UNIVERSIDAD
REY JUAN CARLOS

ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA INFORMÁTICA

MÁSTER OFICIAL EN SISTEMAS TELEMÁTICOS E INFORMÁTICOS

Curso Académico 2010/2011

Trabajo Fin de Máster

DeTraS: Ampliación, promoción y creación de una comunidad

Autor: Edmundo Álvarez Jiménez
Tutor: Gregorio Robles Martínez

A Alber, por los años que disfruté de su generosidad y sincera amistad.
"Lo que hacemos en la vida, tiene su eco en la eternidad."

Agradecimientos

Tras la culminación de este Trabajo Fin de Máster, que pone punto y final a mis andanzas en el Máster Oficial de Sistemas Telemáticos e Informáticos, cierro una etapa de mi vida muy interesante, en la que he adquirido bastantes conocimientos y de la que me llevo buenos recuerdos y amistades.

Antes de nada, me gustaría agradecer a Gregorio por su apoyo y guía durante todos estos meses, siendo partícipe del trabajo realizado y los resultados obtenidos. Asimismo, quisiera agradecer el apoyo ofrecido desde el GSyC y las enseñanzas de los profesores que han conseguido que pudiese alcanzar esta meta.

Sin duda, quiero agradecer a mis padres y hermana su comprensión, dedicación y ánimo durante estos años. Igualmente quiero dar las gracias a mi familia, por su apoyo continuo.

También me gustaría agradecer especialmente a Alberto —o Alber, como solía llamarlo—, porque siempre me demostró su amistad y porque admiro la resistencia y coraje con la que se enfrentó a la vida. Sin duda, un ejemplo a seguir, especialmente en estos tiempos difíciles. Desde aquí envío un abrazo muy fuerte a su familia y novia.

Por supuesto, no puedo dejar de mencionar a los amigos que me han acompañado todos estos años, a los que agradezco su apoyo y fidelidad, y a los nuevos amigos que he podido hacer en los últimos meses, cuya amistad espero dure por muchos años.

Gracias a todos.

Resumen

Las nuevas tecnologías suponen pasar mucho más tiempo frente a ordenadores de lo que lo habíamos hecho con anterioridad con ninguna otra máquina o herramienta. Además, como los computadores permiten realizar una gran cantidad de tareas, en ocasiones es muy sencillo perder la noción del tiempo. Ese tiempo que es, a su vez, algo intangible e imparable, y que los humanos siempre hemos necesitado medir y administrar correctamente.

Los ordenadores, esos dispositivos con los que pasamos tanto tiempo, también pueden ayudarnos a administrar mejor el tiempo y conocer a qué lo dedicamos. Un ejemplo de ello es el sistema DeTraS, cuyo desarrollo inició Carlos García Campos, en el año 2006, y que se pretende mejorar y ampliar a lo largo del presente trabajo.

Aunque la tecnología en ocasiones resulta fría y nos aleja del mundo real durante demasiado tiempo, también nos ofrece nuevas oportunidades de relación con otras personas. Existen, hoy en día, numerosas redes sociales con millones de usuarios, donde es muy fácil encontrar a tus amistades y relacionarte con ellas a través de fotos, mensajes, pequeñas aplicaciones y un largo etcétera. No obstante, hasta ahora no se ha tratado de hacer que las personas se relacionen en torno a aplicaciones. Esas aplicaciones que se usan a diario en miles de equipos informáticos. Esto, que en principio puede parecer una tontería, podría ofrecer nuevas oportunidades de conocer herramientas que nos ayuden en el día a día, así como facilitar la resolución de problemas que encontramos mientras las utilizamos. Todo ello en un ambiente ameno, puesto que mientras obtenemos esos beneficios estaríamos «jugando» con nuestros amigos. Esto es lo que pretende CheckApp, herramienta que se introducirá más adelante.

A lo largo del presente trabajo, se ofrece una visión más amplia de los problemas que se pretenden resolver, se presentan distintas alternativas que han surgido para solucionarlos y se pormenoriza todo el proceso de desarrollo e investigación seguido para conseguir los objetivos marcados.

Índice general

Índice de figuras

Índice de tablas

Índice de listados

Capítulo 1

Introducción

Antes de abordar los objetivos y labores llevados a cabo durante la realización del trabajo aquí descrito, es necesario conocer la idea de la que surgió y el contexto que lo rodea.

La primera etapa del desarrollo software, así como del propio conocimiento humano, consiste en identificar un problema existente que, mediante un proceso más o menos elaborado, se pretende resolver. El presente trabajo no es una excepción a este hecho, y el problema al que se enfrenta es algo complejo e intangible: el tiempo. Concretamente, la idea detrás del trabajo es conocer, de una manera objetiva y rigurosa, en qué tareas invertimos el tiempo que pasamos empleando ordenadores. Por supuesto, esta idea no es más que el punto de partida, cuya evolución podrá observar a lo largo del presente documento.

Con el objeto de dar a conocer de una forma más precisa el contexto que rodea a este trabajo, se ilustrarán, a lo largo del capítulo en curso, diversos ámbitos relacionados con el trabajo presentado, así como múltiples técnicas e ideas que se han aplicado y se aplican para solventar los problemas que se presentan en esos campos.

1.1. Estimación de esfuerzo

Uno de los mayores problemas a los que se enfrentan los gestores de software al planificar un nuevo proyecto informático consiste en la estimación del trabajo a realizar, los recursos necesarios y el tiempo transcurrido hasta su conclusión. Este proceso de evaluación es conocido dentro del campo de la ingeniería del software como estimación de esfuerzo o estimación de costes.

Hemos de tener en cuenta que, a diferencia de lo que ocurre en otras industrias,

en las que se puede conocer con bastante exactitud el coste de un producto a partir del coste de las piezas que lo componen y el tiempo necesario para su creación, siendo estos datos comúnmente conocidos, en el mundo del software no ocurre lo mismo. En este caso, únicamente es posible realizar una estimación del esfuerzo que llevará la construcción de un sistema, algo que determinará en gran medida el coste del mismo.

A la hora de realizar una estimación de esfuerzo siempre estamos asumiendo cierto grado de incertidumbre, que variará en función de diversos factores, siendo los más determinantes: la complejidad del problema, el tamaño del software y la estabilidad de los requisitos.

Evidentemente, a pesar de que el grado de incertidumbre del proyecto no sea muy elevado, sigue siendo muy complicado conocer cuánto esfuerzo va a requerir desarrollar un determinado sistema antes de construirlo, puesto que hay muchos elementos que intervienen, a cual más difícil de cuantificar, como son la experiencia, la organización del trabajo, la habilidad de los desarrolladores, el acierto en la dirección del proyecto, etcétera.

Lógicamente, cuanto mayor tiempo se dedica al desarrollo de un proyecto, más información tendremos sobre él, haciendo que el margen de error de las estimaciones sea menor. Por supuesto, aunque pudiésemos conseguir una estimación perfecta una vez hayamos terminado el proyecto, la principal utilidad de la estimación de esfuerzo es la de poder evaluar a priori, y con una exactitud aceptable, el esfuerzo que va a requerir el desarrollo del proyecto.

Pese a la inexactitud inherente a la estimación de esfuerzo, es de vital importancia para el proyecto realizar dicha valoración de la forma más ajustada posible, ya que nos permitirá reducir costes, así como aumentar los niveles de calidad, algo que puede establecer la diferencia entre obtener beneficios o pérdidas. Para conseguir ese objetivo, es indispensable la experiencia y el tener acceso a una buena información histórica, es decir, conocer datos sobre anteriores proyectos.

Como cabe esperar en un proceso tan complejo, muchas personas han intentado solventar los problemas que entraña la estimación de esfuerzo en la ingeniería del software, ideando para ello un buen número de técnicas con mayor o menor complejidad y éxito. A continuación se presentarán y analizarán brevemente algunas de las técnicas más conocidas e interesantes.

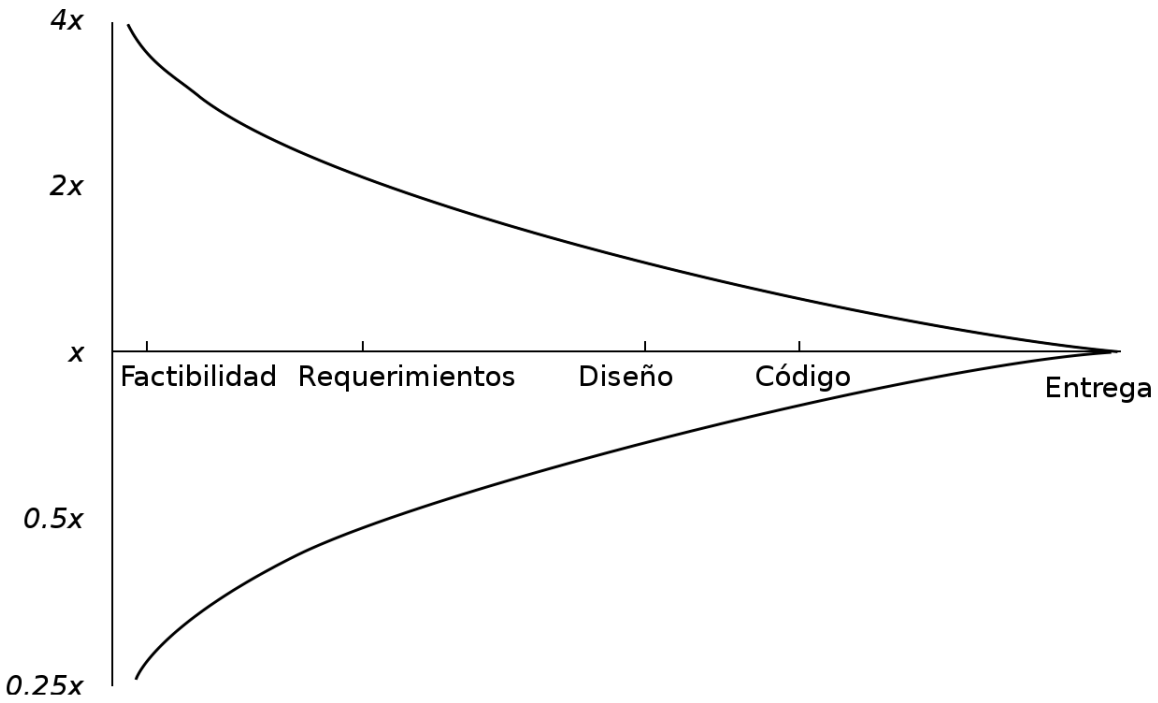


Figura 1.1: Incertidumbre estimada.

1.1.1. Juicio experto

Muchos métodos de estimación se basan en la experiencia de gerentes de proyectos semejantes al que se desea desarrollar. De este modo, si el sistema a desarrollar A es similar al sistema desarrollado B, el esfuerzo requerido para producir A debería ser parejo al de producir B.

Este proceso puede formalizarse solicitando a varios expertos que estimen el esfuerzo de forma pesimista, optimista y más probable. Una vez han sido evaluadas las estimaciones de los expertos, se realiza la media de la distribución beta de probabilidades determinada por esos valores, obteniendo una estimación normalizada. La expresión que define este valor normalizado es la siguiente:

$$\frac{(x + 4y + z)}{6},$$

donde x representa la estimación pesimista, y la optimista y z la más probable.

La técnica Delphi emplea el dictamen de los expertos de una forma más sofisticada. Se solicita a cada experto que efectúe una estimación individual, basada en su experiencia, de forma confidencial. Posteriormente, se calcula el promedio de las esti-

maciones y se muestra su valor a los expertos, dándoles la posibilidad de modificar su anterior valoración. Este proceso se repite hasta llegar a un consenso, siendo todas las estimaciones secretas e individuales durante el mismo.

Los modelos basados en la opinión de expertos dependen directamente de su habilidad para determinar qué proyectos están relacionados y cómo afectan las posibles diferencias existentes entre ellos al esfuerzo necesario. Por tanto, las estimaciones obtenidas empleando estos modelos serán muy subjetivas y posiblemente imprecisas.

1.1.2. Modelo de Bailey–Basili

Bailey y Basili sugirieron en 1987 una técnica de modelado para realizar una ecuación de estimación que reflejase las características de una organización.

Demostraron su técnica utilizando datos de dieciocho proyectos científicos de gran tamaño, obteniendo una ecuación muy precisa:

$$E = 5,5 + 0,73S^{1,16},$$

donde S es la dimensión estimada del sistema en líneas de código.

A su vez, Bailey y Basili explicaron otros factores que afectan al esfuerzo, y que permiten ajustar la ecuación anterior. El proyecto debe ser puntuado entre cero —ausente— y cinco —muy importante— para cada factor indicado en la tabla 1.1, dependiendo de la opinión del gerente del proyecto.

1.1.3. SLIM

Es un método de estimación creado por Lawrence Putnam —de ahí que también sea conocido como método de Putnam—, que muestra la distribución del esfuerzo en relación al tiempo. Es aplicable a proyectos muy grandes —más de setenta mil líneas de código—, aunque es posible ajustar las expresiones para proyectos más pequeños.

SLIM se basa en que la relación entre el esfuerzo requerido y el tiempo necesario para desarrollar un proyecto software no es lineal, sino que es descrita por curvas de Rayleigh. Esto implica que pequeños cambios en el tiempo de desarrollo pueden provocar grandes modificaciones en el esfuerzo requerido.

El elemento central del método de Putnam es la «ecuación del software»:

$$Producto = PP * \left(\frac{Esfuerzo}{B}\right)^{\frac{1}{3}} * Tiempo^{\frac{4}{3}},$$

Metodología total	**Complejidad acumulada**	**Experiencia acumulada**
Diagramas de árbol	Complejidad de la interfaz del cliente	Calificación de los programadores
Diseño descendente	Complejidad de la aplicación	Experiencia de máquina de los programadores
Documentación formal	Complejidad del flujo del programa	Experiencia de los programadores con el lenguaje
Equipos de programadores jefe	Complejidad de la comunicación interna	Experiencia de los programadores con la aplicación
Entrenamiento formal	Complejidad de la base de datos	Experiencia del equipo
Planes de prueba formales	Complejidad de la comunicación externa	
Formalismos de diseño	Cambios al diseño de programas iniciados por el cliente	
Lectura del código		
Carpetas de desarrollo de unidades		

Tabla 1.1: Modificadores del esfuerzo según Bailey-Basili.

donde *Producto* representa cierta medida sobre la funcionalidad del mismo y se mide en líneas de código; PP es el *Parámetro de Productividad*, una constante que engloba los factores del entorno y expresa la productividad de la compañía; el $Esfuerzo$ comprende las distintas etapas por las que pasa el proyecto y se mide en personas por año; B es un parámetro que representa la destreza en función del tamaño del sistema; y el *Tiempo* requerido para completar el desarrollo del software medido en años. Las potencias empleadas en la expresión anterior, fueron obtenidas de forma empírica a mediados de los 80, utilizando datos de 750 sistemas.

La ecuación del software debe estimar el tiempo y el esfuerzo de desarrollo, que son dos incógnitas de la expresión. Además, es necesario conocer el PP de la organización mediante proyectos anteriores y una estimación de las líneas de código del producto.

Para poder estimar el esfuerzo, Putnam propone otra expresión:

$$PIP = \frac{Esfuerzo}{Tiempo^3},$$

donde PIP es un *Parámetro de Incremento de Personal*, valor que se conoce al principio del proceso.

La conclusión que puede extraerse de esta segunda expresión es que el aumento rápido en el personal de desarrollo tiene más efecto en el coste y el esfuerzo que en el tiempo.

Combinando las expresiones anteriores podemos conseguir otra en la que el tiempo no esté presente:

$$Esfuerzo = \left(\frac{Producto}{PP}\right)^{\frac{9}{7}} * PIP^{\frac{4}{7}}$$

1.1.4. Método COCOMO

Uno de los problemas de los métodos vistos en los apartados 1.1.2 y 1.1.3 es su dependencia del tamaño del sistema. Las estimaciones suelen ser necesarias al comienzo, cuando no se dispone de información suficiente para determinar con precisión las dimensiones del sistema. De esta forma, este tipo de modelos terminan intercambiando el problema de estimar el esfuerzo por el problema de estimar el tamaño del sistema. El modelo COCOMO en su versión más reciente reconoce este problema y trata de resolverlo.

El método COCOMO —cuyo nombre es un acrónimo de ***CO**nstructive **CO**st **MO**del* o Modelo Constructivo de Coste— fue propuesto por Barry Boehm en 1981, y es uno

de los modelos de estimación de coste de software más empleados y estudiados. El modelo original ha sido revisado, obteniendo en 1995 una segunda versión del mismo —llamada COCOMO II—, siendo mucho más completa que el modelo original. De ahora en adelante me centraré en esta última versión, al ser la más actual y perfeccionada.

Siendo precisos, COCOMO más que un modelo es una jerarquía de modelos de estimación, es decir, se proponen modelos distintos para las distintas situaciones que atraviesa un proyecto software, ofreciendo así una mayor fidelidad en cada una de ellas. Estos tres modelos son:

Modelo de composición de la aplicación

Se dirige a las primeras etapas de desarrollo, cuando no se conocen con detalle los requisitos de la aplicación y se desarrollan prototipos, a partir de componentes ya existentes, de la interfaz de usuario, la interacción del sistema, etcétera. Como se puede intuir, en estas etapas tan tempranas de desarrollo no se tiene información sobre las dimensiones que tendrá el sistema, por lo que COCOMO II estima el mismo en función de generadores de esfuerzo de alto nivel, como el número de pantallas e informes, que se conocen con el nombre de puntos de aplicación.

Modelo de diseño anticipado

Es empleado cuando los requisitos se estabilizan y se tiene una idea —aunque puede que no sea definitiva— de la arquitectura del software. En este momento se conoce más información para realizar una estimación que en el punto anterior, aunque aún no es suficiente para realizar una valoración precisa. Por este motivo, en lugar de usar puntos de aplicación o líneas de código como medida de tamaño, se emplean puntos de función. Los puntos de función son una técnica que estima la funcionalidad capturada en los requisitos ofreciendo una descripción del sistema más rica que los puntos de aplicación.

Modelo *postarquitectonico*

Este modelo se orienta al momento en el que el desarrollo comienza y se tiene mucha más información que permite realizar una estimación razonablemente precisa del

tamaño del software. En este modelo es posible dimensionar el sistema mediante puntos de función o líneas de código, pudiendo incluirse factores de costo que reflejan la capacidad del personal, el producto y las características del proyecto.

1.1.5. Método de caracterización de la actividad del desarrollador

El método de caracterización de la actividad del desarrollador es fruto del estudio de numerosos proyectos de software libre realizado por tres miembros de esta universidad: Juan José Amor, Gregorio Robles y Jesús M. González–Barahona. Es un método muy reciente, para ser precisos data del año 2006, y surge como una alternativa a los métodos de estimación de esfuerzo tradicionales, con el objetivo de adaptarse mejor a los proyectos de software libre, aunque no es exclusivo para ellos.

