplo de <i>Paymo</i>	29
Visualización de ejemplo (1) de <i>Freckle</i>	30
Visualización de ejemplo (2) de <i>Freckle</i>	30
Visualización de ejemplo de <i>The Time Bird</i>	32
Etapas de desarrollo <i>software</i> (Análisis de Requerimientos)	33
Logotipos de los diferentes sistemas operativos a analizar	41
Audiencia de las diferentes plataformas móviles en el cuarto cuatrimestre de 2010 en España.....	44
Iconos de las tiendas de aplicaciones <i>App Store</i> y <i>Android Market</i> respectivamente	45
Aplicaciones disponibles en <i>Android Market</i> y <i>App Store</i> en Febrero de 2011.....	45
Porcentaje de ventas de <i>smartphones</i> por sistema operativo, en todo el mundo	46
Figuras Android mostrándose vencedoras ante el padre de Apple, Steve Jobs.....	47
Logotipos de <i>Drupal</i> , <i>Joomla</i> y <i>Wordpress</i> , respectivamente.....	48

Distribución de CMS calculada sobre el top 1.000.000 de sitios web.....	52
Distribución de CMS calculada sobre el top 10.000 sitios web.....	52
Tendencias de uso de páginas consultadas con Drupal.....	53
Tendencias de uso de páginas consultadas con Wordpress	53
Tendencias de uso de páginas consultadas con Joomla	53
Drupal es el gestor de contenido elegido	54
Etapas de desarrollo <i>software</i> (Especificación).....	55
Jerarquización de actores de la aplicación móvil	56
Diagrama de casos de uso de la aplicación móvil	56
Jerarquización de actores de la aplicación web	60
Diagrama de casos de uso de la aplicación web	60
Modelo conceptual aplicación móvil	66
Modelo conceptual de la aplicación web	67
Parte del modelo conceptual de Drupal 6	68
Diagrama de secuencia 1. Acerca de la aplicación	69
Diagrama de secuencia 2. Crear etiqueta.....	69
Diagrama de secuencia 3. Editar etiqueta	70
Diagrama de secuencia 4. Deshabilitar etiqueta.....	70
Diagrama de secuencia 5. Activar etiqueta.....	71
Diagrama de secuencia 6. Desactivar etiqueta.....	71
Diagrama de secuencia 7. Consultar ampliaciones	72
Diagrama de secuencia 8. Realizar sugerencia.....	72
Diagrama de secuencia 9. Registro de usuario.....	73
Diagrama de secuencia 10. Solicitud de contraseña	73
Diagrama de secuencia 11. Ver mi cuenta.....	74
Diagrama de secuencia 12. Editar mi cuenta	74
Diagrama de secuencia 13. Editar perfil.....	75
Diagrama de secuencia 14. Ver estadísticas (Etiquetas).....	75
Diagrama de secuencia 15. Ver estadísticas (Etiquetas/Periodos).....	76

Diagrama de secuencia 16. Ver estadísticas (Usuarios/Periodos).....	76
Diagrama de secuencia 17. Ver estadísticas (Etiquetas/Usuarios)	77
Diagrama de secuencia 18. Ver estadísticas (Etiquetas/Usuarios/Periodos)	77
Etapas de desarrollo software (Diseño).....	79
Arquitectura básica de los sistemas con varios clientes conectados simultáneamente.	81
Pila tecnológica de Drupal.....	82
Curva de aprendizaje de Drupal.....	83
Algunas de las funcionalidades del núcleo de Drupal	84
Activación de módulos que ofrecen nuevas funcionalidades.....	85
Ejemplo de ganchos que son llamados por otros ganchos	85
Logotipo del sistema operativo Android	87
Componentes principales de Android.....	89
Ciclo de vida de una Actividad en Android.....	91
Datos de dispositivos por nivel de API a 1 de Abril de 2011.....	92
Ejemplo de código escrito en XML.....	94
Ejemplo de código escrito en JSON	94
Funcionamiento de la comunicación mediante XML	95
Estructura por capas, en Drupal	96
Esquema de contenido de ejemplo en Drupal como patrón de diseño PAC.....	97
Esquema horizontal que podemos encontrar en Drupal.....	98
Mapa de navegación de la aplicación móvil	100
Mapa de navegación por usuarios de la aplicación móvil	101
Conjunto de Actividades en la aplicación móvil	101
Mapa de navegación de la aplicación móvil	102
Mapa de navegación por usuarios de la aplicación web	103
Diseño por páginas en la aplicación web	104
Diagrama de secuencia ampliado, de Inicio de sesión.....	106
Diagrama de secuencia ampliado, de inicio de Activar etiqueta	106
Diagrama de secuencia ampliado, de inicio de Desactivar etiqueta	107

Diagrama de secuencia ampliado, de Crear etiqueta	108
Diagrama de secuencia ampliado, de Editar etiqueta.....	108
Diagrama de secuencia ampliado, del proceso de transformación en Tareas procesadas.	109
Etiqueta	111
Tarea no procesada.....	112
Periodo.....	112
Tarea procesada	113
Perfil	114
Sugerencia	114
Página.....	115
Historia	115
Etapas de desarrollo software (Implementación)	117
Página de permisos de Drupal	119
<i>API KEY</i> para <i>The Time Bird</i>	120
Proceso seguido a la hora de hacer una llamada a la aplicación web.....	122
Parámetros necesarios para iniciar la sesión de usuario	124
Lista de etiquetas en la aplicación móvil	125
Vista <i>tag_list</i>	125
Vista <i>last_unprocessed</i>	126
Proceso de creación de una etiqueta en la aplicación web	127
Vista <i>tag</i>	128
Proceso de activación de una etiqueta en la aplicación web	129
Proceso de desactivación de una etiqueta en la aplicación web	131
Proceso de deshabilitación de una etiqueta en la aplicación web	133
Proceso de transformación de tareas mediante Cron	135
Vista <i>task_transformation_process</i>	135
Vista <i>period_exist</i>	136
Relación de permisos para obtención de datos, entre usuarios.....	137
Vista <i>users_perm</i>	138

Filtro de usuarios.....	138
Filtro de etiquetas.....	139
Resultados ya procesados de las estadísticas	139
Vista <i>statistics_tags</i>	140
Resultados ya procesados de la estadística de etiquetas por periodos.....	140
Vista <i>statistics_tags_periods</i>	141
Resultados ya procesados de la estadística de etiquetas por usuarios.....	141
Vista <i>statistics_tags_users</i>	142
Resultados ya procesados de la estadística de usuarios por periodos.....	143
Vista <i>statistics_users_periods</i>	143
Resultados ya procesados de la estadística de etiquetas por usuarios por periodos	144
Vista <i>statistics_tags_users_periods</i>	145
Logotipo de <i>The Time Bird</i>	146
Botones de menú de la aplicación móvil	146
Página principal de <i>The Time Bird</i>	147
Gráfico de ejemplo de Raphael, de tipo <i>pie</i>	148
Gráfico de ejemplo de <i>Raphael</i> , de tipo progresivo.....	148
Etapas de desarrollo <i>software</i> (Pruebas).....	150
Archivo de configuración para <i>The Time Bird</i> , de Apache	178
Típico mensaje de error durante la instalación de Drupal	179
Página inicial de Drupal	179
Ventana para la instalación de nuevo <i>software</i> de Android, para Eclipse	184
Configuración del <i>SDK</i> de Android.....	185
Aplicación <i>The Time Bird</i> para Android.....	186
Pantalla de inicio de sesión de la aplicación móvil	186
Menú de la pantalla de inicio de la aplicación móvil, primera opción	187
Pantalla Acerca de, de la aplicación móvil	188
Menú de la pantalla de inicio de la aplicación móvil, segunda opción	188
Menú de pantalla de etiquetas de la aplicación móvil, segunda opción	189