En lugar de basar la estimación en el número de líneas de código, como hacen la mayor parte de los métodos de estimación de esfuerzo existentes, sus autores piensan que es posible mejorar la estimación de costes considerando otras métricas basadas en la caracterización de las actividades realizadas por los desarrolladores, es decir, analizando su comportamiento durante el proceso de desarrollo.

Hay que tener en cuenta que en el desarrollo de software «tradicional», existe un grupo fijo de desarrolladores que tienen un horario muy ajustado y deben dedicar un determinado número de horas semanales a sus tareas. Sin embargo, esto no ocurre así en el desarrollo de software libre, ya que es habitual que colaboradores externos al proyecto intervengan en el desarrollo del mismo y dediquen un número indeterminable de horas a esa labor. Los sistemas de estimación de esfuerzo tradicionales fallan en este punto, porque no tienen en cuenta la intervención de esos colaboradores externos y, aunque esas personas no formen parte del equipo de desarrollo, su participación también influye en el esfuerzo final requerido para completar el software.

El método de caracterización de la actividad del desarrollador propone evaluar el coste de un proyecto con la siguiente expresión:

$$coste = \sum coste_{interno} + \sum coste_{desarrolladores\_externos}$$

Asimismo, establece que el coste es función del esfuerzo requerido, y este es, a su vez, función de las actividades realizadas por el desarrollador. Por tanto, es posible estimar el esfuerzo de producción del software a partir de los datos producidos en un proyecto libre —que son detallados en el apartado 1.4.2—.

1.2. Administración del tiempo

El tiempo es un recurso muy valioso que no podemos almacenar, detener o adquirir. Por este motivo, habitualmente nos preocupamos por el tiempo que dedicamos a realizar las tareas que llevamos a cabo —ya sea en el mundo laboral o en el personal—, e intentamos organizarlas y gestionarlas para conseguir emplear mejor el tiempo. Lo más frecuente es que cada uno se organice «a su manera»: establecer un horario indicando y separando distintas actividades, planificar las tareas a realizar y anotarlas en una lista, etcétera. Puesto que estos métodos no siempre son efectivos, han surgido distintas técnicas que prometen organizar mejor nuestra forma de realizar actividades, con el fin de poder utilizar el tiempo de una forma más eficaz. Entre estas técnicas destacan: Getting Things Done[1], los siete hábitos de Stephen Covey[2], Zen To Done[3] y Pomodoro[4].

Al pasar cada vez una mayor cantidad de tiempo usando ordenadores, es interesante conocer cuanto empleamos en las distintas tareas que realizamos frente ellos. Esto tiene una especial relevancia cuando la persona se dedica al desarrollo de software libre, ya que en muchos casos es desarrollado por personas en su tiempo libre.

1.3. Estadísticas de uso de aplicaciones

La necesidad de saber cuáles son los productos o servicios más usados en cualquier campo hace que se efectúen estadísticas con el objetivo de conocer cuál es su impacto frente a la competencia. Algunos ejemplos de estos estudios podemos observarlos en el cálculo y comparación de la audiencia televisiva, del número de abonados en las compañías de telefonía móvil, de la difusión de la prensa, etcétera.

El caso del software no es una excepción, y son datos que interesan a muchas partes:

- Las empresas desean conocer estadísticas de uso de sus aplicaciones para poder compararse con otras, con el objetivo de convencer a más personas de usar sus productos, conseguir mejores contratos y observar cómo evolucionan los intereses de los usuarios. Por poner un ejemplo, en dos de los sectores con mayor com-

[1] http://es.wikipedia.org/wiki/Getting_Things_Done

[2] http://en.wikipedia.org/wiki/The_Seven_Habits_of_Highly_Effective_People

[3] http://zenhabits.net/2007/04/zen-to-done-ztd-the-ultimate-simple-productivity-system

[4] http://www.pomodorotechnique.com

petencia dentro de la industria del software, como son los navegadores web y los sistemas operativos, se producen estudios de cuotas de mercado frecuentemente.

- A los desarrolladores independientes —por ejemplo, de software libre— también les resulta interesante conocer estas estadísticas, ya que pueden ayudarles a conseguir más patrocinadores, a incrementar sus comunidades de usuarios y a implementar características relevantes que tengan otros programas similares con un mayor número de usuarios.

- Estas estadísticas también resultan de interés a los desarrolladores de sistemas operativos, debido a que pueden incluir aplicaciones o implementar funcionalidades que cubran las necesidades más demandadas por los usuarios. Puede observarse como ejemplo la forma en la que las distintas distribuciones de GNU/-Linux seleccionan los programas que componen su sistema por defecto. Aunque la madurez del software incorporado y las opiniones de sus desarrolladores principales juegan un papel fundamental en la selección de software de una distribución, muchas han creado sistemas de recolección de datos de programas instalados para conocer —con el previo permiso del usuario— cuáles son los programas más instalados, y poder tenerlos en cuenta a la hora de confeccionar el software de la distribución.

- De forma similar al punto anterior, los administradores de sistemas desean conocer las aplicaciones usadas en los sistemas bajo su responsabilidad con el fin de gestionar el software instalado en los mismos.

- Los propios usuarios también están interesados en conocer qué aplicaciones utilizan otras personas, con diversos objetivos, aunque destacando el conocer nuevas —y puede que mejores— aplicaciones.

Pese a que, como podemos observar, estas medidas resultan bastante interesantes, en el caso del software es bastante complejo conocer las aplicaciones que utiliza cada persona, principalmente a causa de las numerosas plataformas que conviven, la enorme diversidad de aplicaciones existentes y sus formas de distribución. Por ello, se realizan distintas estadísticas en función de diversos factores: medir las descargas efectuadas de los programas, contabilizar las licencias vendidas, evaluar el número de

visitas a determinadas páginas web, o estudiar las búsquedas realizadas en Internet entre otros.

1.4. Software libre

Para aclarar el concepto de software libre emplearé la definición realizada por la *Free Software Foundation*[5] [11]:

> *El «Software Libre» es un asunto de libertad, no de precio. Para entender el concepto, debe pensarse en «libre» como en «libertad de expresión», no como en «cerveza gratis».*
>
> *«Software Libre» se refiere a la libertad de los usuarios para ejecutar, copiar, distribuir, estudiar, cambiar y mejorar el software.*

Según esta misma organización, podemos distinguir un programa libre de otro privativo por cumplir estas cuatro libertades:

Libertad 0. La libertad de usar el programa, con cualquier propósito.

Libertad 1. La libertad de estudiar el funcionamiento del programa y adaptarlo a nuestras necesidades.

Libertad 2. La libertad de distribuir copias, con lo que se puede ayudar a otros.

Libertad 3. La libertad de mejorar el programa y hacer públicas las mejoras, de modo que toda la comunidad se beneficie.

Para que puedan cumplirse las libertades 1 y 3 es indispensable que el código fuente sea accesible.

1.4.1. Desarrollo de software libre

En la actualidad, existe un gran número de proyectos de software libre. Muchos de ellos son impulsados y mantenidos por empresas, aunque la mayoría son desarrollados por la comunidad del software libre. Esta comunidad no es más que un colectivo de desarrolladores y usuarios distribuidos geográficamente que apoyan el software libre, y que suelen colaborar en proyectos libres usando herramientas a través de Internet.

[5] *Free Software Foundation* es una organización creada con el objetivo de difundir y apoyar el movimiento del software libre. Su sitio web es `http://www.fsf.org`.

Puesto que todo el proceso de desarrollo se suele realizar a través de la Red, cada proyecto genera una gran cantidad de información que puede ser consultada por cualquiera. Entre esa información destacan las decisiones tomadas durante el desarrollo, el código fuente de la aplicación, y la información sobre los defectos que se encuentran en la misma.

Como vimos en el punto 1.1, las técnicas de estimación de esfuerzo requieren gran cantidad de información sobre el sistema en desarrollo y el entorno para poder ofrecer unas estimaciones precisas. En el modelo de software privativo suelen emplearse para obtener estos datos otros proyectos realizados en la propia empresa, al ser los únicos de los que se tiene suficiente información. Esto contrasta con la gran cantidad de proyectos libres que se pueden encontrar, haciendo que sea más sencillo dar con proyectos —o parte de los mismos— que encajen mejor con el sistema a desarrollar. Por si esto fuera poco, dispondremos además de la información generada día a día durante su desarrollo, facilitando el estudio de la aplicación.

1.4.2. Herramientas para manejar la información generada

Debido a la gran cantidad de información que se maneja en un proyecto de software libre, la necesidad de que esta información se encuentre disponible en línea, y las distintas procedencias de la misma, se han creado varias herramientas capaces de facilitar la colaboración e interacción entre los miembros de la comunidad. Algunas de estas herramientas son:

Listas de correo

Como el software libre es desarrollado generalmente por personas que se encuentran distribuidas geográficamente, es vital disponer de una herramienta de comunicación para debatir y tomar decisiones sobre el proyecto.

Las listas de correo juegan un papel fundamental en este aspecto, puesto que permiten enviar mensajes de correo electrónico a todos sus componentes, sin importar si se encuentran en línea en ese momento o el número de personas suscritas a la lista.

Generalmente, estas listas suelen ser públicas y se puede acceder a ellas vía web, por lo que toda la toma de decisiones del proyecto puede ser consultada y analizada por cualquiera que lo desee.

Sistemas de control de versiones

Básicamente, sirven para almacenar el código fuente de la aplicación, aunque tienen muchas más funcionalidades que facilitan el desarrollo de software en grupo. Algunas de estas características son: solucionar problemas de concurrencia cuando varias personas trabajan a la vez en el proyecto; simplificar el control de versiones de los distintos ficheros que componen la aplicación, conociendo información sobre la fecha del cambio, su autoría y los cambios realizados; permitir la creación de varias «ramas de desarrollo», pudiendo existir una versión estable y otra experimental de la misma aplicación con las que se puede trabajar independientemente; y facilitar la vuelta atrás en el código, pudiendo restaurar una versión anterior de un fichero —o varios— si se desea revertir algún cambio.

Sistemas de seguimiento de errores

Los sistemas de seguimiento de errores tienen como principal objetivo recopilar informes sobre defectos del software con el fin de ser solucionados en el futuro.

Cuando estas aplicaciones son usadas en el desarrollo de software libre, suelen ser los propios usuarios de la aplicación los que informan de los errores. Este hecho potencia el concepto de comunidad existente, ya que los desarrolladores se benefician de los reportes de errores de los usuarios, y estos ven que su aportación es tenida en cuenta en el desarrollo de la aplicación.

Normalmente, es posible establecer el grado de relevancia del problema indicado, pudiendo priorizar los mismos en función de su importancia, así como seguir el estado en el que se encuentra la solución al problema.

Además de estos usos, también suelen emplearse este tipo de sistemas para proponer nuevas funcionalidades a los desarrolladores y para gestionar las tareas pendientes de los mismos.

Como es habitual, existen varios sistemas de seguimiento de errores con filosofías diferentes, aunque lo más importante es que consiguen hacer que todo el proceso de comunicación y corrección de errores sea transparente y que cualquiera pueda participar en él, ya sea reportando problemas o incluso proponiendo soluciones a los mismos.

Wikis

En los últimos tiempos están tomando una gran relevancia a la hora de aportar documentación para las aplicaciones.

Una *wiki* es un mecanismo de colaboración consistente en un sitio web cuyo contenido puede ser editado a través de un navegador web. De esta forma, cualquiera —generalmente previo registro— puede aportar sus conocimientos sobre algún aspecto del proyecto libre, construyendo un sitio web con gran cantidad de documentación de forma *colaborativa*, siendo además fácilmente editable y muy dinámico.

Creo que todos en la actualidad conocemos Wikipedia[6], que no es más que una enorme *wiki* empleado como enciclopedia.

1.5. Aplicaciones y redes sociales

Con la popularización de las conexiones a Internet surgieron una serie de aplicaciones que permiten a los usuarios comunicarse a través de la red.

Las aplicaciones software sociales buscan mantener las conexiones entre usuarios, facilitando mecanismos para conversar y compartir contenidos. Para ello, incluyen herramientas de comunicación —que capturan, almacenan y presentan la información transmitida—, y de interacción —que manejan las interacciones entre usuarios o grupos que se comunican—.

Aunque existen numerosos tipos de aplicaciones sociales, desde aplicaciones de mensajería hasta mundos virtuales, en los últimos tiempos las redes sociales son la máxima representación de estas aplicaciones.

Una red social es un servicio o plataforma en línea, que se enfoca en construir y reflejar relaciones sociales entre personas que comparten intereses o actividades. Pese a los problemas de privacidad que derivan de su uso, estas redes han conseguido en muy poco tiempo un gran número de usuarios, así como revolucionar la forma de comunicarse con otras personas.

[6]Wikipedia es un proyecto de la Fundación Wikimedia para construir una enciclopedia libre y políglota. `http://www.wikipedia.org/`

Capítulo 2

Estado del arte

En este capítulo se desarrollará la idea introducida en el capítulo anterior, describiendo el problema que se pretende resolver realizando este proyecto. Asimismo, se repasarán algunas de las ideas y respuestas que han surgido para intentar solventar ese problema.

Posteriormente, se realizará una pequeña revisión de buena parte de las tecnologías empleadas para llevar a cabo el desarrollo del presente trabajo.

2.1. Descripción del problema

Como se comentó en la introducción, el problema fundamental que pretende resolver el presente proyecto es conocer, de una manera efectiva, qué tareas realizamos mientras empleamos un ordenador. A continuación, vamos a analizar las dificultades que presentan varios de los puntos tratados en el capítulo 1, observando que, la solución al problema planteado permitiría solventar algunas de esas dificultades.

En primer lugar, analizando el escenario de la estimación de esfuerzo, hemos observado que muchos métodos de estimación de costes necesitan disponer de datos históricos fiables para conseguir una mayor precisión. Estos datos pueden obtenerse de estimaciones realizadas con anterioridad en proyectos de la misma empresa, así como de proyectos de software libre que compartan ciertas funcionalidades o similitudes con el proyecto a realizar. No obstante, al no existir dos proyectos de software iguales, en ocasiones eso no resulta suficiente o, al menos, podría ser mejorable. Lo ideal sería conocer en detalle las distintas tareas llevadas a cabo durante el desarrollo del proyecto, así como el tiempo dedicado a cada una de ellas. Sin duda, si los proyectos utilizados como base para la estimación emplearon herramientas como las descritas en el pun-

to 1.4.2, podríamos disponer de muchos más datos relativos al proceso de desarrollo pero, aún así, si basamos nuestras observaciones en proyectos de software libre, seguimos sin saber la dedicación que han tenido los distintos desarrolladores, hecho que se explica en el apartado 1.1.5. En este punto, puede observarse como, si consiguiésemos la información obtenida al resolver el problema planteado, sería posible conocer qué tareas realizaron las personas involucradas en el desarrollo de la aplicación, junto al tiempo que invirtieron en esas tareas, logrando una información mucho más completa.

Continuemos analizando el marco de la administración del tiempo, donde podemos ver que es complicado conocer con exactitud las distintas tareas que realizamos cuando empleamos un ordenador. Esto se debe, en buena medida, a la propia experiencia de uso de los computadores, que permiten realizar distintas acciones paralelamente, incluso algunas que no solicitamos explícitamente. De esta manera, en muchas ocasiones, los eventos que suceden cuando empleamos el ordenador nos ayudan a perder la atención y terminamos dedicando más tiempo del deseado a: leer el correo electrónico, realizar una actualización del sistema o hablar con un amigo que nos ha escrito un mensaje de *chat* justo cuando estábamos centrados en otra cuestión. Así, una vez hemos concluido nuestro periodo de uso del computador, probablemente no recordemos algunas de las tareas que hemos llevado a cabo, y será aún más complicado determinar cuánto tiempo hemos empleado en cada una de ellas. Nuevamente, la solución al problema planteado al inicio de este apartado ofrecería datos para percibir la cantidad de tiempo dedicado a cada cometido, haciendo posible evaluar si podríamos ser más productivos.

Finalmente, centrándonos en el contexto de las estadísticas de uso de aplicaciones, es posible advertir que los distintos tipos de medidas realizados para contabilizar el número de usuarios no son consistentes entre sí. Por ejemplo, no tiene sentido comparar el número de licencias vendidas de Windows 7 con el número de descargas de Fedora para extrapolar el número de usuarios de cada sistema. De nuevo, una herramienta que permitiese determinar las aplicaciones empleadas mientras se utiliza el ordenador, podría ofrecer datos relevantes para conocer de manera bastante precisa la cuota de uso de las distintas aplicaciones.

De este modo, hemos comprobado como una herramienta que «observe» las tareas que realiza el usuario frente al ordenador y estime el tiempo dedicado a cada una de

ellas podría resolver todos estos problemas, o al menos mejorar su situación actual. Además, para determinados usos, sería interesante centralizar todos estos datos en un servidor, de manera que puedan realizarse cálculos más complejos con la información, así como obtener estadísticas de una manera más general. En realidad, este es el objetivo marcado por el sistema DeTraS, cuyo desarrollo está en progreso, y que se describirá en la sección 2.4.

Para concluir este punto, me gustaría indicar que, aunque existen varios escenarios en los que puede ser útil DeTraS, en adelante me centraré especialmente en el escenario de seguimiento de actividades de desarrollo.

2.2. Encuestas

Una forma de conocer las actividades que realizan los desarrolladores es tan sencilla como realizar encuestas periódicamente, solicitando datos sobre las tareas desarrolladas y el tiempo que han requerido. Estas encuestas se pueden llevar a cabo de forma telefónica, en papel o empleando alguna aplicación informática.

Evidentemente, este método es muy impreciso, ya que depende de la atención y capacidad de memorización del desarrollador. Así, si se realizase la encuesta cada poco tiempo —imaginemos que se efectuase una diaria—, sería muy tedioso y molesto para los desarrolladores y, probablemente, la responderían sin prestarle demasiada atención. Por contra, si se realizase la encuesta más esporádicamente —pongamos una vez al mes— muchos detalles habrían caído en el olvido, y el recuerdo del tiempo necesitado para completar las actividades sería demasiado vago.

Además, hay que tener en cuenta que algunos de los métodos que suelen emplearse para realizar encuestas pueden ser intrusos y molestos. Por este motivo deberían descartarse las encuestas telefónicas, que son sin duda las más molestas, ya que además de interrumpir al desarrollador mientras está trabajando, le obligan a contestar a las preguntas en ese preciso momento. También debería evitarse emplear encuestas automáticas que utilicen ventanas emergentes para mostrar la encuesta o avisar de que está disponible para su compleción.

2.3. Proyecto Hamster

Hamster es un *applet*[1] de seguimiento temporal desarrollado para GNU/Linux, que forma parte del entorno de escritorio GNOME[2]. Su principal funcionalidad es permitir conocer las tareas que se realizan y cuanto tiempo requieren. La aplicación, a su vez, posibilita la agrupación de actividades en categorías, facilitando el posterior análisis de la información. Además, es capaz de mostrar todos los datos recogidos empleando gráficos diarios, semanales o mensuales, de modo que sea posible analizar la información de una forma más visual.

Esta aplicación puede cumplir en parte con los requisitos para realizar un seguimiento de las actividades que efectúa un desarrollador, aunque tiene una carencia importante: todo su funcionamiento se basa en introducir manualmente la tarea que se va a realizar en el momento preciso en que se comienza a ejecutar. Esto supone un problema cuando se está trabajando debido a que el foco de atención de la persona —en este caso del desarrollador— está centrado principalmente en la tarea que debe realizar, dejando en un segundo plano otras actividades. Por tanto, es probable que el desarrollador no recuerde que debía cambiar la tarea en la aplicación, alterando los datos reales. Asimismo, en el caso de estar llevando a cabo dos tareas de forma simultánea, o actividades que requieren de poco tiempo, puede llegar a ser molesto el tener que editar cada poco tiempo la ocupación actual.

2.4. El sistema DeTraS

El sistema DeTraS —acrónimo de ***De**veloper **Tra**cking **S**ystem* o Sistema de Seguimiento de Desarrolladores— fue creado por Carlos García Campos y entregado como Proyecto Fin de Carrera en el curso 2005/2006. Posteriormente, en el curso 2009/2010, el sistema fue objeto de una serie de ampliaciones y mejoras llevadas a cabo por mí mismo y que fueron presentadas en mi Proyecto Fin de Carrera. En esencia, la idea con la que este proyecto pretende resolver el problema expuesto con anterioridad consiste, según su autor [1], en:

> *... capturar todo lo que el desarrollador está haciendo en su ordenador, para posteriormente analizarlo y estudiarlo.*

[1] En el contexto de los entornos de escritorio, un *applet* es una pequeña aplicación cuya interfaz de usuario reside en un panel del mismo.

[2] `http://www.gnome.org/`

Para llevar a la práctica esta idea, el sistema se encarga de «observar» las acciones que se van realizando en el ordenador, registrando las distintas aplicaciones que usa el desarrollador y el tiempo que se emplea cada una de ellas. Además, DeTraS incluye, junto a esa información, datos sobre el inicio o cierre de la sesión del usuario, así como de los periodos de inactividad —entendiendo que la sesión de usuario está inactiva cuando lleva más de cinco minutos sin que se haya empleado el teclado o el ratón, y/o cuando el salvapantallas está en funcionamiento—. Todo ello constituye una fuente de información muy valiosa a la hora de analizar el esfuerzo del desarrollador en las distintas etapas del desarrollo software.

No obstante, las tareas del sistema DeTraS no se limitan a reunir la información indicada previamente, sino que permite, a su vez: enviar la información acumulada durante el uso del programa a un servidor, con el fin de poder estudiar los datos de uno o varios usuarios detalladamente; proporciona a los usuarios varios niveles de filtrado de la información enviada a los servidores, con el fin de conservar su privacidad; provee un portal web para observar de forma sencilla la información recogida en el servidor, ofreciendo estadísticas en base a la misma; y facilita la generación de informes con los datos almacenados en el servidor, suministrando una serie de *scripts* genéricos y una biblioteca de programación que simplifica la creación de nuevos *scripts*.

Como se puede advertir, partiendo de la idea expuesta con antelación, es posible resolver muchos de los problemas existentes en otros métodos de cálculo de esfuerzo, haciendo este proceso transparente para el desarrollador y bastante exacto.

Pese a que el sistema DeTraS es funcional y bastante interesante en su estado actual, tiene algunos aspectos que deben ser mejorados o ampliados para intentar aumentar su uso y difusión, algo que nos permitiría realizar estudios valorando el esfuerzo de desarrolladores de todo el mundo, al mismo tiempo que la comunidad de software libre se implica en la evolución del sistema. Esos son los objetivos del presente trabajo, que partirá de la base aquí detallada.

2.5. Herramientas y tecnologías utilizadas

Como puede imaginar el lector, las herramientas y tecnologías informáticas empleadas para llevar a cabo el desarrollo del trabajo aquí expuesto son muy variadas y numerosas. En este apartado se describirán brevemente las más importantes que se

han utilizado, así como su relación con el trabajo. Nótese que algunos de los objetivos y componentes del software desarrollado que se indican en esta sección serán introducidos en el apartado 4.2.

2.5.1. C

C [14, 15] es un lenguaje de programación creado en 1972 por Dennis M. Ritchie en los Laboratorios Bell. Se trata de un lenguaje de propósito general con una sintaxis sencilla de aprender, estructuras de datos simples pero muy versátiles y un rico conjunto de operadores.

Este lenguaje de programación está a medio camino entre los lenguajes de alto nivel y los de bajo nivel, ya que tiene una estructura y características típicas de los lenguajes de alto nivel, permitiendo a su vez realizar acciones a muy bajo nivel.

A lo largo de su vida, C ha sufrido varios procesos de estandarización —en 1989 por el organismo ANSI y en 1990 por el ISO— que permiten conseguir un código independiente de la máquina si no se emplean elementos dependientes del compilador o el sistema operativo.

Pese a no tratarse de un lenguaje de alto nivel estrictamente hablando, C tiene varias ventajas frente a estos lenguajes, erigiéndose como una muy buena opción a la hora de desarrollar programas: es muy eficiente y pueden emplearse características de bajo nivel para optimizar la implementación; es posible emplearlo en prácticamente todos los sistemas; y permite realizar programas modulares y emplear bibliotecas de programación existentes.

La cercanía de C con el nivel de la máquina también acarrea mayor complejidad en algunas ocasiones, como a la hora de gestionar la memoria, e incluso problemas de seguridad si no se trabaja con cuidado, como el famoso y temido *buffer overflow*[3].

Este lenguaje de programación es usado, junto a las bibliotecas GLib y GObject, para el desarrollo de TempusFugit.

2.5.2. GLib y GObject

Aunque C es un buen lenguaje para programar, tiene algunas carencias: la biblioteca estándar no es tan rica como las de otros lenguajes y no es orientado a objetos. Las

[3]El error *buffer overflow* —o desbordamiento de *buffer*— ocurre cuando se copian en un área de memoria más datos de los que puede contener, existiendo la posibilidad de sobreescribir otros datos existentes.

bibliotecas GLib [16] y GObject [17] forman parte de las herramientas de desarrollo de GNOME y surgen con la idea de paliar esas lagunas.

GLib es una biblioteca de utilidades de propósito general que proporciona tipos de datos, macros, conversiones de tipos, utilidades para cadenas de caracteres y ficheros entre otras.

GObject hace realidad el desarrollo orientado a objetos en C, ofreciendo un sistema de tipos y otras características de los lenguajes orientados a objetos. Además, está diseñado para trabajar con otros lenguajes a través de *bindings*[4].

Cabe destacar que ambas bibliotecas no son dependientes del resto del entorno de escritorio GNOME y que están disponibles en varias plataformas, por lo que su empleo no impide que el código desarrollado sea independiente de la máquina.

El desarrollo de TempusFugit es realizado sobre la base que proporcionan C junto a estas dos bibliotecas.

2.5.3. XML

XML [19], siglas en inglés de Extensible Markup Language (lenguaje de marcas extensible), es un metalenguaje extensible de etiquetas desarrollado por el World Wide Web Consortium (W3C).

No se trata de un lenguaje en particular, sino una manera de definir lenguajes para diferentes necesidades, proponiéndose como un estándar para el intercambio de información estructurada entre diferentes plataformas.

DeTraS emplea documentos XML para enviar, usando el protocolo SOAP, los datos almacenados en el ordenador del usuario al servidor, donde son tratados y almacenados en una base de datos.

2.5.4. Python

Python [20, 21] es un lenguaje de programación creado por Guido van Rossum a principios de los años 90 cuyo nombre está inspirado en el grupo de cómicos ingleses Monty Python. Se trata de un lenguaje interpretado, con *tipado* dinámico, fuertemente *tipado* y orientado a objetos. Posee una sintaxis muy limpia y que favorece un código legible.

[4]Un *binding* es una adaptación de una biblioteca de programación a otro lenguaje.

El intérprete de Python está disponible en multitud de plataformas (UNIX, Solaris, Linux, DOS, Windows, OS/2, Mac OS, etc.) por lo que si no utilizamos librerías específicas de cada plataforma nuestro programa podrá correr en todos estos sistemas sin grandes cambios. Además, todo el proyecto es gratuito y de código abierto, con todas las ventajas que ello conlleva.

Este lenguaje de programación se emplea en múltiples componentes de DeTraS: el *applet* para GNOME, el programa Squealer, el servicio web, la generación de informes, la aplicación web Dazer, así como en la aplicación social CheckApp.

2.5.5. Libnotify

Libnotify [18] es una biblioteca que permite mostrar notificaciones de escritorio de forma amigable, sin entrometerse en las actividades del usuario. Tiene soporte para GTK+ y Qt, que son las bibliotecas para desarrollo de interfaces más empleadas en el mundo GNU/Linux.

Esta biblioteca ha sido empleada para mejorar la experiencia de usuario con el *applet* de DeTraS para GNOME.

2.5.6. PyCha

PyCha [33] —acrónimo de ***Py**thon **Cha**rts*—, es un paquete de Python destinado a dibujar gráficas empleando la biblioteca Cairo. Pretende ofrecer gráficas con una apariencia visual agradable y fáciles de crear, a costa de limitar el catálogo de gráficas proporcionado.

Las gráficas locales del *applet* de DeTraS se han llevado a cabo empleando este paquete.

2.5.7. Django

Django [22] es un *framework* de código libre para el desarrollo de aplicaciones web, escrito en Python, que sigue el patrón arquitectónico Modelo–Vista–Controlador.

El principal objetivo de Django es permitir la creación de sitios web complejos de forma sencilla, incentivando la reusabilidad y la conexión entre componentes.

Las aplicaciones web Dazer y CheckApp han sido construidas empleando el entorno y las utilidades que Django provee.

2.5.8. HTML

HTML [23], siglas de *HyperText Markup Language* (Lenguaje de Marcado de Hipertexto), es el lenguaje de marcado predominante para la elaboración de páginas web. Es usado para describir la estructura y el contenido en forma de texto, así como para complementar el texto con objetos tales como imágenes. HTML se escribe en forma de «etiquetas», rodeadas por corchetes angulares (<,>). HTML también puede describir, hasta cierto punto, la apariencia de un documento, y puede incluir *scripts* (por ejemplo Javascript) que pueden influir en el comportamiento de navegadores web y otros procesadores de HTML.

Este lenguaje de marcado se emplea para mostrar los contenidos de Dazer y CheckApp en los navegadores web.

2.5.9. CSS

Las hojas de estilo en cascada [24] (*Cascading Style Sheets* o CSS) son un lenguaje empleado para la presentación de documentos escritos en un lenguaje de marcado, es decir, nos permite modificar el formato y la apariencia del documento. Su aplicación más usual es formatear páginas web escritas en HTML y XHTML, aunque el lenguaje también puede aplicarse a cualquier tipo de documento XML.

Dentro del ámbito del presente trabajo, las hojas de estilo CSS han sido empleadas para dar la apariencia actual a la aplicación social desarrollada.

2.5.10. SQLite

SQLite [26, 27] es una biblioteca software que implementa un sistema de gestión de bases de datos relacional (SGBDR). A diferencia de otros SGBDR, SQLite no es un proceso independiente con el que el programa principal se comunica. En su lugar, la biblioteca SQLite se enlaza con el programa y es empleada a través de llamadas a la misma.

El conjunto de la base de datos es almacenado como un sólo fichero en la máquina donde se ejecuta el programa, reduciendo la latencia de las llamadas a la base de datos.

DeTraS emplea SQLite por defecto para gestionar las bases de datos que contienen la información obtenida en los ordenadores de los clientes y en el servidor remoto. No obstante, cabe destacar que el sistema está preparado para poder emplear otro SGBDR

si fuese necesario.

2.5.11. MySQL

MySQL [34] es un SGBDR libre que se ejecuta como un servidor, ofreciendo acceso a varios usuarios a múltiples bases de datos. Es empleado en multitud de proyectos, tanto de código libre como propietarios, que necesitan una base de datos completa, como Joomla, Wordpress, Wikipedia, Google y Facebook.

La aplicación social CheckApp, hace uso de MySQL para almacenar la información necesaria para su funcionamiento, ya que es un SGBDR mucho más completo, versátil, rápido y fiable que SQLite, especialmente cuando se realizan muchas transacciones.

2.5.12. Glade, GTK+ y PyGtk

GTK+ [28] o *The GIMP Toolkit* es un conjunto de bibliotecas multi–plataforma para desarrollar interfaces gráficas de usuario, principalmente para los entornos gráficos GNOME, XFCE y ROX aunque también se puede usar en el escritorio de Windows, MacOS y otros. Inicialmente fueron creadas para desarrollar el programa de edición de imagen GIMP, sin embargo actualmente se usan bastante por muchos otros programas en los sistemas GNU/Linux. Junto a Qt es una de las bibliotecas más populares para X Window System.

PyGTK [29] es un *binding* en Python para la biblioteca gráfica GTK+. De este modo, es posible desarrollar interfaces gráficas potentes empleando GTK+ en Python.

Glade [30] es una herramienta de desarrollo visual de interfaces gráficas mediante GTK/GNOME. Es independiente del lenguaje de programación y genera un archivo XML que puede emplearse para construir la interfaz gráfica de usuario en tiempo de ejecución mediante bibliotecas.

Estos tres elementos son claves en el desarrollo del *applet* para el escritorio GNOME de DeTraS. Como puede adivinar, la interfaz gráfica fue diseñada con Glade, implementando su comportamiento a través de PyGTK.

2.5.13. Otras

Además de las herramientas y tecnologías detalladas con anterioridad, me gustaría destacar otras que no intervienen directamente en la construcción de los componentes

del trabajo presentado, pero que han sido claves para facilitar el desarrollo del mismo.

Bazaar

Bazaar [31] es un sistema de control de versiones distribuido patrocinado por Canonical Ltd., diseñado para facilitar la contribución en proyectos de software libre y código abierto.

Este sistema de control de versiones ha sido empleado durante todo el desarrollo del código de los componentes de este trabajo.

Launchpad

Launchpad [32] es una aplicación web para el soporte del desarrollo software, especialmente de software libre, que ofrece los siguientes servicios: alojamiento de código, sistema de seguimiento de errores, sistema de seguimiento de especificaciones y nuevas características, una base de conocimiento de la comunidad, facilidades para traducir aplicaciones, y una herramienta para mantener repositorios de paquetes.

A lo largo del desarrollo del trabajo, he venido empleando varias de las herramientas existentes en Launchpad, puesto que permiten centralizar el desarrollo de DeTraS y CheckApp en un único lugar, que está disponible para mí y para otros miembros que forman —y formarán— parte de la comunidad creada en torno al mismo.

SourceForge

SourceForge [35] es una aplicación web realizada para ofrecer soporte al desarrollo de software libre, ofreciendo entre otros: alojamiento de código, sistema de seguimiento de errores, almacenamiento de descargas, listas de correo, *wikis*, alojamiento web, bitácora y base de datos. En la actualidad es empleado para alojar un gran número de proyectos.

Durante el desarrollo del trabajo se han empleado diversas herramientas ofrecidas por SourceForge para complementar a aquellas ofrecidas por Launchpad, especialmente las *wikis* y las listas de correo.

Apache

El servidor web Apache [36] es el más usado en la actualidad con mucha diferencia. Es un servidor web de código libre, que ofrece una gran funcionalidad y rendimiento,

además de permitir una amplia configuración.

Este servidor es empleado para servir la aplicación CheckApp, alojada en un servidor facilitado por el GSyC.

WordPress

WordPress [37] es una herramienta de publicación de bitácoras libre, desarrollada sobre PHP y MySQL. A su vez, WordPress también cuenta con una plataforma que ofrece alojamiento de bitácoras a los usuarios que lo deseen.

Buena parte de la difusión de DeTraS y CheckApp se ha llevado a cabo a través de una bitácora creada en esta plataforma.

Freshmeat

Freshmeat [38] mantiene el mayor índice de aplicaciones software para Unix y multi–plataforma. Está especialmente orientado a proyectos de software libre, y permite a sus usuarios estar al tanto de las últimas novedades de sus aplicaciones favoritas.

Este sitio web también ha sido clave en la difusión de DeTraS y CheckApp.

Herramientas para compilar y construir la aplicación

autogen y automake. Facilitan la configuración, compilación, instalación y limpieza de los componentes del trabajo.

pbuilder. Es una herramienta automática para la construcción de paquetes Debian. Permite realizar el proceso en un entorno chroot básico, garantizando que la definición del paquete Debian se ha realizado correctamente.

Capítulo 3

Objetivos

3.1. Propósito del trabajo

El presente trabajo tiene dos propósitos principales, que se describirán brevemente en este apartado.

Por un lado, se pretende ampliar, mejorar y dar visibilidad al sistema DeTraS descrito en el apartado 2.4. Por ello, el trabajo aquí descrito comparte el mismo propósito que el del sistema DeTraS: obtener informes de actividad de los desarrolladores de forma automática —capturando las distintas tareas que efectúan en su ordenador—, con el fin de observar y analizar el esfuerzo dedicado a las distintas fases del desarrollo de una aplicación. Todo ello, cumpliendo las siguientes características:

- No interrumpir ni retrasar el desarrollo de las labores del desarrollador.
- Registrar toda la información requerida, obteniendo una precisión elevada.
- Respetar la privacidad de los usuarios.

Por otro lado, el segundo propósito buscado por este trabajo es el de ofrecer una herramienta sencilla de usar, que permita a sus usuarios interactuar con otras personas en torno a aplicaciones, pudiendo indicar que se está empleando una determinada aplicación y realizar comentarios sobre las mismas.

3.2. Descripción de objetivos

Es preciso detallar los distintos objetivos que se pretenden conseguir al desarrollar este trabajo, partiendo de los propósitos introducidos en el apartado anterior.