Ventana emergente para la creación de etiquetas, en la aplicación móvil	189
Pantalla de Etiquetas, de la aplicación móvil.....	190
Proceso seguido a la hora de crear una tarea	191
Menú contextual de una etiqueta en la aplicación móvil	192
Parte superior de la página principal de la aplicación web	193
Página de registro de <i>The Time Bird</i> , en la aplicación web.....	194
Petición de nueva contraseña, en la aplicación web	195
Página principal para usuarios registrados en la aplicación web	195
Página de cuenta de usuario, en la aplicación web	196
Estadística de etiquetas.....	196
Estadística de etiquetas por periodos	197
Estadística de usuarios por periodos	198
Estadística de etiquetas por usuarios.....	198
Estadística de etiquetas por usuarios por periodos	199
Página de sugerencias.....	199

Índice de tablas

Características destacadas, disponibles en <i>Tick</i>	15
Otras características de <i>tick</i>	16
Características destacadas, disponibles en <i>timr</i>	16
Otras características de <i>timr</i>	17
Características destacadas, disponibles en <i>Task Coach</i>	18
Otras características de <i>Time Coach</i>	18
Características destacadas, disponibles en <i>Officetime</i>	19
Otras características de <i>Officetime</i>	20
Características destacadas, disponibles en <i>tick</i>	21
Otras características de <i>FreshBooks</i>	21
Características destacadas, disponibles en <i>toggl</i>	22
Otras características de <i>toggl</i>	23

Características destacadas, disponibles en <i>Klok</i>	24
Otras características de <i>Klok</i>	25
Características destacadas, disponibles en <i>Cashboard</i>	25
Otras características de <i>Cashboard</i>	26
Características destacadas, disponibles en <i>Harvest</i>	27
Otras características de <i>Harvest</i>	27
Características destacadas, disponibles en <i>Paymo</i>	28
Otras características de <i>Paymo</i>	29
Características destacadas, disponibles en <i>Freckle</i>	30
Otras características de <i>Freckle</i>	31
Características destacadas, disponibles en <i>The Time Bird</i>	31
Visualización de ejemplo de <i>The Time Bird</i>	32
Costes totales de recursos humanos	37
Costes del <i>software</i> utilizado.....	39
Costes <i>del hardware</i> utilizado.....	39
Comparativa entre sistemas operativos	41
Contrato 1. Acerca de la aplicación	69
Contrato 2. Crear etiqueta	69
Contrato 3. Editar etiqueta	70
Contrato 4. Deshabilitar etiqueta	70
Contrato 5. Activar etiqueta	71
Contrato 6. Desactivar etiqueta	71
Contrato 7. Consultar ampliaciones.....	72
Contrato 8. Realizar sugerencia	73
Contrato 9. Registro de usuario	73
Contrato 10. Solicitud de contraseña.....	74
Contrato 11. Ver mi cuenta	74
Contrato 12. Editar mi cuenta	75
Contrato 13. Editar perfil	75

Contrato 14. Ver estadísticas (Etiquetas)	76
Contrato 15. Ver estadísticas (Etiquetas/Periodos)	76
Contrato 16. Ver estadísticas (Usuarios/Periodos)	77
Contrato 17. Ver estadísticas (Etiquetas/Usuarios)	77
Contrato 18. Ver estadísticas (Etiquetas/Usuarios/Periodos)	78
Algunas características de Android relevantes para <i>The Time Bird</i>	92
Pruebas referentes al inicio de sesión	151
Pruebas referentes al cierre de sesión	151
Pruebas referentes a la creación de etiquetas.....	151
Pruebas referentes a la edición de etiquetas	151
Pruebas referentes a la activación y desactivación de etiquetas	151
Pruebas referentes a la deshabilitación de etiquetas.....	152
Pruebas referentes a la compatibilidad de dispositivos.....	152
Pruebas referentes a la elaboración de las distintas estadísticas	152
Pruebas referentes a la elaboración de sugerencias.....	152
Pruebas referentes a comprobaciones en la interfaz de usuario	153
Pruebas referentes a comprobaciones de seguridad y permisos de usuario	153
Pruebas referentes a compatibilidad de la aplicación web.....	153
Pruebas referentes a rendimiento de la aplicación.....	153
Costes totales finales de recursos humanos	159

10.2. Instalación y configuración de un sitio web con Drupal

Antes de proceder al desarrollo de cualquier aplicación web con Drupal, hay que realizar una serie de tareas para tenerla alojada y configurada en un servidor, de forma correcta.

10.2.1. Configuración del servidor e instalación de Drupal

Configuración del servidor

La primera tarea es instalar los servicios necesarios en el servidor. En nuestro caso, tenemos un servidor con *Ubuntu 10.04* como SO base.

Los servicios a instalar, necesarios para alojar la aplicación web son *PHP* y *MySQL* (Bases de datos), así como un SGBD que permita la creación, edición, consulta y eliminación de las tablas que se necesiten administrar. El gestor utilizado ha sido *PHPMyAdmin*⁴¹.

Una vez instalados los servicios, es necesario crear la base de datos que utilizará Drupal. Para ello, accedemos vía web a la dirección `http://nuestrodominio/phpmyadmin` y creamos una nueva base de datos con el nombre deseado. En nuestro caso, **ttb**.

No será necesario acceder de nuevo al SGBD, a no ser que se quieran realizar copias de seguridad, borrados de datos masivos o el borrado de la base de datos, ya que Drupal ofrece una gestión completa de todos los datos almacenados en la base de datos.

Una vez tengamos la carpeta de la última versión de Drupal disponible (actualmente la 6.22), se coloca en la carpeta `/var/www/` del servidor. Seguidamente se accede a los archivos de configuración de Apache para enlazar el dominio a la carpeta de la aplicación, mediante la creación de un archivo con el nombre de la propia aplicación en el directorio `/etc/apache2/sites-available`. El archivo ha de contener los atributos necesarios para que Apache enlace al sitio.

```
<VirtualHost *:80>
    ServerAdmin thetimebird@atenealabs.com
    ServerName thetimebird.atenealabs.com
    ServerAlias thetimebird.atenealabs.com
    ServerAlias thetimebird.com
    DocumentRoot /srv/www/thetimebird.atenealabs.com/
    ErrorLog /srv/logs/thetimebird-error.log
    CustomLog /srv/logs/thetimebird-access.log combined
</VirtualHost>
```

Ilustración 130: Archivo de configuración para **The Time Bird**, de Apache

El sitio se activará utilizando el comando `a2ensite thetimebird.atenealabs.com`.

Instalación de Drupal

Una vez hecho lo anterior, ya se puede proceder a la instalación de Drupal accediendo desde cualquier navegador web al dominio adquirido.

La instalación se realiza en inglés, por defecto, aunque posteriormente se puede traducir a muchos otros idiomas, aunque no sea nuestro caso.

Una vez seleccionada la opción de idioma, Drupal comprueba si se puede instalar en el sistema, para ello se tienen que haber creado una carpeta llamada `files` y un archivo llamado `settings.php` en la dirección `/sites/default/` desde la carpeta de la aplicación y

⁴¹ **phpMyAdmin** es una herramienta escrita en *PHP* con la intención de manejar la administración de *MySQL* a través de páginas web, utilizando Internet.

darles permisos de escritura. En el caso de no realizado todos los pasos necesarios se mostrará el siguiente mensaje indicando los pasos a seguir para poder realizar la instalación correctamente.

- The Drupal installer requires that you create a settings file as part of the installation process.
 Copy the `./sites/default/default.settings.php` file to `./sites/default`.
 Change file permissions so that it is writable by the web server. If you are unsure how to grant file permissions, please consult the [on-line handbook](#).
 More details about installing Drupal are available in `INSTALL.txt`.
- The directory `sites/default/files` does not exist. An automated attempt to create this directory failed, possibly due to a permissions problem. To proceed with the installation, either create the directory and modify its permissions manually, or ensure that the installer has the permissions to create it automatically. For more information, please see `INSTALL.txt` or the [on-line handbook](#).

Ilustración 131: Típico mensaje de error durante la instalación de Drupal

Acabados de crear los archivos necesarios y habiendo dado permiso de lectura y escritura al archivo `settings.php`, Drupal nos pide que introduzcamos el nombre de la base de datos (en nuestro caso **ttb**) y el usuario y contraseña para poder acceder al SGBD y realizar los cambios necesarios. Finalizado este proceso, nos encontraremos en la página frontal del gestor de contenidos, con la que se puede empezar a trabajar.

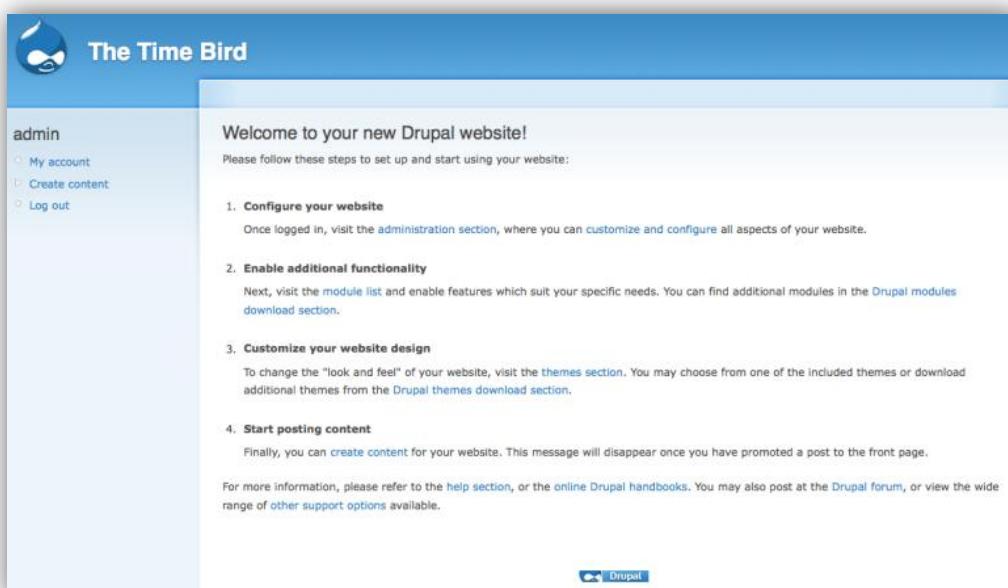


Ilustración 132: Página inicial de Drupal

Nota

Si por un casual, se tuviera que realizar una instalación de una aplicación web ya desarrollada, el proceso de instalación sería algo distinto, ya que una vez movida la carpeta al directorio correspondiente (`/var/www/`) y creado el archivo de configuración de Apache correspondiente, no se realizaría la instalación vía web, sino que se tendría que acceder al archivo `settings.php` alojado en el directorio `/sites/default/` navegando desde la carpeta principal de la aplicación.

Una vez allí, se tendría que abrir el archivo, y modificar la línea donde pone:

`$db_urb='mysqli://usuario_bd:password_bd@localhost/nombre_bd` donde:

- `usuario_bd` es el usuario con el que se accede al SGBD.
- `password_bd` es la contraseña utilizada para acceder SGBD.
- `nombre_bd` es el nombre de la base de datos utilizada, en nuestro caso, **ttb**.

Ajustes de memoria

Drupal tiene por defecto asignada una cantidad de memoria para las tareas realizadas en la aplicación web. Ésta es baja y aunque en principio es suficiente para la mayoría de aplicaciones, en **The Time Bird** puede darse el caso en el que se consuma una gran cantidad de memoria, debido a la creación continua de nodos debido al proceso de análisis del sitio web. Para ajustar la memoria necesaria, fijada en 512 Megabytes, se puede realizar de dos modos diferentes.

- Modificación del archivo `php.ini` situado por defecto en la ruta `/etc/php5/apache2` del servidor. Una vez abierto el archivo, hay que establecer un nuevo límite de memoria. Encontraremos una línea que por defecto nos señala una memoria límite de 16 Megabytes, por lo que se tendrá que modificar por:

```
memory_limit = 512M
```

- Modificación del archivo oculto `.htaccess` que se encuentra en el directorio raíz de la aplicación web, en el que hay que añadir la siguiente línea de código:

```
php_value memory_limit 512M (PHP 5)
```

Si se utiliza la versión de **PHP 4**, habría que añadir una línea de código al archivo de configuración de Drupal `settings.php` situado en el directorio `/sites/default/`:

```
ini_set('memory_limit','512M')
```

Habilitar URLs limpias

La habilitación de las *URLs* limpias permite que todo el contenido vía *URL* sea indexado más fácilmente por los distintos motores de búsqueda existentes.

Por defecto, en Drupal las *URLs* aparecen con la cadena `?q=` que no es amigable con los buscadores. Es decir, que páginas escritas de la forma `http://ttb.com/?q=node/32` serán ignoradas.

Para habilitarlas en el servidor web, tenemos que realizar los siguientes pasos:

1. Abrir una consola y escribir `apache2ctl -M` para saber si el módulo *rewrite* está disponible. Si se da el caso negativo, podemos introducir la dirección siguiente en el navegador y así confirmar que existe el *rewrite.load*:

```
/etc/apache2/mods-available
```

Una vez confirmado, es necesario crear un enlace simbólico desde *mods-enabled* a *mods-available*. Para ello es necesario introducir en la consola:

```
cd /etc/apache2/mods-enabled
```

```
sudo ln -s ../mods-available/rewirite.load
```

2. Entrar en la carpeta `/etc/apache2/sites-enabled`, y modificar en el archivo `sites-enabled`, todas las líneas donde salga `AllowOverride None` por `AllowOverride All` y salga el directorio donde está instalado nuestra aplicación web.

3. Por último solo hace falta reiniciar *Apache* mediante el comando:

```
/etc/init.d/apache2/restart
```

Una vez hemos hecho esto, solo falta activar las *URLs* limpias en Drupal, y para ello, solo tenemos que navegar por **Administración → Configuración del sitio → URLs limpias** y activar la opción correspondiente.

10.2.2. Configuración inicial de la aplicación web

Una vez nos encontramos en la web, hay diferentes opciones de configuración que cabe tener presentes a la hora de realizar los primeros pasos. Las más importantes, están en el menú de administración y son la **configuración del sitio y construcción del sitio**.