3.2.1. Simplificar las opciones de privacidad

Con el fin de facilitar la elección y comprensión de los diversos niveles de privacidad que ofrece el sistema DeTraS, se desarrollará una nueva ventana de configuración de privacidad que permita a los usuarios escoger el nivel deseado de privacidad de una manera aún más simple e intuitiva. De este modo, se pretende ayudar a los usuarios a escoger fácilmente sus preferencias en materia de privacidad, así como hacerles comprender que la aplicación es respetuosa con su privacidad, eliminando posibles problemas y temores al respecto a la hora de emplear el sistema DeTraS.

3.2.2. Mejorar la presentación de datos recogidos localmente

La versión del sistema DeTraS utilizada como punto de partida para el presente trabajo, permite observar los datos recogidos de forma local —el usuario puede consultar los datos en su propia máquina— y remota —examinando datos de diferentes usuarios y ubicaciones—.

En la actualidad, DeTraS provee de herramientas bastante completas que facilitan la visualización remota de datos, así como el análisis y la generación de informes a partir de los mismos. No obstante, la consulta local de datos no se encuentra tan elaborada, por lo que se considera necesario desarrollar una serie de ampliaciones y mejoras en este sentido, ofreciendo la posibilidad de consultar los datos de una forma inteligible y amigable, a la vez que se presentan gráficas y estadísticas que faciliten la comprensión de los mismos.

3.2.3. Difusión de DeTraS

El desarrollo software es una tarea entretenida e interesante, a la par que un reto en ocasiones. Todo ese proceso se lleva a cabo para obtener una aplicación que, generalmente, se espera que sea utilizada con algún propósito. En el caso de DeTraS esto no es una excepción, por lo que el presente trabajo busca promover el uso del sistema desarrollado.

Para la elaboración de este trabajo no se disponen de grandes recursos, por lo que se emplearán algunas de las «armas» que brinda Internet para intentar ampliar el número de usuarios del sistema DeTraS: alojar el proyecto en un sitio web que permita controlar el desarrollo del mismo, ofreciendo a su vez una plataforma donde los usua-

rios de la aplicación puedan colaborar con el proyecto; publicar y mantener entradas para DeTraS en sitios web que sirven para catalogar aplicaciones e informar sobre sus novedades y actualizaciones; crear un sitio web en el que se detallen diversos aspectos del desarrollo, se compartan guías y novedades; y utilizar aplicaciones sociales para permitir a los usuarios estar informados de las novedades del proyecto.

Además de lo comentado previamente, se intentará promocionar la aplicación por otros caminos, como pueden ser: instalando la parte cliente de DeTraS en los laboratorios de GNU/Linux de la Universidad Rey Juan Carlos; redactando y publicando una ponencia para su lectura en conferencias de software libre; y/o publicando artículos en revistas o bitácoras especializadas.

Todo este proceso servirá, a su vez, para ayudar a cumplir el objetivo descrito en la sección 3.2.4.

3.2.4. Creación de una comunidad en torno a DeTraS

Los proyectos de software libre suelen estar estrechamente relacionados con la comunidad formada por los usuarios, desarrolladores y simpatizantes del mismo. Esta comunidad colabora en el proyecto de distintas formas: aportando ideas, detectando defectos en el software, promocionando la aplicación, desarrollando, etcétera.

El presente trabajo tiene como uno de sus objetivos la creación de una comunidad en torno a DeTraS, algo que está fuertemente ligado al objetivo descrito en el punto 3.2.3. Es sencillo observar que si una aplicación no tiene un determinado número de usuarios, no puede surgir una comunidad en torno a ella. No obstante, es preciso trabajar para conseguir formar esa comunidad, ya que el mero hecho de tener usuarios no garantiza la formación de la misma.

Con el fin de conseguir formar esta comunidad, se intentará implicar a los usuarios de DeTraS en diversos aspectos relacionados con el proyecto, entre otros: realizando encuestas para evaluar su satisfacción, solicitando nuevas ideas para incorporar a la aplicación, instando a que informen de los defectos encontrados en el programa y organizando concursos de creación de arte para su uso en el proyecto. Todo ello se realizará a medida que se amplíe el número de usuarios del sistema DeTraS, a través de las plataformas empleadas para la administración y promoción del proyecto.

3.2.5. Desarrollo de una aplicación social

Otro de los objetivos marcados para este trabajo consiste en la creación de una herramienta con carácter social que permita el intercambio de conocimientos y la interacción entre personas a partir de las aplicaciones que emplean habitualmente. Esta herramienta se ha llamado CheckApp y, para evitar dificultar el acceso a la misma, se desarrollará como una aplicación web, de manera que pueda ser empleada por cualquier persona con un navegador web y una conexión a Internet.

La idea es desarrollar una aplicación similar a Foursquare[1] aunque orientada al software, es decir, consistiría en la creación de una aplicación web que permita a los usuarios indicar el programa que están empleando y compartirlo con sus amigos. Para que la experiencia sea más interesante, se podrán crear comentarios sobre las aplicaciones empleadas y se «recompensará» a los usuarios más activos.

Como puede observar el lector, esta aplicación web no está muy relacionada con la orientación que ha tenido el proyecto DeTraS hasta el momento. No obstante, viendo la popularidad de las aplicaciones web sociales en la actualidad, se pretende explorar ese campo y, a su vez, esta herramienta puede ser de utilidad para otros propósitos, arrojando datos interesantes sobre el uso de aplicaciones —aunque, como ocurre con otras herramientas orientadas a este tipo de estadísticas, su precisión no sea demasiado alta—, y ofreciendo un canal en el que los desarrolladores puedan difundir y promocionar sus aplicaciones.

3.2.6. Análisis de datos recogidos

Muchos de los objetivos detallados hasta el momento están bastante alejados de labores de programación, acercándose más a labores de investigación. Esto implica una mayor incertidumbre y complejidad a la hora de examinar los resultados obtenidos tras la elaboración del presente trabajo. Por este motivo, se va a realizar un seguimiento y análisis de diversos aspectos relacionados con estos objetivos, tratando de ofrecer una serie de conclusiones relacionadas con los mismos.

[1] `http://foursquare.com/`

3.2.7. Corrección de defectos

El desarrollo de aplicaciones conlleva defectos de software que deben ser subsanados para obtener un producto de mayor calidad. Uno de los objetivos del presente trabajo radica en resolver los problemas detectados en la base existente de la aplicación, así como corregir los defectos encontrados durante el desarrollo del trabajo, obteniendo al final una aplicación lo más depurada posible.

Capítulo 4

Descripción informática

Este capítulo describe el proceso de desarrollo software llevado a cabo para obtener el software que cumple con los objetivos marcados en el capítulo anterior.

4.1. Metodología

El desarrollo de todo producto software se realiza bajo un determinado modelo de proceso que establece una serie de pasos a realizar, el orden en el que deben ejecutarse y sus relaciones para, partiendo del problema que se pretende resolver inicialmente, guiar el desarrollo hasta el fin de la vida del producto software implementado para resolver el problema.

En la actualidad, existen multitud de procesos software que se adaptan mejor a determinadas circunstancias que a otras. No existe un proceso que se adapte a todas las situaciones, por lo que es preciso analizar detenidamente el proyecto que se va a realizar antes de tomar una decisión.

Para desarrollar el presente proyecto, he escogido un modelo de proceso en espiral, debido a que he considerado que era el que mejor encajaba con el software existente y los objetivos marcados. Los motivos para tomar esa decisión pueden consultarse en la sección 4.1.2.

4.1.1. Modelo en espiral

Este modelo fue definido por primera vez en 1988 por Barry Boehm, y es considerado uno de los modelos de desarrollo más avanzados.

El proceso se representa como una espiral más que como una secuencia de actividad-

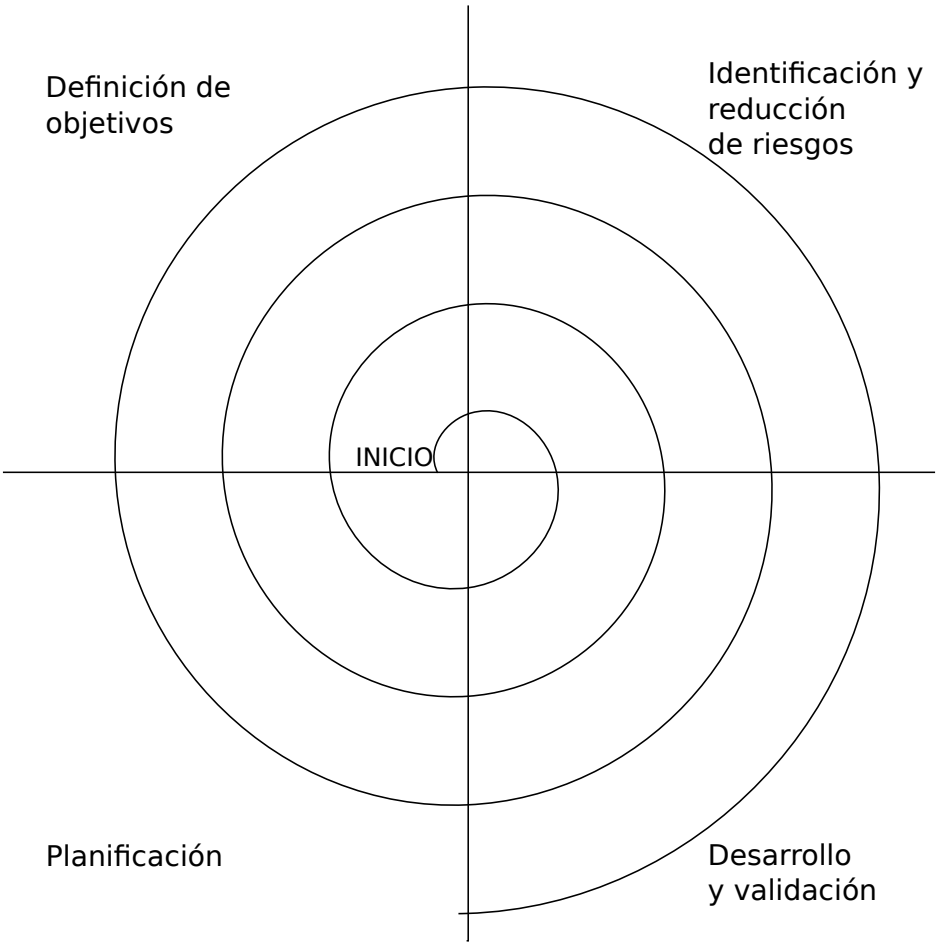


Figura 4.1: Modelo de desarrollo en espiral.

des con vuelta hacia atrás, permitiendo integrar el resultado de cada vuelta a la espiral con el resultado anterior. Además, no hay «etapas» fijas tradicionales, ligadas a actividades como la especificación o el diseño, sino que cada vuelta que se da a la espiral determina las actividades a realizar.

El desarrollo en espiral se puede dividir en las siguientes etapas:

Definición de objetivos. Se identifican los objetivos específicos para cada fase del proyecto.

Identificación y reducción de riesgos. Se identifican y analizan los riesgos clave, sirviendo esta información para minimizarlos.

Desarrollo y validación. El sistema se desarrolla y valida usando casos de prueba, comprobando que los requisitos especificados se cumplen de manera satisfactoria.

Planificación. Se revisa el proyecto y se trazan planes para la siguiente vuelta a la espiral.

4.1.2. Aplicación del modelo al proyecto

Tras detallar las bondades del modelo de desarrollo en espiral, se puede observar que este modelo permite realizar cambios de una forma incremental en el sistema, es decir, cada iteración tiene unos objetivos definidos y su conclusión integra los resultados de la iteración con el sistema existente. Puesto que varios de los objetivos marcados para el presente trabajo consisten en realizar modificaciones al sistema DeTraS existente, se aprecia como el modelo en espiral encaja perfectamente en el caso que aquí nos compete.

Además de los distintos cambios marcados en los objetivos y otras tareas más propias de investigación que no se detallan en este apartado, también existe una meta consistente en desarrollar una aplicación nueva. El modelo de desarrollo en espiral también se adapta a la construcción de nuevos productos software, por lo que este modelo también será empleado en ese caso.

Existen, asimismo, dos razones más para elegir este modelo de desarrollo frente a otros: permite añadir requisitos de forma sencilla y ofrece un resultado al concluir cada iteración. De este modo, se favorece el lanzamiento de versiones no completas de

la aplicación, que permitan obtener realimentación de los usuarios ante los cambios introducidos, al mismo tiempo que se facilita la corrección de errores y la adición de nuevas características, ya que los requisitos se marcan en cada iteración de la espiral.

4.2. Arquitectura general

En este apartado repasaremos la arquitectura de los dos productos software que serán desarrollados a lo largo de este trabajo.

4.2.1. Arquitectura general de DeTraS

El sistema DeTraS está formado por un conjunto de herramientas que permiten seguir las actividades realizadas por desarrolladores. Si se analiza este sistema como un todo, se puede advertir que sigue un modelo de arquitectura cliente–servidor.

Las arquitecturas cliente–servidor están integradas por tres componentes: el cliente, el servidor y la forma en que se comunican. En el caso de DeTraS, el cliente será la parte del sistema que ejecute en los ordenadores de cada uno de los desarrolladores que lo utilicen; el servidor será una máquina dedicada que recibirá los datos de cada uno de los clientes y servirá informes e información a partir de los datos recopilados; y ambos componentes se conectarán a través de una red de ordenadores empleando los protocolos HTTP y HTTPS.

La figura 4.2 muestra gráficamente la arquitectura descrita en las líneas anteriores.

Cliente

Como ya se ha destacado, el cliente es el componente del sistema que se ejecuta en el ordenador de cada uno de los usuarios, existiendo, por tanto, un cliente por cada usuario que tenga la aplicación.

En el sistema DeTraS, la parte cliente está compuesta por tres aplicaciones:

Observador. Es el encargado de detectar y registrar las actividades realizadas por el usuario de la aplicación. El nombre de este componente de DeTraS es TempusFugit.

Notificador. Su nombre es Squealer y tiene como misión enviar los datos recogidos por TempusFugit al servidor de la aplicación.

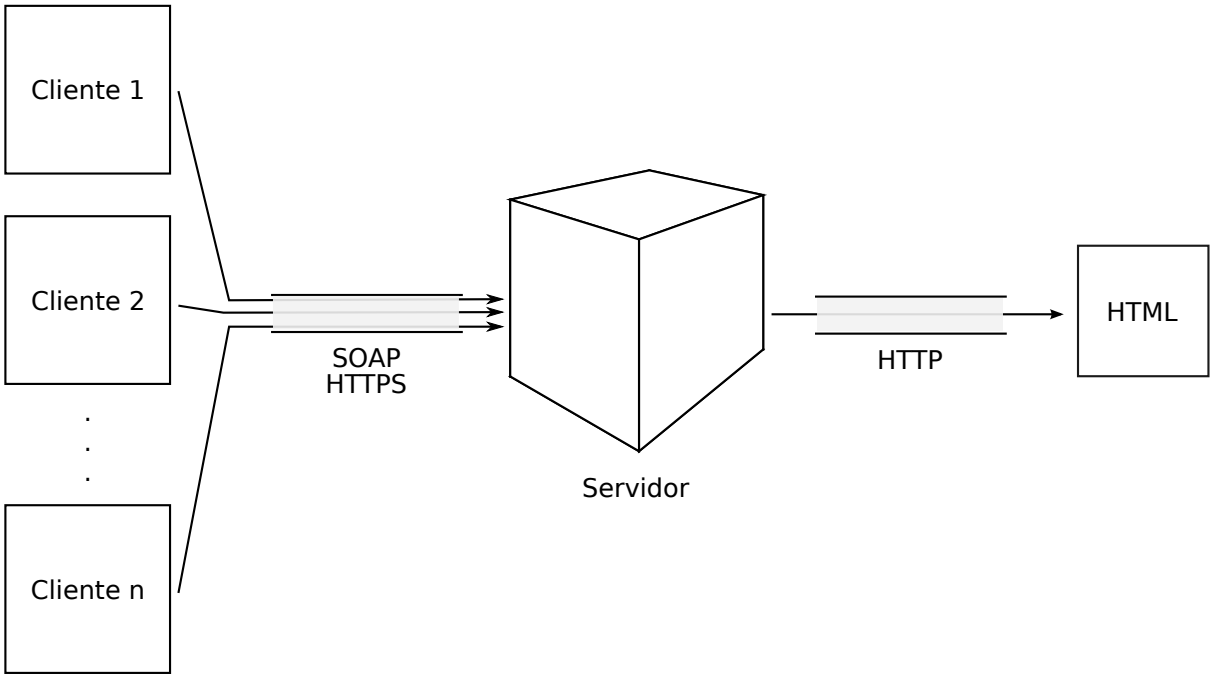


Figura 4.2: Diagrama de la arquitectura de DeTraS.

***Applet* de escritorio.** Este componente permite controlar las aplicaciones anteriores y modificar diversos parámetros de configuración, todo ello de forma gráfica y sencilla.

Servidor

Este elemento del sistema se ejecutará habitualmente en una máquina remota, donde se recibirán y almacenarán de forma persistente los datos enviados por los clientes. Asimismo, ofrecerá la posibilidad de obtener informes e información útil a partir de los datos recogidos.

El servidor de DeTraS está compuesto por las siguientes herramientas:

Recogida de datos. Servicio web cuya tarea es realizar la recepción y almacenamiento de los datos que envían los distintos clientes de DeTraS.

Generador de informes. Permite realizar informes sobre la actividad de los desarrolladores accediendo a los datos recogidos en la base de datos.

Analizador de datos. Esta aplicación web, llamada Dazer, facilita la generación de gráficas y la obtención de información útil partiendo de los datos almacenados

en el servidor. Los clientes de esta aplicación serán aquellos usuarios que accedan a la misma a través de su navegador web.

Comunicación

En el caso de los clientes que envían sus datos al servidor empleando Squealer, la comunicación se llevará a cabo mediante SOAP sobre HTTPS, permitiendo el intercambio seguro de datos entre las aplicaciones, evitando que sean filtradas por cortafuegos o que sean leídas por un atacante.

Cuando los clientes accedan a la aplicación web Dazer, se comunicarán con la misma a través del protocolo HTTP, empleando para ello el navegador web de su preferencia.

4.2.2. Arquitectura general de CheckApp

Como puede imaginar el lector, al ser CheckApp una aplicación web también seguirá un esquema cliente–servidor aunque, en este caso, todo el peso de la lógica de la aplicación estará en el lado del servidor.

El lado del cliente únicamente será encargado de mostrar los datos ofrecidos por el servidor, pudiendo llevar a cabo algunas tareas para mejorar la experiencia de usuario al manejar las vistas.

La comunicación entre el cliente y el servidor se llevará a cabo a través del protocolo HTTP, empleándose para ello un navegador web a elección del usuario.