Configuración del sitio

En la configuración del sitio, encontraremos las opciones de configuración genéricas de toda la aplicación, así como las opciones de configuración que aporten otros módulos instalados. Se podrá cambiar el logo, el título y la misión de la web. También se puede

cambiar el mensaje de pie de página, opciones de rendimiento y mantenimiento del sitio, idioma, y otras opciones de menor importancia.

Construcción del sitio

Antes de empezar a desarrollar cualquier aplicación, muy probablemente sea necesaria la instalación de muchos módulos que permitan realizar las funcionalidades deseadas, la creación de roles y permisos de acceso para cada uno de ellos, así como otras opciones como la creación de bloques y otros elementos vistos en el capítulo de diseño.

Cabe destacar que la construcción del sitio, ya sea instalación o configuración de los módulos, solo se puede llevar a cabo por el administrador, a no ser que éste haya creado permisos para que otros roles puedan realizar dichas tareas. Los carpetas correspondientes a los módulos descargados podrán ser introducidos en la carpeta de módulos de Drupal (`/sites/all/modules/`). Su núcleo, viene integrado con varios módulos activados por defecto, así como otros opcionales. A continuación se procede a describir de forma resumida, la utilidad tanto de los módulos obligatorios que siempre tendrán que estar activados, como los opcionales, para dar una idea de las posibilidades que tiene por defecto este gestor:

Obligatorios

- **Block:** Sirve para gestionar los bloques o cajas de contenido, que utiliza Drupal para mostrar los elementos de navegación, inicio de sesión de usuarios, etc.
- **Filter:** Sirve para gestionar el contenido de toda la aplicación que se permite filtrar a los buscadores.
- **Node:** Permite que se puedan crear nuevos tipos de contenido en la aplicación.
- **System:** Gestión de la configuración general de la aplicación, realizada por los administradores.
- **User:** Permite gestionar el sistema de acceso y registro de los usuarios de la aplicación.

Opcionales (los más importantes)

- **Comment:** Permite realizar comentarios al contenido creado en la aplicación.
- **Menu:** permite la creación de menús de navegación en la aplicación, lo que lo hace un módulo prácticamente obligatorio para toda aplicación web.
- **Taxonomy:** Permite categorizar el contenido de Drupal en diferentes vocabularios con términos definidos por los usuarios
- **Help:** Activa la ayuda *online* en la aplicación.

- **Path:** Permite renombrar los enlaces del contenido creado en la aplicación, algo realmente importante si se quiere tener una presentación bonita de las URL's. Además permite crear patrones de reemplazo para cada uno de los tipos de contenido.
- **Update status:** Módulo muy útil si la aplicación va a necesitar una continua actualización de sus servicios y mejora de sus funcionalidades. Permite comprobar la disponibilidad de actualizaciones en los módulos usados en la aplicación, así como actualizaciones del sistema base.

A partir de aquí, es descubrir un mundo nuevo, e iniciar un aprendizaje lento, debido a la gran cantidad de posibilidades que ofrece un gestor de contenidos como Drupal.

10.3. Instalación y configuración de un entorno Android

Antes de proceder con el desarrollo de cualquier aplicación móvil con Android, necesitamos instalar y configurar un entorno de trabajo, que permitirá al desarrollador depurar el código introducido, y a la vez, probar sus aplicaciones. Para ello, es necesario seguir una serie de pasos, descritos a continuación (cabe destacar que el proceso seguido es para la instalación del entorno en **Mac OS X**).

Eclipse⁴²

Lo primero, es descargar de la siguiente dirección (a no ser que ya esté instalado en el sistema) el *IDE* con el que se va a trabajar, llamado Eclipse:

<http://www.eclipse.org/downloads/packages/eclipse-classic-362/heliossr2>

Pluguin ADT

A continuación procedemos a instalar el *plugin* ADT (*Android Development Toolkit*) para Eclipse, diseñado para integrar completamente un entorno con el que poder desarrollar aplicaciones Android.

Es tan simple como ir a la barra de menú de Eclipse **Help → Install New Software...** y añadir la dirección correspondiente que se observa en la siguiente ilustración.

⁴² **Eclipse** es un entorno de desarrollo integrado de código abierto multiplataforma para desarrollar aplicaciones. Esta plataforma, típicamente ha sido usada para desarrollar entornos de desarrollo integrados, como el IDE de Java llamado *Java Development Toolkit* y el compilador *ECJ* que se entrega como parte de Eclipse.

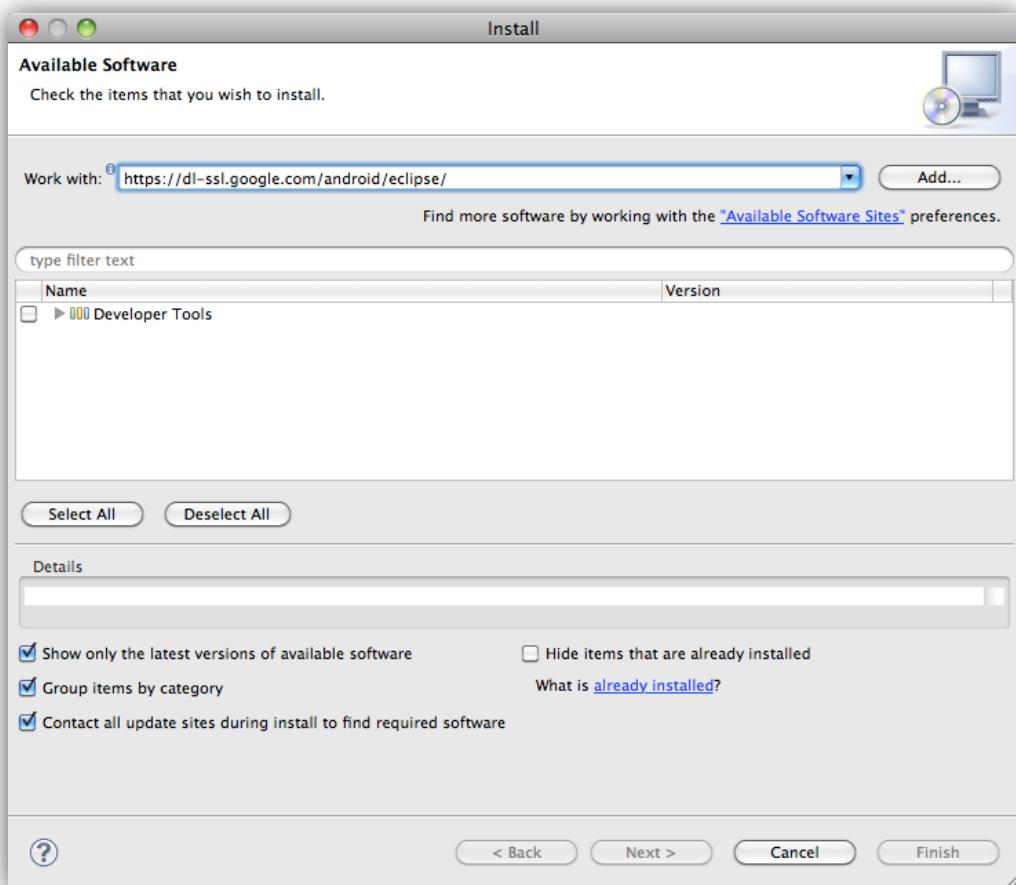


Ilustración 133: Ventana para la instalación de nuevo software de Android, para Eclipse

Una vez cargados los paquetes, se pueden descargar todas las librerías disponibles que nos sean necesarias.