4.3. Primera iteración: corrección de defectos iniciales

4.3.1. Especificación

El desarrollo de esta iteración y, por tanto, del presente trabajo, parte de la versión 0.4.1 de DeTras que, tras haber sido probada durante algún tiempo, demostró contener algunos problemas que debían ser subsanados. En concreto, estos son los fallos encontrados:

- Error en la detección de dependencias en el fichero `autogen.sh`.
- Imposibilidad de instalar la aplicación mediante el paquete Debian en sistemas cuya versión de Python por defecto no fuese la 2.6.

- Ocasionalmente existían datos temporales erróneos en el cálculo de estadísticas en el servidor.
- Cuando existía un error enviando los datos del cliente, el *applet* mostraba mensajes contradictorios.
- Los datos locales del usuario podían ser borrados si se ejecutaba dos veces Squealer en el mismo minuto.
- Adaptación de los *scripts* de inicio y apagado del servidor al sistema de arranque por dependencias[1].

Además de los errores citados, se decide realizar mejoras en el despliegue de los ficheros al realizar la instalación, cumpliendo las recomendaciones de Debian al respecto. También se toma la decisión de cambiar el paquete Python empleado para trabajar con XML por motivos de eficiencia y para poder realizar una mejor comprobación de los documentos.

4.3.2. Diseño e Implementación

Tras analizar y encontrar las causas de los errores indicados en la especificación, se ha procedido diseñar e implementar soluciones a los mismos. Dado que algunos de los errores detectados no están relacionados con el objetivo del proyecto, se analizarán y describirán las soluciones aplicadas a aquellos errores que sí lo estén.

Cálculo de información temporal erróneos

Después de estudiar el problema, se determinó que el error se encontraba en la parte cliente, concretamente en Squealer. A la hora de recopilar eventos, TempusFugit anota los momentos de inicio y fin de sesión, es decir, agrega un evento cuando comienza a analizar el escritorio del usuario y otro cuando deja de hacerlo. Para realizar los cálculos temporales, el servidor necesita conocer toda esa información al completo. Sin embargo, si se enviaban los datos al servidor mientras TempusFugit estaba corriendo en la máquina del cliente, no se anotaba el evento de fin de sesión, produciendo el error de cálculo.

[1]Puede encontrarse más información sobre este nuevo sistema de arranque la dirección: `http://wiki.debian.org/LSBInitScripts/DependencyBasedBoot`

Tras valorar varias alternativas, se ha decidido que la mejor forma de solventar el problema era que Squealer comprobase si TempusFugit estaba ejecutándose. Si determina que es así, incluirá los eventos de inicio y fin de sesión necesarios para que la información sea coherente. Por tanto, se ha modificado el código de Squealer para que compruebe el estado de TempusFugit al iniciar su ejecución, y se ha agregado a las clases `EventsReader` y `Register` el código necesario para insertar el evento de fin de sesión y el de inicio de sesión, respectivamente.

Borrado de datos locales

Este error se producía porque, cada vez que se enviaban los datos locales al servidor de DeTraS usando Squealer, se almacenaba esa información en un fichero local para facilitar su posterior consulta. El nombre de ese fichero estaba compuesto por una cadena de caracteres fija unida a la fecha, hora y minutos del momento del envío. Por este motivo, si en el mismo minuto se ejecutaba dos veces Squealer, se sobreescribía ese fichero, eliminando los datos registrados.

Puesto que uno de los campos en los que se iba a centrar el presente trabajo, aunque de manera indirecta, era el sistema de almacenaje de eventos —ver apartado 4.5—, se tomó la decisión de aliviar el problema momentáneamente incluyendo en el nombre del fichero de copia el segundo en el que se realizó la misma. De esta manera, el problema es mucho más difícil de reproducir, evitando además buscar una solución más compleja que fuese desechada en poco tiempo.

Mensajes contradictorios al encontrar un error

El error era producido porque se mostraba el mensaje de envío de datos satisfactorio sin tener en cuenta el resultado de la ejecución de Squealer. Para solventar esta situación, se agregó en el *applet* el código necesario para comprobar el resultado de la ejecución de Squealer antes de lanzar el mensaje.

4.4. Segunda iteración: mejoras en la realimentación

4.4.1. Especificación

Pese a no ser un objetivo de desarrollo del presente trabajo, un usuario del sistema solicitó mejorar la forma en la que el *applet* realimentaba al usuario. Como durante la

primera iteración se detectaron dos pequeños problemas en el *applet* y esta petición no era demasiado costosa de realizar, se decidió que era el momento oportuno de acometerla.

Uno de los problemas detectados hacía que la imagen del *applet* no ocupase toda la superficie del mismo, quedando un espacio sin colorear. El otro problema era un mensaje de error que aparecía en ocasiones al eliminar el *applet* del panel. Como ambos problemas eran menores, en adelante me centraré exclusivamente en las mejoras de realimentación.

4.4.2. Diseño

Después de barajar distintas alternativas para ofrecer más información al usuario de una manera no intrusa, se decidió emplear Libnotify para mostrar mensajes de información mientras se utiliza el *applet*. Se escogió Libnotify debido a que es soportado por los entornos de escritorio más utilizados y está instalado por defecto en las distribuciones más populares. No obstante, se ha decidido que, si el usuario no tiene instalada esta biblioteca, el programa debe funcionar como lo hacía hasta ahora, evitando problemas de dependencias.

Otra decisión llevada a cabo es que, en caso que se produzca un error, se emplearía un diálogo emergente en lugar de Libnotify. De esta manera, el aviso llegaría a todos los usuarios, y, al ser distinto a los mensajes generados durante el funcionamiento normal de la aplicación, tendría menos probabilidades de ser ignorado.

Así, en caso que el usuario tenga instalada Libnotify, se mostrarán mensajes cuando TempusFugit inicie o detenga su actividad, cuando Squealer comience a subir los datos del usuario, y cuando Squealer haya subido y almacenado los datos de forma satisfactoria.

4.4.3. Implementación

El uso de la biblioteca Libnotify en Python es muy sencilla. Simplemente hay que importar el módulo `pynotify`, inicializar la biblioteca indicando el nombre de la aplicación, generar la notificación que se desea mostrar creando un objeto de la clase `Notification` y, finalmente, mostrar la notificación con el método `show` de dicho objeto. Puede ver un ejemplo de mensaje realizado con esta biblioteca en la figura 4.3.

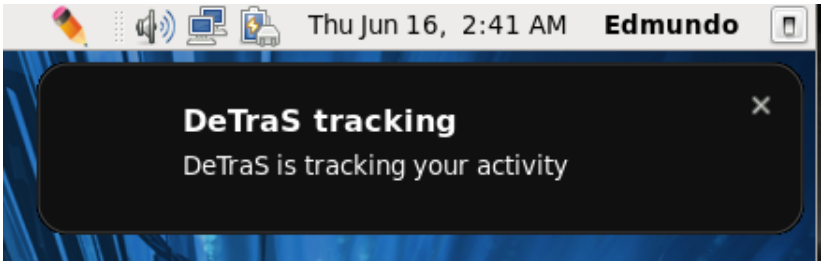


Figura 4.3: Ejemplo de notificación al usuario.

4.5. Tercera iteración: presentación de datos locales

4.5.1. Especificación

Con esta iteración se pretende ofrecer una interfaz gráfica donde se muestren los eventos recogidos por el sistema DeTraS, al mismo tiempo que se complementan con gráficas e información relevante para los usuarios.

Antes de nada, es preciso definir una política de almacenamiento y recuperación de datos. Hasta la fecha, todo el almacenaje de datos realizado de forma local por DeTraS se hacía con ficheros. Esto tiene una clara ventaja si estamos consultando la información recogida desde una terminal, ya que puede accederse a la información con facilidad, aunque tiene una serie de problemas en el caso que nos afecta. Dado que esta es una decisión de gran calado en el desarrollo del trabajo, voy a dedicar unos párrafos a justificar la misma.

Al inicio de esta iteración, TempusFugit almacenaba todos los datos registrados en el fichero `tempusfugit.xml`, que se encontraba en la carpeta `.detras` dentro del directorio de usuario. Cada vez que se enviaban correctamente los datos al servidor, Squealer hacía una copia de ese fichero y la comprimía en ese mismo directorio, vaciando posteriormente el fichero `tempusfugit.xml` para que TempusFugit siguiese almacenando allí los datos recogidos.

Esta política de almacenamiento no es óptima si deseamos ofrecer los datos de una forma dinámica, ya que no conocemos dónde están almacenados los datos de un lapso de tiempo. ¿Qué ocurriría si el usuario desease ver los eventos de ayer y tuviésemos que buscar en cada uno de los ficheros comprimidos con datos ya enviados? Incluso en caso que se llegase a conocer esa información, ¿sería necesario leer todos y cada uno de los ficheros comprimidos para ofrecer las estadísticas y gráficas en la aplicación? Así,

se concluyó que era necesario realizar un cambio en la política de almacenamiento de datos.

En este punto, se realizó un *brainstorming*[2], proponiendo y analizando varias políticas de almacenamiento de datos. Algunas de ellas necesitaban una modificación relativamente profunda del planteamiento existente, otras un cambio total del sistema para pasar a un almacenamiento en base de datos. También aparecieron propuestas mixtas, aunque estas introducían una mayor complejidad y duplicidad de datos. Finalmente, decidí realizar un almacenamiento en base de datos debido a su simplicidad, a su mejor rendimiento y a su facilidad de mantenimiento. Se utilizará SQLite como sistema de gestión de base de datos debido a su sencillez de instalación —recordemos que este software se instalará y ejecutará en el ordenador del usuario, por lo que este es un factor relevante—, y a la integración que tiene con la aplicación, evitando depender de procesos externos.

Una vez definida la política de almacenamiento local de datos, se identificaron las partes del sistema que debían ser adaptadas para funcionar empleando una base de datos. En concreto, es necesario adaptar TempusFugit y Squealer para que gestionen los datos en la base de datos.

Finalmente, se analizó la información que debía ser incluida en la ventana de presentación de datos. Se deseaba que el usuario tuviera constancia del tiempo total que había registrado cada día, ofreciendo, a su vez, esa información de forma visual, enfrentándola a la de otros días. Asimismo, se pretendía ofrecer información de un periodo más amplio de tiempo, para mostrar la evolución de los datos registrados.

4.5.2. Diseño

Dado que se ha tratado el cambio en la política de almacenamiento por separado de la presentación de datos locales, dividiré el diseño e implementación de cada uno de los problemas abordados.

[2]El *brainstorming* o tormenta de ideas, es una técnica para obtener nuevas ideas, consistente en realizar las propuestas que se nos ocurran sin detenernos a analizarlas. Posteriormente, una vez obtenidas varias ideas, se examina cada una de ellas para encontrar la más idónea.

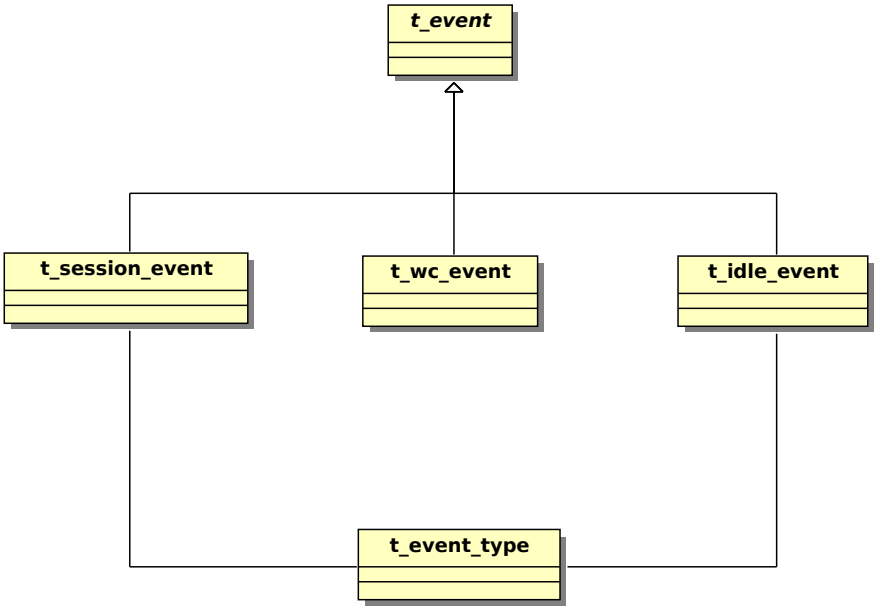


Figura 4.4: Clases del modelo de datos para el almacenamiento local.

Cambio en la política de almacenamiento

Pese a que este cambio afectará tanto a TempusFugit como a Squealer, se va a estudiar el alcance del mismo en ambas aplicaciones conjuntamente.

Inicialmente, se identificaron los datos que debían almacenarse en la base de datos con el fin de poder diseñar una solución con un nivel de abstracción adecuado. El diseño de la base de datos es muy sencillo, como puede comprobar en la figura 4.4. Básicamente, existe una jerarquía de clases para almacenar los eventos que se registren. Además, se ha creado la tabla `t_event_type`, que contiene valores para los distintos tipos de evento que puedan existir.

Gracias al buen diseño realizado con anterioridad en TempusFugit y Squealer, en el que se había contemplado un posible cambio de sistema de almacenamiento, la adaptación de estos programas al uso de una base de datos no fue complicado. En el caso de TempusFugit fue necesario crear una jerarquía de clases que permitiese guardar los eventos registrados en una base de datos. En Squealer se han agregado varias clases para leer y modificar los datos almacenados en la base de datos. Puede observar ambos diseños en las figuras 4.5 y 4.6 respectivamente. Nótese que se han incluido clases relacionadas para comprender mejor los diagramas ofrecidos.

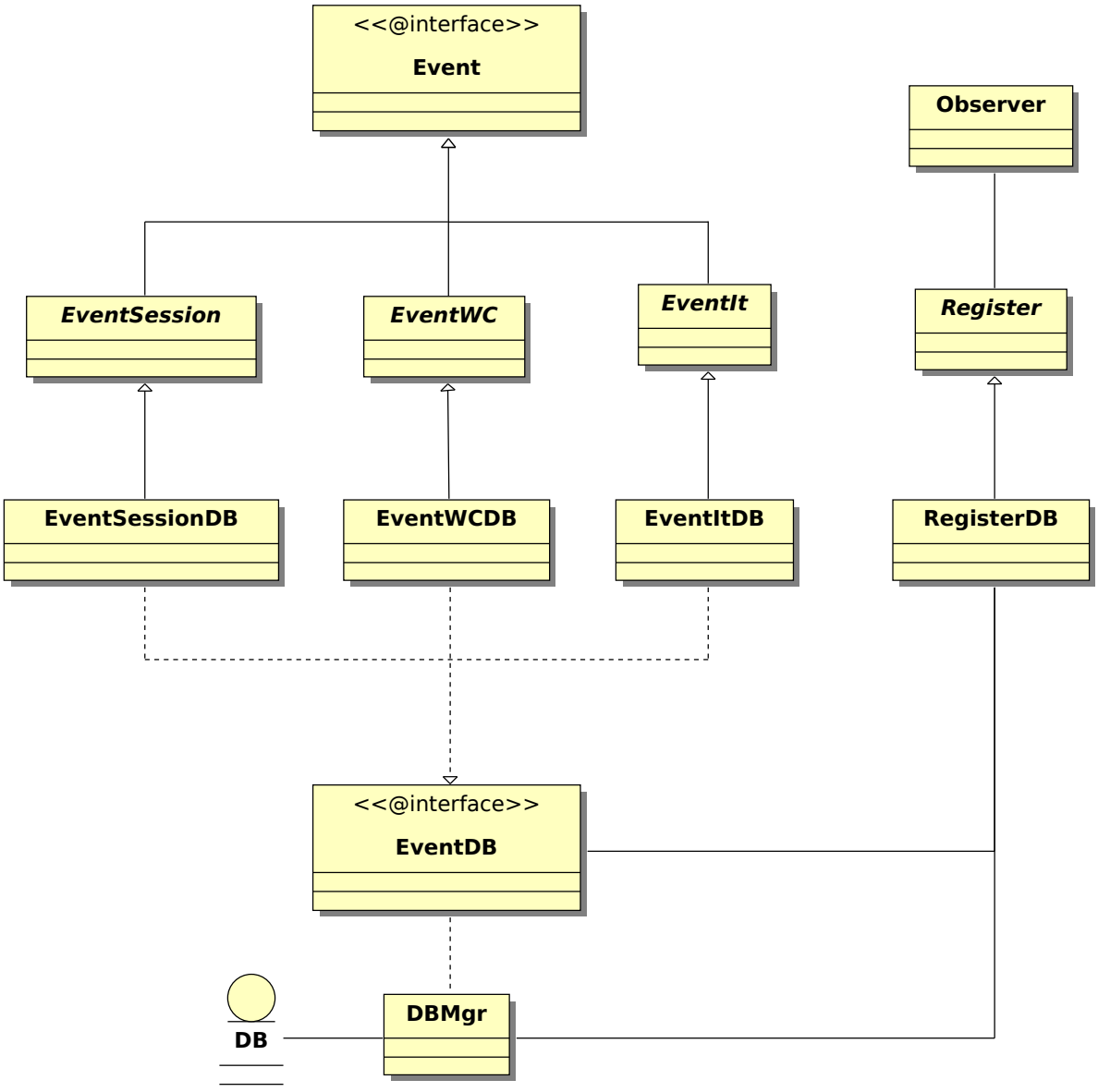


Figura 4.5: Jerarquía de clases de TempusFugit para almacenamiento en BD.

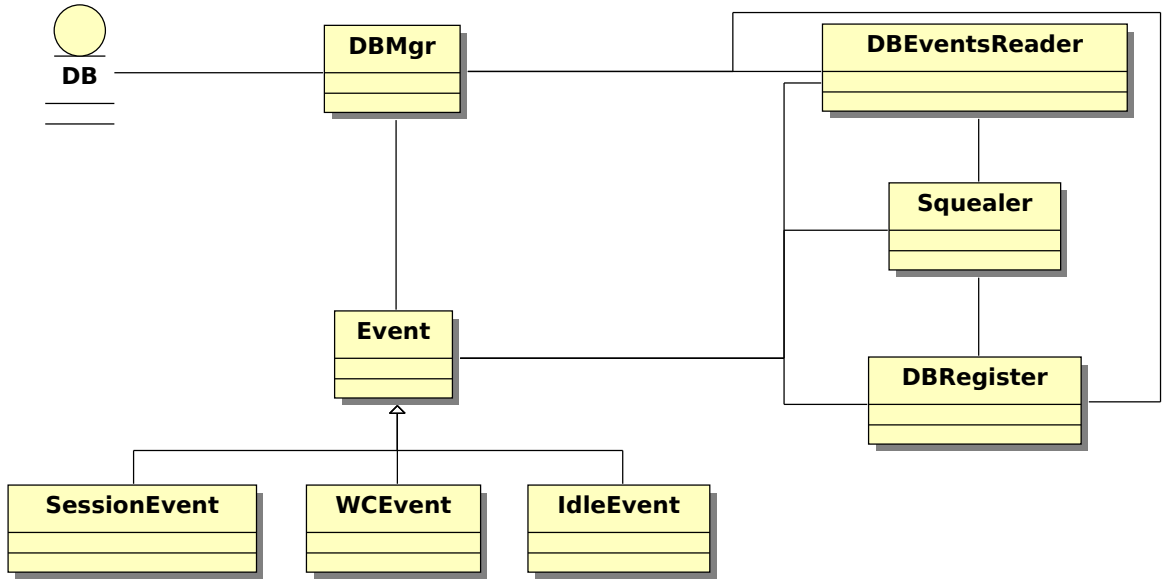


Figura 4.6: Jerarquía de clases de Squealer para almacenamiento en BD.

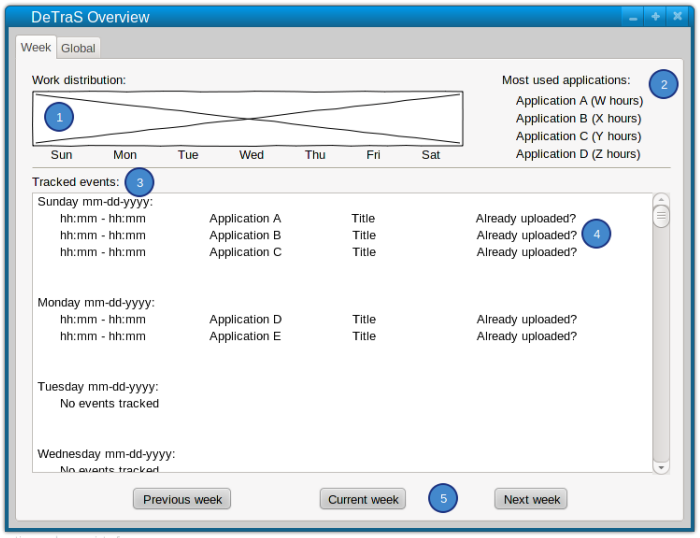
Presentación de datos locales

Una vez identificados los datos que se desean mostrar en la ventana, es necesario crear y evolucionar un boceto de la misma. Estos bocetos permiten seleccionar mejor los elementos que compondrán la interfaz, así como su disposición y ordenación en pantalla. En la figura 4.7 se muestra uno de los bocetos realizados para desarrollar la ventana de presentación de datos locales.

Cabe destacar que, se ha decidido que la ventana de presentación de datos locales esté incluida en el *applet* de escritorio, y que su estructura estará basada en el patrón MVC[3]. Por tanto, la tarea más importante del diseño de esta funcionalidad radica en discernir la separación entre los distintos componentes.

Modelo. El modelo estará formado por las clases necesarias para leer datos de la base de datos, así como clases que representen los objetos allí almacenados. La mayor parte de este diseño es compartido con el realizado en el punto anterior para la lectura de datos en Squealer. Quisiera recalcar, aunque pueda resultar evidente,

[3]Modelo Vista Controlador (MVC) es un estilo de arquitectura de software que separa los datos de una aplicación, la interfaz de usuario, y la lógica de control en tres componentes distintos.



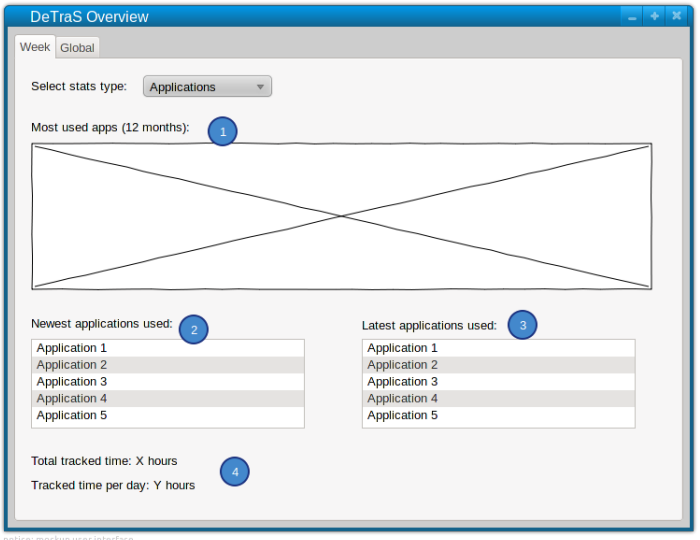
1. Graph showing working time per day.

2. List of most used applications on that week.

3. List of tracked events grouped by day of week.

4. Field to indicate which events haven't been uploaded to DeTraS server.

5. Buttons to change week.

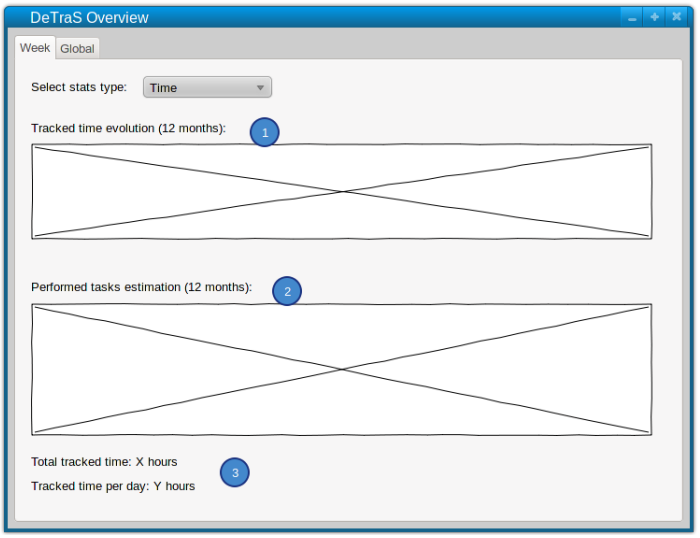


1. Pie chart showing most used apps for 12 months.

2. List of newest applications used.

3. List of latest applications used.

4. General statistics.



1. Graph showing tracked time for 12 months.

2. Pie chart clasifying performed tasks estimation for 12 months.

3. General statistics.

Figura 4.7: Boceto de presentación de datos locales.

que se evitará a toda costa duplicar el código de las clases de estos dos diseños.

Controlador. Básicamente está formado por la clase `Overview`, que contiene diversos métodos para manejar el modelo.

Vista. Está formada por la clase `OverviewDialog`, que realiza las operaciones necesarias para interactuar con el usuario y el controlador.

4.5.3. Implementación

Cambio en la política de almacenamiento

En lo que respecta a TempusFugit, la implementación ha consistido en la codificación y prueba de las distintas clases que componían el diseño mostrado en la figura 4.5. Cada una de las clases que representan los distintos eventos existentes conoce el método al que debe llamar de la interfaz `DBMgr`, que es la encargada de ofrecer una interfaz independiente de la base de datos empleada. Por tanto, cuando la aplicación registra un evento usando la clase `RegisterDB`, se llamará al método adecuado de `DBMgr` para almacenar el evento en cuestión.

Como muestra de la implementación de este cambio de política, puede observar en el listado 4.1 el código que permite almacenar un evento genérico en la base de datos. Tenga en cuenta que SQLite no permite trabajar con objetos, así que el programador debe «simular» la herencia de clases en el modelo de la base de datos. El código mostrado en el listado es llamado independientemente del tipo de evento a almacenar, con el objeto de salvar los datos comunes a cualquier evento y devolver un identificador que permita asociar esa entrada a la del evento concreto que se va a almacenar en su tabla correspondiente.

```
1  static guint64
2  tf_db_mgr_save_event (TfDBMgr *db_mgr, gulong init_time)
3  {
4      guint64 id = 0;
5      int rc = 0;
6      int param = 1;
7
8      debug ("Inserting event...\n");
9
10     /* Initialization of insertion statement */
11     if (!db_mgr->priv->event_insertion)
```

```
12      {
13          debug ("Prepare␣session␣statement\n");
14          rc = sqlite3_prepare_v2 (db_mgr->priv->db_handler,
15                  EVENT_INSERTION_QUERY, -1,
16                  &db_mgr->priv->event_insertion, 0);
17
18          if (rc != SQLITE_OK)
19          {
20              g_critical ("Error␣preparing␣SQL␣statement:␣%s\n",
21                      sqlite3_errmsg(db_mgr->priv->db_handler));
22              return 0;
23          }
24      }
25
26      debug ("Binding␣values:␣%ld\n", (long)init_time);
27      sqlite3_bind_int (db_mgr->priv->event_insertion, param++,
           init_time);
28
29      debug ("Let's␣execute␣the␣statement\n");
30      rc = sqlite3_step(db_mgr->priv->event_insertion);
31
32      switch (rc)
33      {
34          case SQLITE_DONE:
35              debug ("Returned␣DONE\n");
36              break;
37          case SQLITE_BUSY:
38              debug ("Returned␣BUSY\n");
39              break;
40          case SQLITE_ERROR:
41              debug ("Returned␣ERROR\n");
42              break;
43          case SQLITE_MISUSE:
44              debug ("Returned␣MISUSE\n");
45              break;
46          default:
47              debug ("Undefined␣value:␣%d\n", rc);
48      }
49
50      if (rc != SQLITE_DONE)
51      {
52          g_critical ("Error␣executing␣query:␣%s\n",
53                  sqlite3_errmsg(db_mgr->priv->db_handler));
54          return 0;
55      }
56
57      id = (guint64)sqlite3_last_insert_rowid(db_mgr->priv->db_handler)
```

```
        ;
58
59      sqlite3_reset (db_mgr->priv->event_insertion);
60      sqlite3_clear_bindings (db_mgr->priv->event_insertion);
61
62      debug ("Generic␣event␣inserted\n");
63
64      return id;
65  }
```

Listado 4.1: Código para almacenar un evento genérico en la base de datos.

En el caso de Squealer, también se han implementado las clases mencionadas en el diseño. Así, la clase `DBMgr` ofrece una interfaz de acceso a la base de datos que es usada por `DBEventsReader` para leer los datos de la misma. La clase `DBEventsReader` permite, asimismo, obtener la lista de eventos como una lista de objetos `Event` o como un objeto de la clase `lxml.etree` —que no es más que la representación de un documento XML en la biblioteca lxml—, a los objetos que la usan.

Cuando Squealer envía los datos correctamente al servidor, la clase `DBRegister` llama al método `update_sent_flag` de `DBMgr`, actualizando el campo `sent` de los datos enviados para que no vuelvan a ser enviados al servidor de DeTraS.

Presentación de datos locales

Para codificar la presentación de datos locales se han creado las clases indicadas en el apartado de diseño, y se ha concebido una interfaz muy similar a la mostrada en el boceto de la figura 4.7 empleando Glade para lograrlo.

Pese a no tener una dificultad demasiado elevada, este punto ha requerido un esfuerzo bastante alto debido a un problema inesperado: el coste computacional. El problema radica en que el sistema DeTraS, por su diseño, genera una cantidad bastante elevada de datos que, a la hora de ser tratados y analizados, termina resultando muy costoso. Si a esto le sumamos que Python y PyGTK no ofrecen un rendimiento comparable a otros lenguajes de programación como C, el resultado es que la primera versión desarrollada, sin incluir ninguna optimización, era realmente lenta.

Por supuesto, el problema de la lentitud debía ser resuelto, así que se procedió a analizar el código y estudiar las alternativas ofrecidas por Python para mejorar el rendimiento de la aplicación.

En primer lugar, se corrigió un error en la actualización de las gráficas, que se llevaba a cabo constantemente. Así, se crearon las clases `Plot` y `PlotConfig`, que permiten desacoplar la definición de los gráficos de la configuración de los mismos. Además, la clase `Plot` extiende a la clase `DrawingArea` de PyGTK, que es un *widget* de GTK destinado a ser usado para pintar gráficos. Con estos cambios, el rendimiento de la aplicación mejoró considerablemente, aunque seguía siendo algo lenta.

A continuación, se analizaron más puntos donde el rendimiento de la aplicación podía ser mejorado, y se modificaron algunas estructuras de datos, así como la forma de recorrer las listas, haciendo uso de *list comprehensions*[4] y expresiones generadoras[5] en lugar de bucles normales. Tras realizar esta serie de cambios y alguna otra optimización, se consiguió un rendimiento bastante mejor, de modo que es posible emplear la aplicación con un rendimiento aceptable.

Se consideró la posibilidad de reescribir parte de la aplicación en C para intentar mejorar aún más el rendimiento, pero finalmente se descartó porque en este punto, el mayor cuello de botella es el dibujo de gráficas en la interfaz, algo que es difícil de optimizar sin modificar código ajeno o cambiar las tecnologías empleadas. Como el rendimiento actual es aceptable, se consideró que ese cambio excedía los objetivos de este trabajo y, por tanto, debía ser pospuesto.

4.6. Cuarta iteración: simplificar opciones de privacidad

4.6.1. Especificación

Partiendo del diálogo para configurar las opciones de privacidad existente en el *applet* de DeTraS, se pretende realizar una modificación del mismo que permita realizar la configuración de una forma más sencilla e intuitiva.

Esta iteración se aprovechará también para desarrollar una interfaz de texto muy simple, con el fin de acceder a la información recogida por DeTraS de forma local desde la línea de comandos.

[4]Las *list comprehensions* de Python permiten recorrer una lista de forma más compacta y eficiente.

[5]Las expresiones generadoras (*generator expressions*) de Python permiten inicializar las listas de forma dinámica cuando se van a emplear, en lugar de hacerlo de una sola vez. De este modo, permite distribuir el procesamiento a lo largo del tiempo.

4.6.2. Diseño

Simplificar opciones de privacidad

El primer paso llevado a cabo ha sido realizar distintos bocetos de la nueva interfaz gráfica, uno de los cuales puede observarse en la figura 4.8. Debido al diseño del *applet* del sistema DeTraS, el código no sufrirá prácticamente ningún cambio, salvo cambios en los métodos que manejan los eventos de los elementos de la interfaz modificados.

Los cambios diseñados para la ventana de preferencias de privacidad han sido pensados para que no precisen ninguna alteración en el fichero que almacena las preferencias o las clases que las manejan. Simplemente, se ha conseguido encontrar una solución donde un solo control realice cambios en el tipo de filtrado que se quiere realizar y si el usuario consiente o no enviar su nombre.

Interfaz de texto para consultar datos locales

El diseño de esta interfaz de texto utiliza el patrón MVC, por lo que se va a reutilizar tanto el modelo como el controlador desarrollados en la iteración anterior para mostrar datos localmente. Así, simplemente habrá que diseñar una clase que se encargue de recibir las opciones desde la línea de comandos y mostrar los resultados por pantalla.

4.6.3. Implementación

Simplificar opciones de privacidad

Tras realizar el diseño final de la pantalla de privacidad en Glade, se ha integrado con el resto de pestañas que componen el diálogo de preferencias de DeTraS.

Posteriormente, se han modificado los métodos encargados de manejar la interfaz de usuario de las preferencias de privacidad para adaptarse a la nueva vista.

Interfaz de texto para consultar datos locales

Como se comentó en la etapa de diseño, se ha creado una nueva aplicación en Python que, básicamente, añade una clase —llamada `OverviewCLI`— que se conecta con el controlador desarrollado en la iteración anterior y permite al usuario interactuar con los datos existentes en la base de datos de eventos.

Para recoger los datos introducidos en la terminal se ha empleado el paquete de Python argparse, que facilita enormemente la comprobación de las distintas acciones

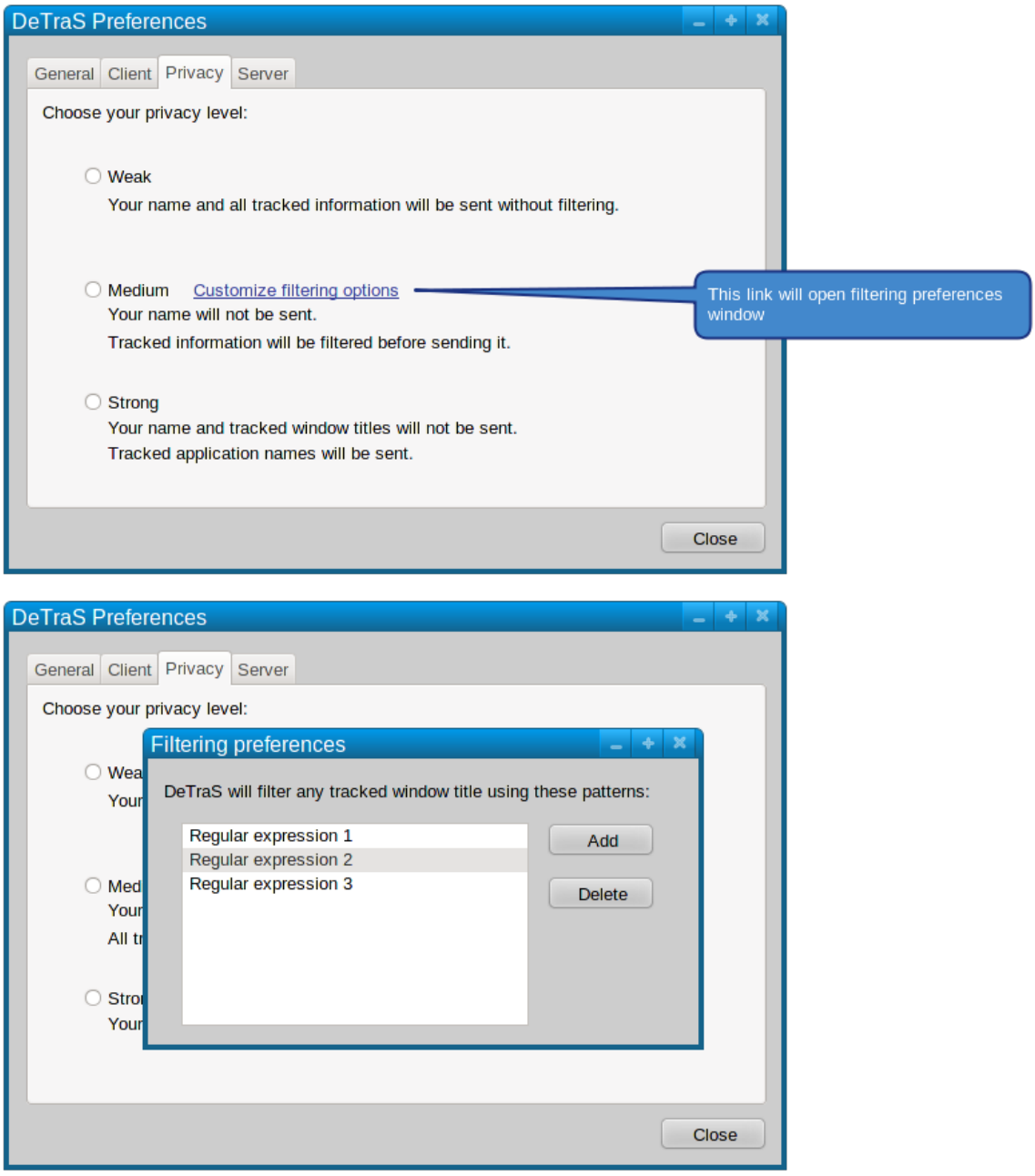


Figura 4.8: Boceto de las opciones de privacidad.

y opciones introducidas.