SDK

También es necesario descargar e instalar el entorno de desarrollo *SDK*. Para descargarlo hay que acceder a la dirección <http://developer.android.com/sdk/index.html>. Una vez descomprimido el archivo, tenemos decirle a eclipse donde lo puede encontrar, y para ello, están simple como abrir Eclipse, e ir en la barra de menú a **Eclipse → Preferencias** y clicar en la pestaña **Android**. Una vez allí, vemos un cuadro de texto, donde hay que buscar el directorio descomprimido del *SDK*, en nuestro caso, `/Users/cesaryomismo/Android-sdk-mac_x86`.

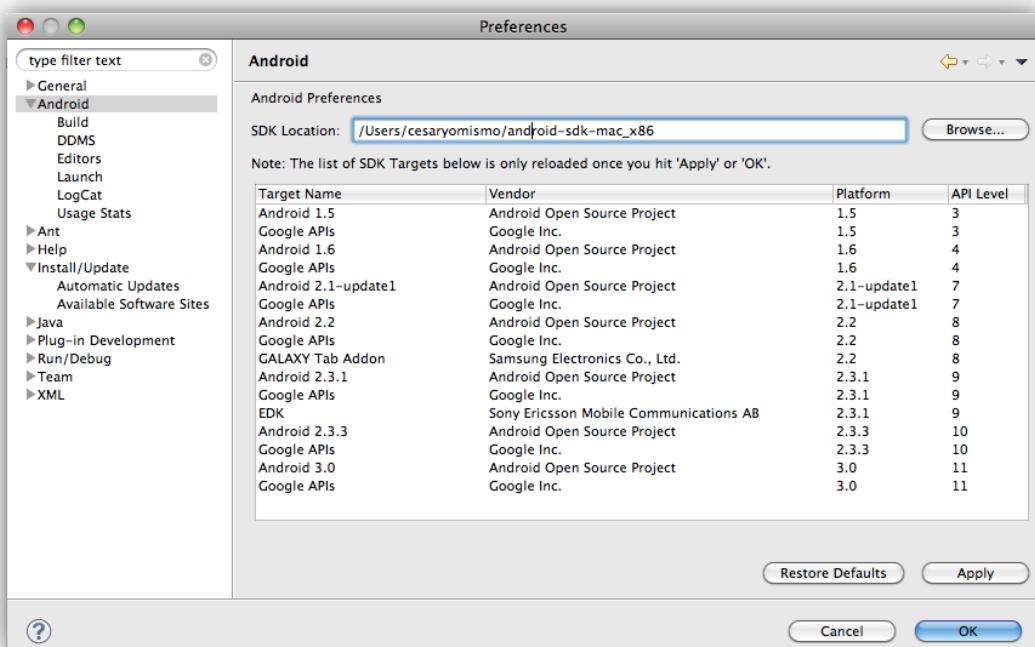


Ilustración 134: Configuración del *SDK* de Android

Para instalar nuevas herramientas de desarrollo, simplemente hay que ir a **Window → Android SDK and AVD Manager** en la barra de menú, y seleccionar de todas ellas, las que sean necesarias. En nuestro caso, sería necesario el *SDK 2.3* de Android, ya que es con el que se va a desarrollar la aplicación.

Una vez instaladas las herramientas necesarias, ya es posible crear un nuevo proyecto Android, y empezar a desarrollar nuestra aplicación.

10.4. Manual de usuario

A continuación se va a describir de forma detallada, cada una de las funcionalidades que ofrece ***The Time Bird***, tanto en la aplicación móvil como web.

Cabe mencionar que la sencillez del producto hace relevante el uso de un manual, pero uno de los objetivos del proyecto, es que todo usuario pueda consultar siempre que lo desee, el modo de uso de cada una de las funcionalidades ofrecidas.

10.4.1. Aplicación móvil

Una vez registrados en la página web de ***The Time Bird***, <http://thetimebird.com> procedemos a descargar la aplicación para Android, disponible en el *Android Market*.

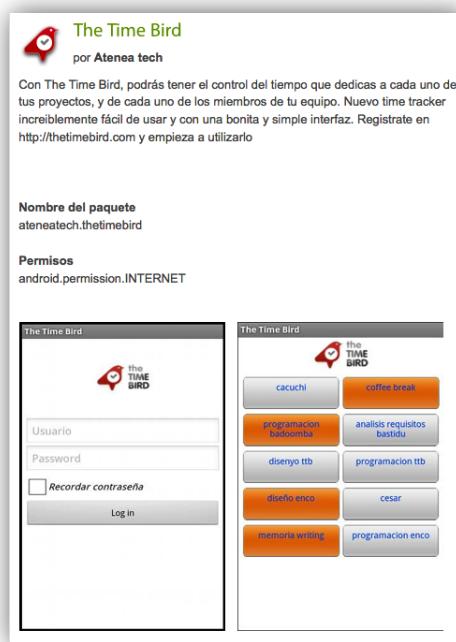


Ilustración 135: Aplicación ***The Time Bird*** para Android

Inicio de Sesión

Una vez instalada y abierta la aplicación, nos encontramos con la pantalla de inicio de sesión de usuario.



Ilustración 136: Pantalla de inicio de sesión de la aplicación móvil

Una vez aquí, hay tres posibles acciones a realizar, dos de ellas accediendo desde el menú de Android.

a) Inicio de Sesión

Para poder acceder a nuestros datos personales, hay que iniciar sesión con un nombre de usuario y una contraseña obtenidas del registro de la aplicación web, por lo que, una vez registrados, es tan sencillo como introducir nuestro nombre de usuario en el primer campo, introducir nuestra contraseña en el segundo y darle al botón de **Inicio de sesión**. Es recomendable marcar la opción de recordar contraseña si vamos a ser los únicos en utilizar el terminal, o vamos a utilizar la aplicación con bastante frecuencia.

Si no hay problemas de conectividad y hemos introducido nuestros datos de forma correcta, el inicio de sesión se realizará correctamente.

b) Más sobre la aplicación

Siempre que lo deseemos, podemos consultar los datos de interés general que el autor ha puesto a disposición para todos los usuarios. Para acceder a ellos, lo único que hay que acceder es acceder al menú y clicar la opción **Acerca de**.



Ilustración 137: Menú de la pantalla de inicio de la aplicación móvil, primera opción

Aparecerá una pantalla como la que se muestra a continuación, en la que podemos encontrar una pequeña explicación de lo que ofrece **The Time Bird**, así como agradecimientos del autor.



Ilustración 138: Pantalla **Acerca de**, de la aplicación móvil

Lo único que tenemos que hacer para volver a la pantalla inicial, es apretar el botón de **OK**.

c) Cerrar la aplicación

Si queremos cerrar la aplicación y así no seguir utilizando memoria en el sistema, solo es necesario seleccionar la opción **Salir**.

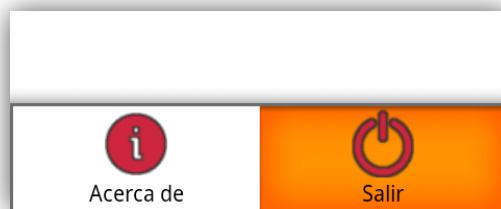


Ilustración 139: Menú de la pantalla de inicio de la aplicación móvil, segunda opción

Pantalla principal

Una vez el usuario hayamos iniciado una sesión, aparecerá una pantalla con las etiquetas que hayamos creado hasta el momento. Si es la primera vez que utilizamos **The Time**

Bird, la pantalla aparecerá en blanco, por lo que será necesario añadir nuevas etiquetas, que irán apareciendo en pantalla.

A continuación se describen las funcionalidades que podemos realizar, una vez iniciada la sesión.

a) Crear etiquetas

Una de las principales funcionalidades de **The Time Bird**, es que pueda crear etiquetas, haciendo éstas referencia a cualquier proyecto, cliente, proceso u otro concepto en el que se quiera empezar a trabajar. Para ello es tan simple apretar el botón de menú, y seleccionar **Nueva etiqueta** en la barra de menú.

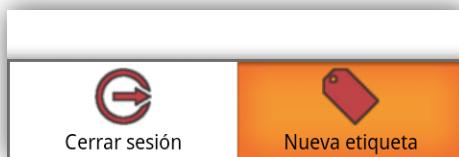


Ilustración 140: Menú de pantalla de etiquetas de la aplicación móvil, segunda opción

Una vez hecho, saldrá una ventana emergente, como se muestra en la siguiente ilustración.

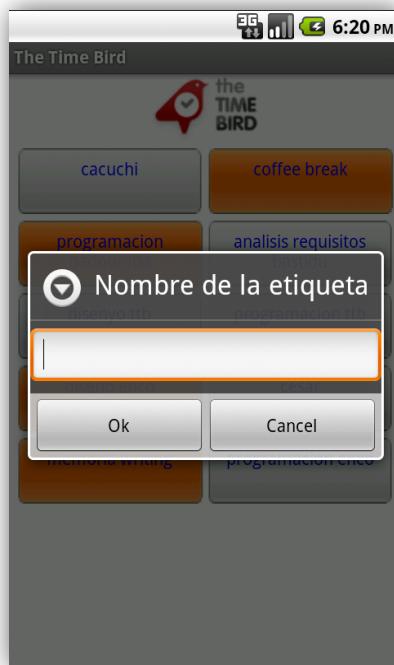


Ilustración 141: Ventana emergente para la creación de etiquetas, en la aplicación móvil

Introducimos el título que queramos asociar a la etiqueta, y le damos a **Ok**. Una vez hecho esto, se creará un nuevo botón en la pantalla con su respectivo título.

Cabe mencionar, que se realizan comprobaciones de nombre de etiquetas. Al ser una de las opciones su deshabilitación (opción **d**) y así no cargar la pantalla con muchos, si intentamos crear una **Etiqueta** que ya existe y está activada, la aplicación le mostrará un mensaje informativo de dicho caso. En el caso en que se esté intentando crear una etiqueta ya existente en el sistema, pero deshabilitada anteriormente, se realizará la activación de la etiqueta, y se podrá seguir contabilizando el tiempo, sumado al que ya se había contabilizado anteriormente.

b) Activar/Desactivar etiquetas

Las etiquetas creadas, se pueden activar y desactivar cuando deseemos. Nuestra faena es únicamente la de apretar cada uno de los botones correspondientes a las etiquetas que se desean activar. De esta forma, el sistema registra el evento y se van creando tareas con las etiquetas seleccionadas, de forma automática,

Cabe destacar, que tenemos un margen de tiempo para rectificar en nuestras decisiones, ya que al crearse las tareas por minutos, desde la activación de la primera etiqueta podremos hacer tantos cambios como queramos en ese minuto, antes de empezar con una nueva tarea. A continuación se puede observar una imagen con etiquetas activadas (color naranja) y desactivadas (color gris).



Ilustración 142: Pantalla de Etiquetas, de la aplicación móvil

Para que entendamos mejor el proceso seguido por el sistema a la hora de crear tareas, se ha realizado un esquema en el que podemos ver claramente el efecto de nuestras acciones.

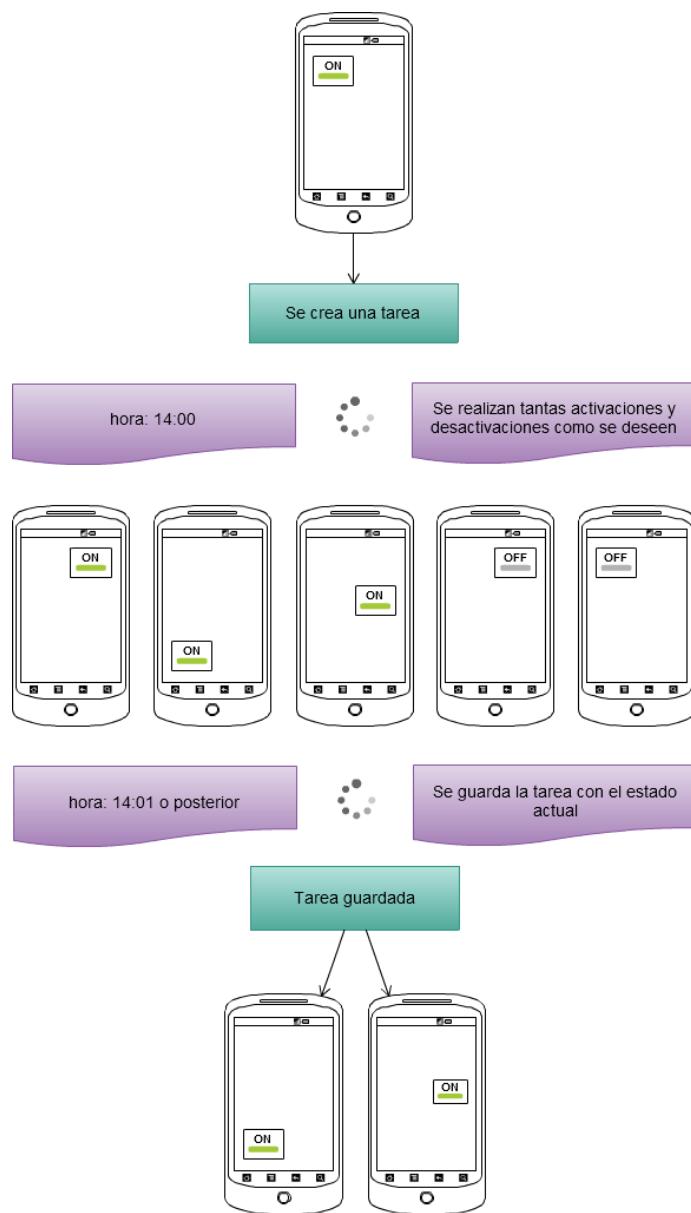


Ilustración 143: Proceso seguido a la hora de crear una tarea

c) Editar etiquetas

Si queremos editar el nombre de alguna de nuestras etiquetas creadas, es tan fácil como mantener pulsado el botón correspondiente a la etiqueta deseada, durante un par de segundos. Una vez haya aparecido el menú como el que se puede ver en la imagen a continuación, seleccionamos la opción **Editar**, y aparecerá una ventana emergente exactamente igual a la utilizada a la hora de crear etiquetas (opción **a**), donde podremos escribir el nuevo título deseada para la etiqueta seleccionada.