4.7. Quinta iteración: CheckApp

4.7.1. Especificación

Puesto que esta iteración generará una aplicación completamente nueva, voy a incluir en este apartado una definición formal de los requisitos planteados antes de llevarla a cabo.

1. Cada uno de los usuarios dispondrá de un perfil personal, que podrá estar asociado a otros perfiles de usuarios, que serán sus amigos dentro de la plataforma.

2. Existirá una lista de aplicaciones, que podrá ser ampliada por los usuarios. Por defecto, el usuario que inserte una aplicación, será el encargado de editar su información. Los creadores de una aplicación podrán tomar el control de su perfil contactando con el administrador de la página.

3. Los usuarios podrán indicar que están usando una aplicación —acción que he denominado *check–app*—. Sólo se podrán realizar un máximo de 5 *check–apps* por día.

4. Los usuarios podrán indicar un comentario en su *check–app*. Asimismo, podrán insertar comentarios en las aplicaciones. También será posible responder a esos comentarios.

5. Existirá una pequeña recompensa para los usuarios, reconociendo los méritos que consigan usando CheckApp. En principio se darán méritos en función del número de *check–apps* y comentarios realizados. La recompensa será un pin virtual, que podrán lucir orgullosos en su perfil.

6. Aunque no sea un requisito en sí mismo, se intentará minimizar el tratamiento de datos personales de los usuarios, eliminando posibles problemas de privacidad.

4.7.2. Diseño

CheckApp es desarrollada empleando el *framework* de desarrollo web Django, que obliga a las aplicaciones a adoptar un patrón MVC. Así, el punto más importante del

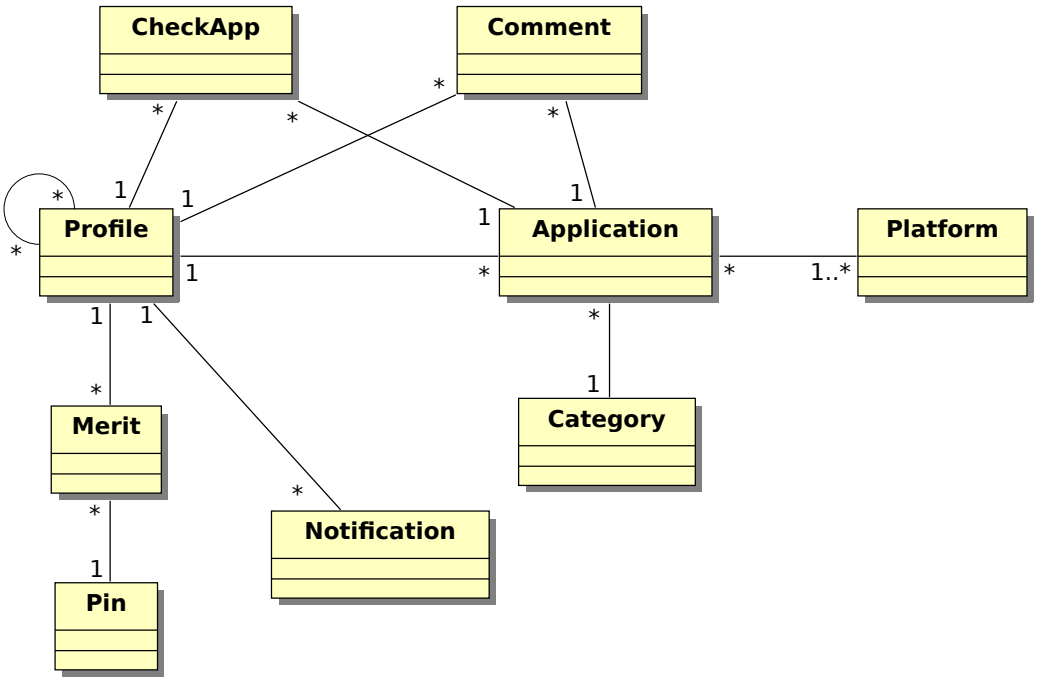


Figura 4.9: Clases del modelo de CheckApp.

diseño de la aplicación será determinar cómo dividirla en los tres componentes que forman parte del patrón.

Modelo. El modelo debe definir y almacenar los datos que maneja la aplicación. Así, se ha diseñado el diagrama de clases mostrado en la figura 4.9, para poder almacenar en la base de datos de la aplicación los distintos usuarios, aplicaciones y entidades relacionadas.

Controlador. Es el componente que contiene la lógica de la aplicación. Se han definido diversas clases —que pueden observarse en la figura 4.10— para permitir que cuando el usuario interactúe con las vistas, estas acciones se procesen y se trasladen al modelo si fuese preciso.

Vista. Es el elemento del patrón que ve y maneja el usuario. Es importante crear una interfaz de usuario intuitiva, amigable, atractiva y que responda adecuadamente a las acciones que el usuario realice. En este caso, al ser CheckApp una aplicación web, las vistas serán realizadas en HTML, complementado con CSS para definir su diseño y otras tecnologías para mejorar la experiencia de usuario, y serán

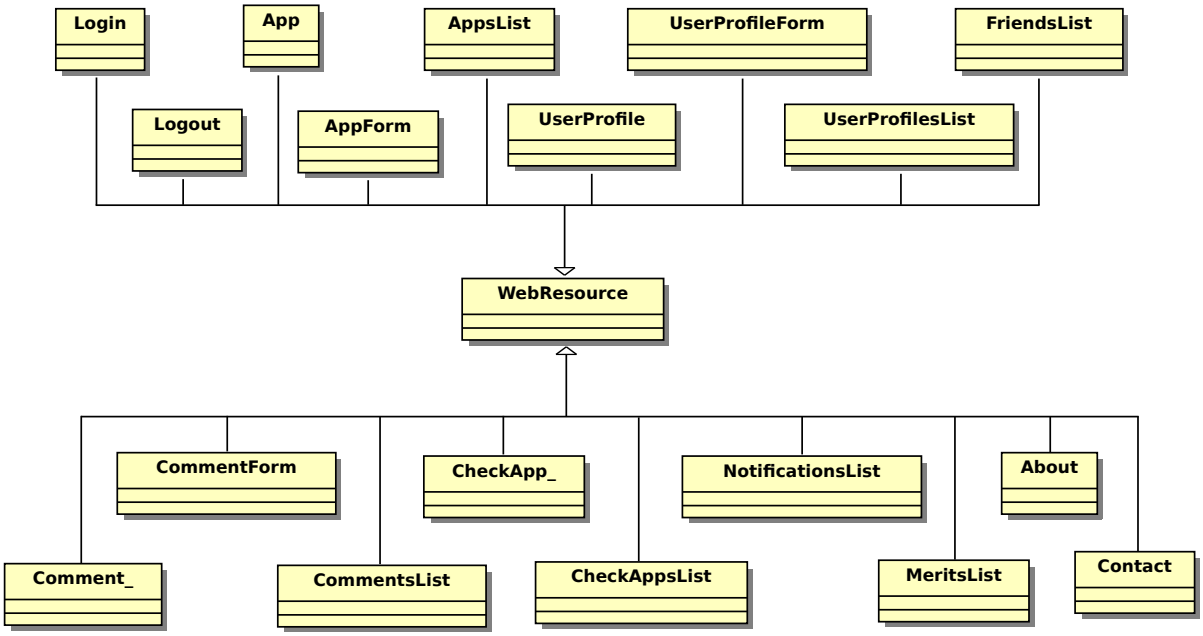


Figura 4.10: Clases del controlador de CheckApp.

accesibles mediante un navegador web. Al igual que otras interfaces gráficas definidas durante el trabajo, se ha realizado una evolución de la misma empleando bocetos.

4.7.3. Implementación

En primer lugar, se codificaron las distintas clases indicadas en el diseño del modelo de la aplicación. A su vez, se crearon las mismas en la base de datos con el fin de simplificar la fase de depuración.

Posteriormente, se definieron las distintas URL[6] que tendrían los recursos ofrecidos por la aplicación. Cada vez que se definían las URL de un recurso, se codificaban las clases relacionadas con el mismo en el controlador de la aplicación, así como una primera versión de las vistas del recurso.

[6]Un localizador uniforme de recursos, más comúnmente denominado URL —*Uniform Resource Locator*—, 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, como por ejemplo documentos textuales, imágenes, vídeos, presentaciones digitales, etc.

Cabe destacar que la clase `WebResource`, de la que heredan el resto de clases que conforman la lógica de la aplicación —salvo algunas clases complementarias—, permite a las clases que la extienden soportar los distintos métodos definidos por HTTP, es decir, los métodos *GET*, *POST*, *PUT* y *DELETE*, así como manejar parámetros que se incluyan dentro de las propias URL de la aplicación.

Para completar la implementación, una vez definidas y codificadas todas las clases y vistas que definen a los recursos, se mejoró la apariencia de la aplicación y se realizaron numerosas pruebas para verificar su funcionamiento.

4.7.4. Despliegue

Puesto que el GSyC[7] ha cedido amablemente un servidor para la realización de mi anterior Proyecto Fin de Carrera y el presente Trabajo Fin de Máster, se ha empleado el mismo para, además de albergar el servidor que recibe y almacena los datos recogidos por DeTraS, alojar la aplicación web CheckApp.

En primer lugar, se realizó la instalación de los distintos programas que precisa la aplicación para funcionar: un servidor Apache, Python 2.5 o superior, Django 1.2, South[8], MySQL y PostFix. CheckApp necesita instalar un servidor de correo electrónico, PostFix en este caso, para enviar enlaces de recuperación de contraseñas de usuario.

Una vez configurados Python y Django, se descargó la aplicación y se editaron las preferencias de la misma, modificando diversos datos para ajustarse al sistema en el que se estaba instalando y para que no compartiese contraseñas con la versión distribuida de la aplicación.

Seguidamente, se creó el usuario empleado por la aplicación en MySQL y se configuraron sus permisos para poder generar todo el modelo desde Django.

Posteriormente, tras comprobar que se pudo generar el modelo de datos correctamente, se llevó a cabo la configuración del servidor Apache, definiendo un *VirtualHost* para la aplicación con las directivas necesarias para servir las distintas páginas y los ficheros estáticos.

Por último, se adquirió el dominio `checkapp.net` para poder acceder a la aplicación de forma cómoda y profesional. En el momento de escribir este trabajo la apli-

[7] Grupo de Sistemas y Comunicaciones de la Universidad Rey Juan Carlos

[8] South es una herramienta que se integra con Django y permite realizar migraciones de los datos cuando se producen modificaciones del modelo.

cación continúa desplegada en el dominio indicado. Si lo desea, puede comprobar el funcionamiento de la red social desarrollada visitando `http://checkapp.net`.

Capítulo 5

Estrategias de promoción y distribución

Este capítulo recoge las distintas tareas llevadas a cabo a lo largo del presente trabajo, al margen del desarrollo informático descrito en el capítulo 4, y que están encaminadas a aumentar la difusión y distribución de las aplicaciones desarrolladas.

5.1. Ampliar la difusión de DeTraS

Como se mencionó en los objetivos, el presente trabajo no dispone de grandes recursos para invertir en publicidad o llevar a cabo acciones mediáticas. Así, se ha intentado ampliar la difusión del sistema DeTraS utilizando las herramientas que ofrece Internet. A continuación se presentan y describen algunas de las labores realizadas para intentar aumentar el número de personas que utilizan y conocen DeTraS.

5.1.1. Creación de páginas del proyecto

Uno de los primeros pasos que se llevaron a cabo al retomar el desarrollo del proyecto fue la creación de una cuenta en un sitio web que permitiese alojar el código y ofreciese herramientas para gestionar el desarrollo del proyecto.

Tras valorar varias alternativas existentes, decidí escoger Launchpad como plataforma web para gestionar el proyecto debido a que tenía cierta experiencia previa con la plataforma, la integración con Bazaar —el sistema de control de versiones escogido para el proyecto— es muy buena, brinda la opción de crear repositorios personales de paquetes para Ubuntu y Debian, y ofrece una serie de herramientas bastante útiles y modernas.

Así, se migró el código existente a la página del proyecto en Launchpad[1] y se comenzó a emplear esa plataforma para alojar el código, gestionar los errores detectados en la aplicación y, posteriormente, alojar las descargas de los distintos lanzamientos que se fueron produciendo, manteniendo también un repositorio de paquetes para los usuarios que quisiesen instalar DeTraS de ese modo.

No obstante, a medida que se avanzaba en el desarrollo del proyecto, se comprobó que existían algunas herramientas que podían ser de interés pero que Launchpad no ofrece. Por tanto, se decidió crear un perfil en SourceForge para DeTraS[2], empleando sus servicios de listas de correo, alojamiento de descargas y *wiki*.

Al mismo tiempo, se creó un registro para DeTraS en Freshmeat[3], donde se han ido anunciando desde entonces las distintas versiones publicadas junto a cierta información del sistema, como capturas de algunas aplicaciones que lo componen y enlaces relacionados con el mismo.

5.1.2. Mejora en la documentación

Un producto software necesita disponer de documentación que ayude a los usuarios si lo necesitan, especialmente cuando comienzan a utilizar el sistema. Por este motivo, se decidió trabajar en la documentación del proyecto.

Al inicio, toda la documentación de uso e instalación del sistema DeTraS estaba incluida en el propio archivo de distribución, junto al código fuente del programa. En primera instancia, se decidió mantener este sistema de documentación, aunque sería necesario complementarlo con documentos más elaborados y completos una vez se consiguiesen ciertos avances en el código de la aplicación.

Una vez alcanzados los hitos planteados en el desarrollo, se elaboró un manual de uso de la aplicación en formato PDF que se puso a disposición de los usuarios para su descarga. Este manual especificaba, de una forma bastante detallada, las funciones de los distintos componentes de DeTraS y solicitaba a los usuarios que informasen de cualquier error, sugerencia o duda.

Finalmente, tras comprobar el éxito de descargas del manual de usuario, se decidió que sería una buena idea ampliar la documentación del proyecto. Así, se rellenó la *wiki*

[1] Página de DeTraS en Launchpad: `https://launchpad.net/detras`
[2] Página de DeTraS en SourceForge: `http://sourceforge.net/projects/detras`
[3] Página de DeTraS en Freshmeat: `http://freshmeat.net/projects/detras`

del proyecto en SourceForge con contenido sobre la instalación, uso, preguntas más frecuentes y maneras de colaborar con DeTraS. Por supuesto, está invitado a consultar la *wiki* del proyecto DeTraS visitando la siguiente dirección:

`http://sourceforge.net/apps/mediawiki/detras/index.php`

5.1.3. Creación de una bitácora

Otro de los pasos dados para intentar ganar usuarios consistió en la creación de una bitácora en Wordpress[4] para publicar novedades e información del proyecto, junto a guías que escribía para ayudar a otras personas en diversas tareas que había realizado en el desarrollo del mismo. En general, tal y como comentaré en el apartado 5.2.1, pretendía convertir la bitácora en el centro de la comunidad de DeTraS.

Pese a esforzarme en generar información para la bitácora, no terminaba de conseguir las visitas deseadas, así que decidí unir la bitácora a dos planetas[5] de los muchos que agregan noticias relacionadas con el software libre: Planeta Linux[6] y LinuxPlanet [7]. Si usted sigue cualquiera de esos dos planetas, estará enterado de las noticias surgidas en torno al proyecto DeTraS.

5.1.4. Promoción en redes sociales

Esta fue otra de las medidas llevadas a cabo para difundir el proyecto DeTraS. Concretamente, se creó una cuenta de Twitter para el proyecto[8], en la que se publican, de manera muy breve —tal y como exige el funcionamiento de esta red social—, nuevos lanzamientos de la aplicación y nuevos contenidos en la bitácora.

5.1.5. Creación de un logotipo para DeTraS

Se consideró que la creación de un logotipo para el proyecto DeTraS sería buena idea, ya que permitiría asociar una imagen a todo el proyecto, distinguiéndolo de otros.

[4]Bitácora de DeTraS: `http://detrasproject.wordpress.com/`

[5]En este contexto, el término planeta se refiere a un sitio web cuyo contenido está compuesto exclusivamente por los mensajes de un conjunto de bitácoras que normalmente comparten una temática común.

[6]`http://www.planetalinux.org/`

[7]`http://www.linuxplanet.org/`

[8]Cuenta de DeTraS en Twitter: `http://twitter.com/detrasproject`

En primera instancia, se realizó el concurso indicado en el apartado 5.2.2, intentando que el logotipo fuese realizado y elegido por la comunidad. No obstante, tras el fracaso de dicho concurso, yo mismo diseñé el logotipo del proyecto usado en la actualidad.

5.1.6. Acercar la aplicación a más distribuciones

Pese a que los usuarios de Ubuntu y Debian tenían distintas alternativas para instalar DeTraS de una forma sencilla, los usuarios de otras distribuciones no lo tenían tan fácil. Tenían que descargar y compilar el código fuente de la aplicación, haciendo que la instalación fuese demasiado compleja para usuarios nuevos o para aquellos sin conocimientos técnicos. De este modo, se han creado paquetes RPM de DeTraS para Fedora —una distribución muy popular entre los usuarios de GNU/Linux—, permitiendo simplificar la instalación.

5.2. Creación de una comunidad en torno a DeTraS

De cara a fomentar la creación de una comunidad en torno a DeTraS se han llevado a cabo distintas medidas que se explican en esta sección.

5.2.1. Creación de una bitácora

La creación de la bitácora comentada en el apartado 5.1.3 tenía como objetivo conseguir un determinado número de lectores que, tras leer los artículos de la bitácora, y conocer las bondades del sistema DeTraS, se interesasen por el mismo. De conseguir esa masa crítica de usuarios, la bitácora sería una pieza clave en la interacción con la comunidad, permitiendo discutir el rumbo del proyecto de forma colaborativa, organizar diversos concursos y obtener realimentación de forma directa.