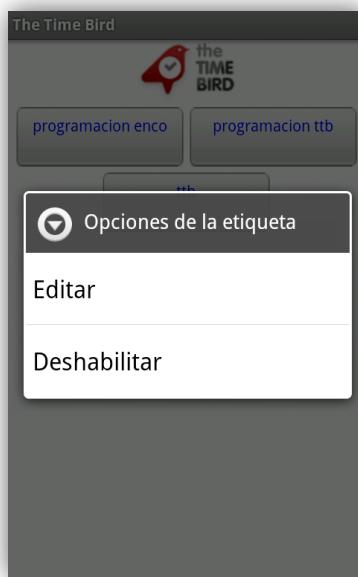


Ilustración 144: Menú contextual de una etiqueta en la aplicación móvil

d) Deshabilitar etiquetas

Otra opción es la de deshabilitar las etiquetas creadas con anterioridad. Para hacerlo, tenemos que seguir los mismos pasos que para editarlas (opción c). Mantenemos pulsado el botón de cualquiera de las etiquetas durante un par de segundos. Una vez hecho esto aparecerá un menú y tenemos que seleccionar la opción **Deshabilitar**. La etiqueta desaparecerá de la pantalla principal y no se volverá a habilitar a no ser que se cree una etiqueta con el mismo nombre.

e) Cerrar sesión

Para cerrar la sesión en **The Time Bird**, es tan fácil como ir a la opción de menú, y seleccionar la opción **Cerrar sesión** en la barra de menú.

Si cerramos la sesión, no perderemos datos, ya que siempre que iniciamos una sesión se cargan todas las etiquetas con el estado (activa/inactiva) en el que las dejamos por última vez.

10.4.2. Aplicación web

La aplicación web, tiene una navegabilidad buena y es sencilla de utilizar, por lo que su funcionamiento no es complejo. Su descripción, al igual que en la aplicación móvil, se va a dividir en las pantallas que puede visitar el usuario.

Para simplificar el manual, no se mencionarán las funcionalidades meramente informativas o poco relevantes en la desenvoltura de la aplicación.

Pantalla de inicio

Una vez hemos accedido al enlace de la aplicación web <http://thetimebird.com>, podemos realizar una serie de acciones, aparte de leer toda la información que el autor ha puesto a nuestra disposición y que explica ligeramente las utilidades de ***The Time Bird*** (incluso sin la necesidad de habernos registrado).



Ilustración 145: Parte superior de la página principal de la aplicación web

a) Registro

Lo primero que tenemos que hacer para poder disfrutar de la mayoría de funcionalidades de importancia de ***The Time Bird***, es registrarnos en la aplicación web. Para ello, lo único que hace falta es darle a la opción **Registrarse**. Una vez hecho esto, aparecerá una pantalla como la que se muestra a continuación.

Create new account [Log in](#) [Request new password](#)

Username: *

Spaces are allowed; punctuation is not allowed except for periods, hyphens, and underscores.

E-mail address: *

A valid e-mail address. All e-mails from the system will be sent to this address. The e-mail address is not made public and will only be used if you wish to receive a new password or wish to receive certain news or notifications by e-mail.

Usuarios que pueden ver tus datos:

<input type="checkbox"/> admin
<input type="checkbox"/> Raymon
<input type="checkbox"/> Cesar
<input type="checkbox"/> acanadas
<input type="checkbox"/> elros10
<input type="checkbox"/> Georgina
<input type="checkbox"/> brendabsg
<input type="checkbox"/> dwightmml
<input type="checkbox"/> menor
<input type="checkbox"/> kbza55
<input type="checkbox"/> wakumaku

[Create new account](#)

Ilustración 146: Página de registro de **The Time Bird**, en la aplicación web

Una vez aquí, introducimos el nombre de usuario que queramos utilizar en el sitio web, y nuestra dirección de correo electrónico, a donde será enviada la contraseña temporal y un enlace con el que podremos cambiarla por la deseada.

Hecho esto, procedemos a seleccionar la lista de usuarios que podrán ver nuestros datos. Esto es así, por si trabajamos en equipo, y alguien más necesita acceder a nuestros datos, y así poder obtener estadísticas sobre nuestro trabajo. En **The Time Bird**, no existen cargos de trabajo, preferimos una jerarquización de permisos en la lectura de datos, en la que a los usuarios que le hayamos dado permiso para ver nuestros datos, también verán los datos de aquellas personas que hayan aceptado que nosotros viéramos sus datos

b) Inicio de sesión

Para poder acceder a las estadísticas creadas a partir de nuestros datos, es necesario que hayamos iniciado una sesión con nuestro nombre de usuario y contraseña. Para ello, lo único que hay que hacer es introducir los campos mencionados y darle al botón **Log in**.

c) Pedir una nueva contraseña

El mero hecho de tener creados muchos usuarios y contraseñas en diferentes sitios web, a veces hace que se nos olvide alguno de ellos. Drupal, permite recuperar nuestra

contraseña, enviándonos un correo a nuestra dirección, con dicha información. Para ello, lo único que hay que hacer es clicar sobre el enlace **Request new password** que encontramos debajo del botón de inicio de sesión y como se puede ver en la siguiente ilustración, lo único que tenemos que hacer es introducir nuestra cuenta de correo.

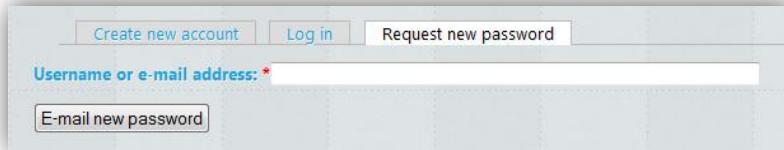


Ilustración 147: Petición de nueva contraseña, en la aplicación web

Pantalla principal de usuario

Una vez hayamos entrado en la aplicación web como usuarios registrados, como podemos ver en la siguiente imagen, hay un menú principal y un menú de navegación lateral, con los que podremos acceder a las distintas funcionalidades.

Cabe mencionar que el acceso a páginas como la de **Ampliaciones** o **Gráficas de ejemplo**, no serán descritas, ya que son meramente informativas.



Ilustración 148: Página principal para usuarios registrados en la aplicación web

a) **Mi cuenta**

Siempre que hayamos iniciado sesión, podemos acceder a los datos de nuestra cuenta. Para ello, solo hace falta elegir el enlace **Mi cuenta** en la parte superior derecha de la aplicación. Veremos una página datos como los que se muestran a continuación.

The screenshot shows a user profile page with the following details:

- Profile:** Submitted by menor on April 27, 2011, at 0:58.
- Users who can view your data:** Cesar, elros10, lotells, Androjuegos.
- History:** Member for 1 week 6 days.

Ilustración 149: Página de cuenta de usuario, en la aplicación web

La información que podremos ver, es el nombre de usuarios a los que hemos dado permiso para acceder a estadísticas sobre nuestros datos, así como el tiempo que llevamos registrados en la aplicación web. Si seleccionamos la pestaña **Editar**, podremos cambiar nuestro correo electrónico y/o contraseña, y si seleccionamos la pestaña **Perfil**, podremos cambiar los usuarios a los que damos permiso para obtener nuestros datos.

b) Estadísticas

La visualización de estadísticas sobre los datos enviados a la aplicación web, es una de las funcionalidades principales en **The Time Bird**. Se pueden obtener hasta cinco tipos de estadísticas diferentes, todas ellas descritas a continuación.

- **Etiquetas:** Si deseamos obtener todo el tiempo dedicado a cualquiera de las etiquetas creadas, lo único que tenemos que hacer, es ir al enlace lateral de **Etiquetas**, seleccionar las etiquetas deseadas en el filtro superior, y darle al botón **Apply**. Una vez hecho esto se obtendrá por cada una de las etiquetas seleccionadas, un gráfico como el mostrado a continuación.



Ilustración 150: Estadística de etiquetas

- **Etiquetas / Periodos:** Si deseamos obtener una gráfica con el progreso de tiempo dedicado a ciertas etiquetas, lo único que tenemos que hacer es ir al enlace lateral de **Etiquetas / Periodos**, seleccionar las etiquetas deseadas y el tiempo de inicio y final en los filtros superiores y darle al botón **Apply**. Una vez hecho esto se obtendrán dos gráficos; el primero de ellos con el progreso del tiempo total dedicado a cada etiqueta entre los días seleccionados, y el segundo, con los porcentajes del tiempo de uso de las etiquetas seleccionadas.

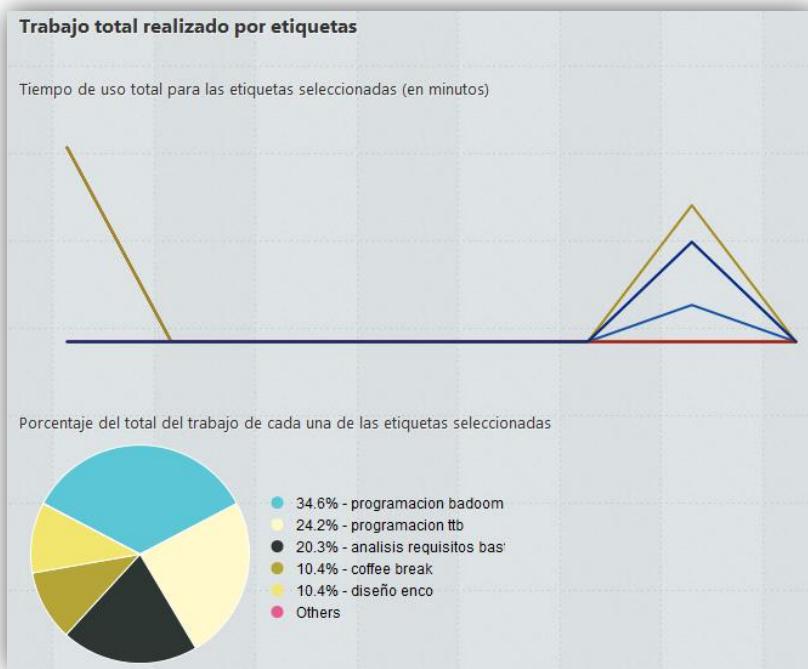


Ilustración 151: Estadística de etiquetas por períodos

- **Usuarios / Periodos:** Si deseamos obtener una gráfica con el progreso del tiempo total dedicado a todas las etiquetas de ciertos usuarios, lo único que tenemos que hacer es ir al enlace lateral de **Usuarios / Periodos**, seleccionar los usuarios deseados y el tiempo de inicio y final en los filtros superiores y finalmente darle al botón **Apply**. Una vez hecho esto, se obtendrán dos gráficos; el primero de ellos con el progreso de tiempo total dedicado por cada usuario, a todas sus etiquetas, durante los días seleccionados, y el segundo, con los porcentajes de tiempo de trabajo total de cada uno de los usuarios seleccionados.

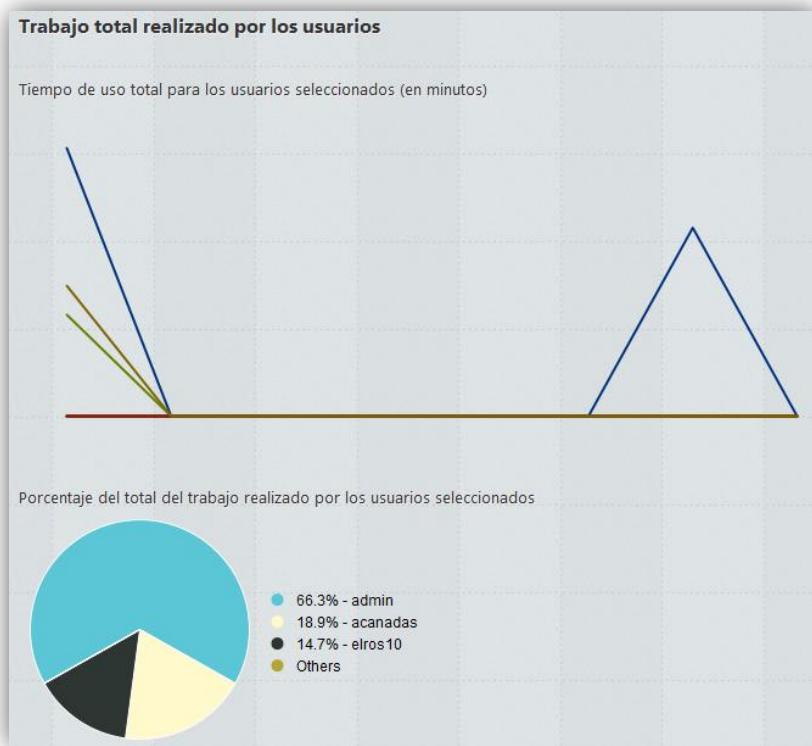


Ilustración 152: Estadística de usuarios por períodos

- **Etiquetas / Usuarios:** Si deseamos obtener una gráfica con el tiempo total dedicado por ciertos usuarios a ciertas etiquetas, lo único que tenemos que hacer, es ir al enlace lateral de **Etiquetas / Usuarios**, seleccionar las etiquetas y usuarios deseados en los filtros superiores y darle al botón **Apply**. Una vez hecho esto, se obtendrá por cada una de las etiquetas seleccionadas, un gráfico con el tiempo que le ha dedicado por cada uno de los usuarios seleccionados.



Ilustración 153: Estadística de etiquetas por usuarios

- **Etiquetas / Usuarios / Periodos:** Si deseamos obtener una gráfica con el progreso del tiempo dedicado por ciertos usuarios a ciertas etiquetas, lo único que tenemos que hacer es ir al enlace lateral de **Etiquetas / Usuarios / Periodos**, seleccionar las etiquetas deseadas, usuarios deseados, el tiempo de inicio y final en los filtros superiores y darle al botón **Apply**. Una vez hecho esto se obtendrán tantos gráficos como etiquetas seleccionadas, siendo cada uno de ellos como el mostrado a continuación.



Ilustración 154: Estadística de etiquetas por usuarios por períodos

c) Sugerencias

Uno de los requerimientos a la hora de elaborar la aplicación web, es ofrecer las herramientas necesarias para estar en continuo contacto con los usuarios finales. Por ello tenemos a nuestra disposición un formulario de sugerencias, que se puede utilizar siempre que queramos realizar alguna sugerencia sobre ciertas mejoras, o incluso ampliaciones y/o errores del sistema. La sugerencia realizada será atendida por el responsable de la aplicación, que nos contestará si así lo cree conveniente.

Ilustración 155: Página de sugerencias

d) Cerrar sesión

Para cerrar la sesión actual, lo único que tenemos que hacer, es clicar sobre el enlace que así lo indica, en la parte superior derecha de la aplicación.