5.2.2. Concurso de creación de arte

Una de las ideas que se llevaron a cabo para involucrar más a los usuarios de DeTraS en el proyecto fue la realización de un concurso de creación de arte para el propio proyecto. El concurso consistía en el diseño de un icono para DeTraS —que sería empleado además como logotipo del proyecto—. Para ello, los concursantes disponían de

un periodo de tres semanas, tras los cuales se llevaría a cabo una votación para escoger al agraciado. Desgraciadamente, no pude ofrecer un premio suculento, aunque, además de prometer alguna recompensa tangible, se galardonaba al ganador con aparecer en los créditos de la aplicación y en la bitácora.

5.2.3. Solicitudes de colaboración

Pese a no representar un esfuerzo demasiado significativo, quería destacar que durante todo el proceso de desarrollo y promoción se mantuvo una postura participativa, solicitando activamente a los usuarios conocer sus opiniones y participar, ya fuese indicando errores, proponiendo ideas o ayudando en el control del funcionamiento de la aplicación.

5.3. Modelos de negocio de CheckApp

Pese a no ser uno de los objetivos del trabajo, quería dejar constancia que no sólo se ha trabajado en el desarrollo de CheckApp, sino que se ha valorado el futuro del proyecto más allá del presente trabajo.

A pesar de que se ha dispuesto de poco tiempo para trabajar en la distribución de CheckApp, se han seguido algunos de los pasos destacados en el punto 5.1. La acogida del proyecto por parte de los usuarios parece haber sido bastante buena pese a tratarse del primer lanzamiento del mismo.

Además, se han estudiado algunos modelos de negocio que podrían hacer que CheckApp obtuviese beneficios para financiar el desarrollo del proyecto.

Como ya se ha comentado con anterioridad, se ha desplegado la versión actual de DeTraS en un servidor, haciendo que cualquier persona con acceso a Internet pueda abrir su cuenta de usuario en la aplicación. En un futuro, si la red social consigue la popularidad necesaria, los desarrolladores y distribuidores de software podrían estar interesados en promocionar sus aplicaciones en CheckApp, y podrían representar una fuente de ingresos interesante.

Una de las posibilidades de obtención de beneficios radicaría en la promoción en resultados de búsqueda. La idea es que los dos o tres primeros resultados de la búsqueda fuesen enlaces a aplicaciones promocionadas. Por tanto, los desarrolladores de software podrían pagar por aparecer en esa zona privilegiada cuando la búsqueda es-

tuviese relacionada con su aplicación. El precio de aparecer en la zona de promoción se incrementaría en función de la demanda del término de búsqueda en el que se desease aparecer.

Otra de las posibilidades pasaría por ofrecer opciones de promoción adicionales en el perfil de aplicación a los desarrolladores software. Así, podría venderse un componente para el perfil de aplicación que permitiese, por ejemplo, crear encuestas dentro de CheckApp.

Finalmente, también podrían obtenerse beneficios creando méritos especiales para determinadas aplicaciones software. De este modo, si un desarrollador desea promocionar su software, podrían crearse y anunciarse méritos exclusivos para los usuarios que hagan *check–app* o comenten en aplicaciones de ese desarrollador.

Capítulo 6

Resultados Experimentales

Una vez se han completado las acciones descritas en los capítulos 4 y 5, es momento de comprobar los resultados que han producido.

6.1. Resultados de desarrollo

Puesto que el funcionamiento básico del sistema DeTraS ya se demostró en el Proyecto Fin de Carrera entregado el curso pasado, en esta ocasión voy a analizar dos de las principales novedades desarrolladas: la interfaz de datos locales de DeTraS y la aplicación web CheckApp.

6.1.1. Presentación de datos locales

Con el fin de comprobar el funcionamiento de la ventana de presentación de datos locales, se examinarán los datos recogidos en mi equipo durante la realización del presente trabajo. Para demostrar que la información ofrecida por la herramienta desarrollada es correcta, se han subido los mismos datos a una instancia independiente del servidor de DeTraS, permitiendo comparar los resultados obtenidos por el *applet* de DeTraS —en el que está incluida la presentación de datos locales— con los resultados de Dazer, la herramienta de análisis de datos globales.

Semana del 30 de mayo al 5 de junio de 2011

Una de las funciones de la vista de presentación de datos locales permite consultar los datos obtenidos semana a semana. En este ejemplo analizaremos la semana del 30 de mayo al 5 de junio de 2011. La suma del tiempo recogido durante cada día de esta

Día	Tiempo registrado
30/05	02:13
31/05	03:17
01/06	01:10
02/06	00:00
03/06	01:30
04/06	04:11
05/06	04:41

Tabla 6.1: Cálculo del tiempo registrado localmente (30/05/2011 – 05/06/2011).

Aplicación	Tiempo registrado (minutos)
Terminator	481
Chromium-browser	442
Firefox	46
Spotify	19
Evolution	17

Tabla 6.2: Cinco aplicaciones más usadas según el *applet* de DeTraS (30/05/2011 – 05/06/2011).

semana se puede consultar en la tabla 6.1, que puede contrastarse con la salida ofrecida por la aplicación en la figura 6.1.

Básicamente, puede verificar que la gráfica mostrada en la parte superior izquierda de la captura de la ventana de la aplicación se ajusta perfectamente a los datos suministrados, realizando a su vez una comparación del tiempo registrado entre cada uno de los días de la semana.

Desgraciadamente, no es posible ofrecerle todos los eventos que maneja DeTraS para que compruebe el resultado de las distintas estadísticas debido a que el volumen de datos es demasiado elevado. En la semana que estamos analizando se produjeron un total de 936 eventos que la aplicación debe analizar para calcular la información que muestra en pantalla. No obstante, podemos contrastar los resultados de las aplicaciones más empleadas en esa semana con los datos que nos ofrece la herramienta de análisis Dazer comparando las tablas 6.2 y 6.3.

Cabe destacar que DeTraS elimina de estas estadísticas los periodos de inactividad del usuario, por lo que los datos ofrecidos sólo tienen en cuenta el tiempo de uso real del sistema.

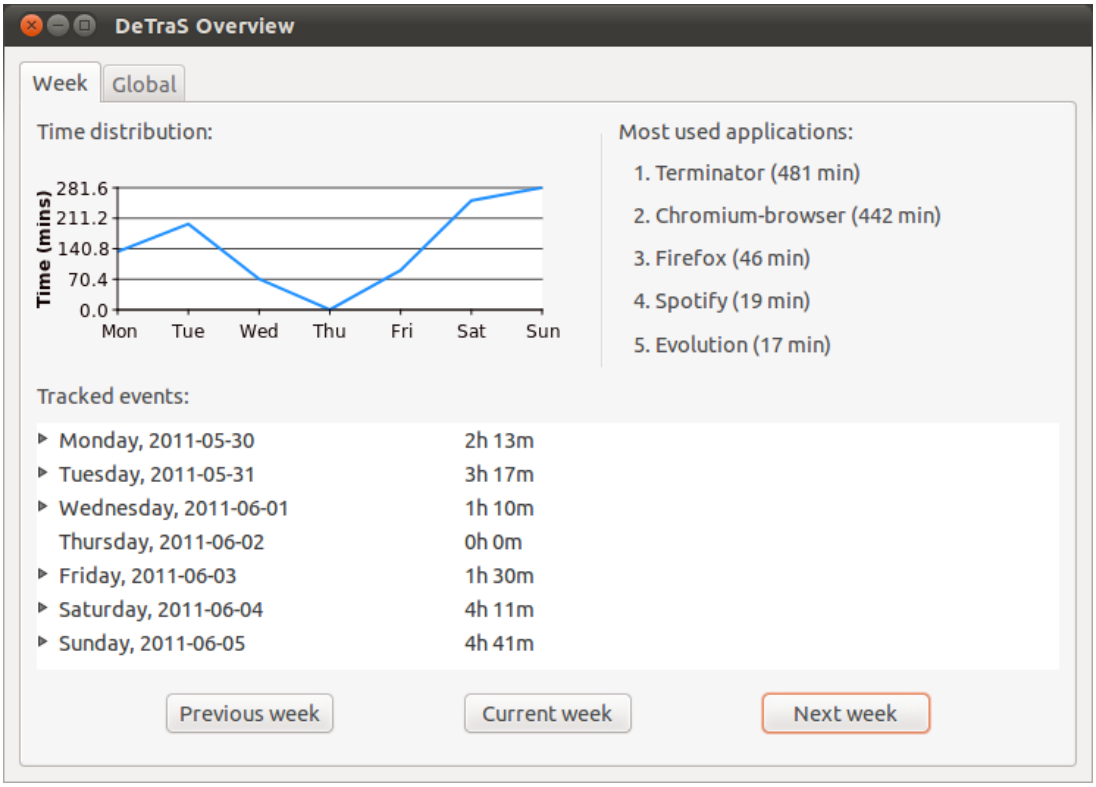


Figura 6.1: Vista de datos locales (30/05/2011 – 05/06/2011).

Aplicación	Tiempo registrado (minutos)
Terminator	481.74
Chromium-browser	442.38
Firefox	46.74
Spotify	19.86
Evolution	17.58

Tabla 6.3: Cinco aplicaciones más usadas según Dazer (30/05/2011 – 05/06/2011).

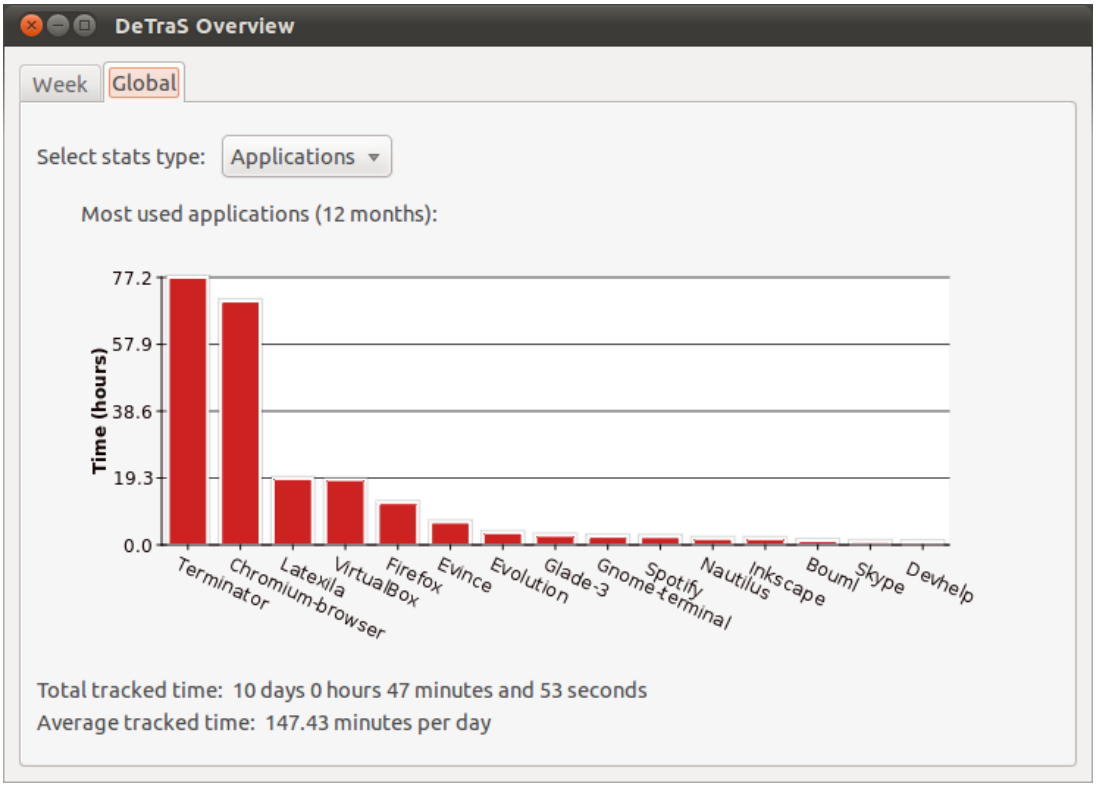


Figura 6.2: Vista de aplicaciones más usadas del *applet* de DeTraS (14/03/2011 – 19/06/2011).

Datos globales

En esta ocasión vamos a analizar los resultados totales mostrados en la presentación de datos locales. Nuevamente, la mejor manera de comprobar su corrección es compararlas con los resultados obtenidos en Dazer. Por un lado, puede comparar las figuras 6.2 y 6.3, que representan las aplicaciones más empleadas por mí desde el 14 marzo de 2011 —fecha en que se terminó de implementar el almacenamiento de datos locales en base de datos— hasta el 19 de junio de 2011. También puede consultar en la tabla 6.4 el tiempo exacto que se han empleado las distintas aplicaciones de las gráficas.

Como podrá observar, la información ofrecida cuadra perfectamente, lo que constituye una buena señal.

Por último, vamos a analizar los resultados obtenidos cuando analizamos la evolución del uso de DeTraS a lo largo del tiempo. Para ello, vamos a volver a comparar las respuestas ofrecidas por Dazer y el *applet* de DeTraS mostradas en las figuras 6.4 y 6.5.

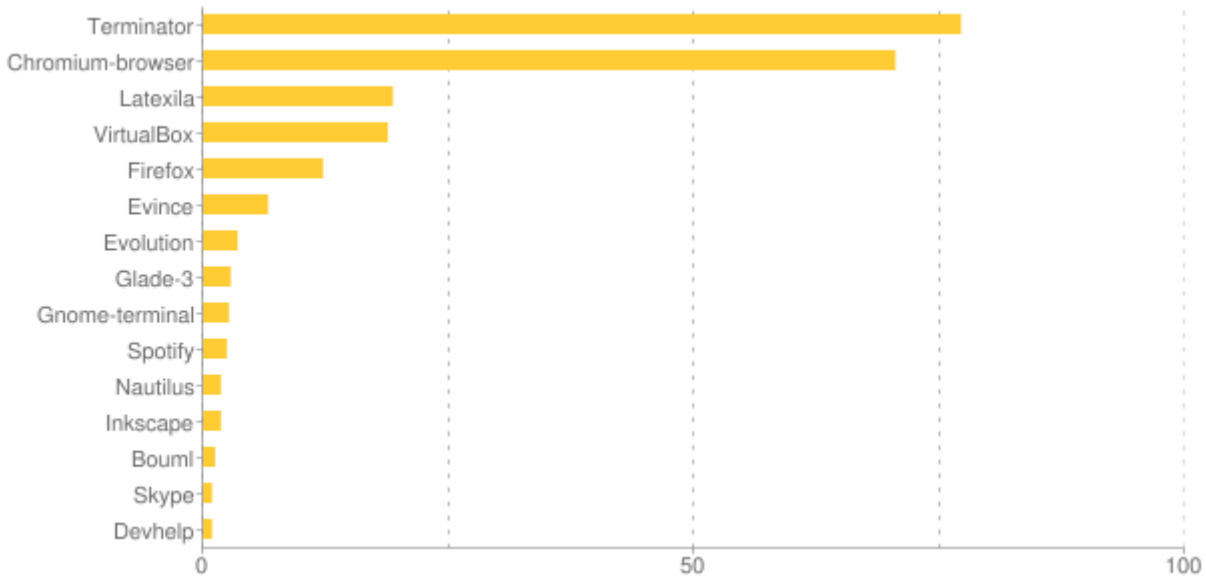


Figura 6.3: Gráfica de aplicaciones más usadas de Dazer (14/03/2011 – 19/06/2011).

Aplicación	Tiempo registrado (horas)
Terminator	77.218
Chromium-browser	70.507
Latexila	19.326
VirtualBox	18.832
Firefox	12.206
Evince	6.640
Evolution	3.520
Glade-3	2.771
Gnome-terminal	2.592
Spotify	2.386
Nautilus	1.826
Inkscape	1.786
Bouml	1.236
Skype	0.923
Devhelp	0.875

Tabla 6.4: Aplicaciones más usadas según Dazer (14/03/2011 – 19/06/2011).

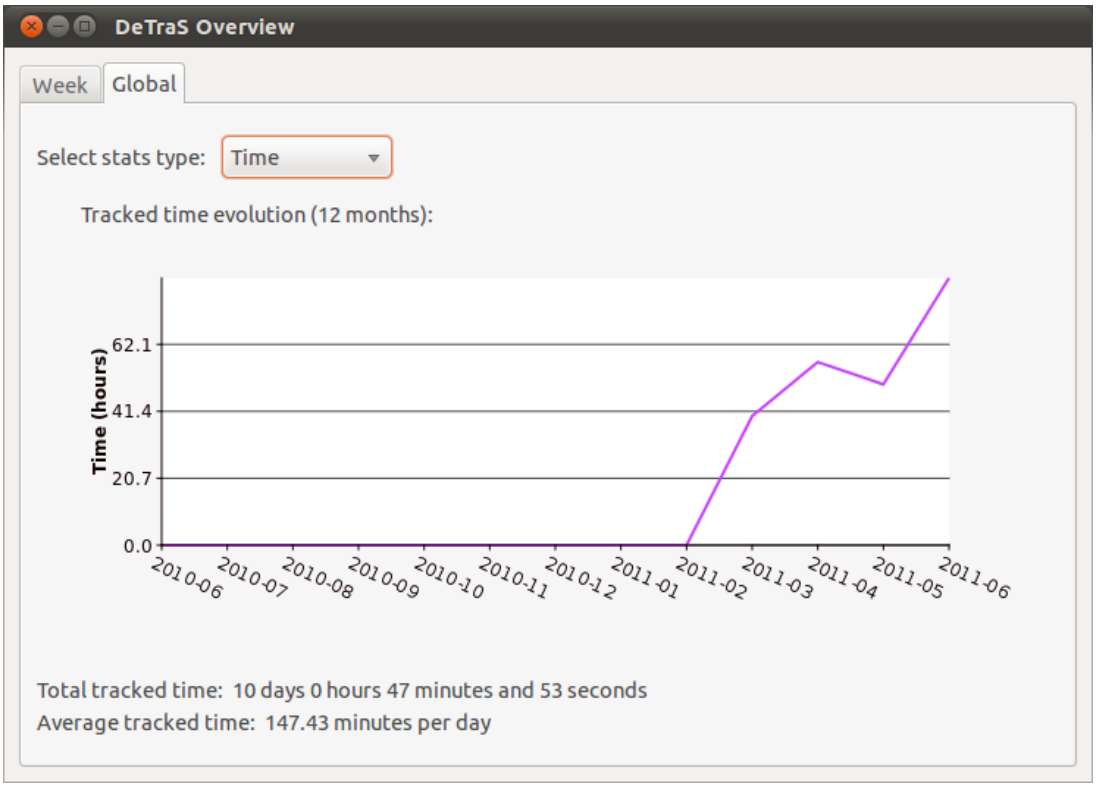


Figura 6.4: Vista de evolución de datos recogidos del *applet* de DeTraS (14/03/2011 – 19/06/2011).

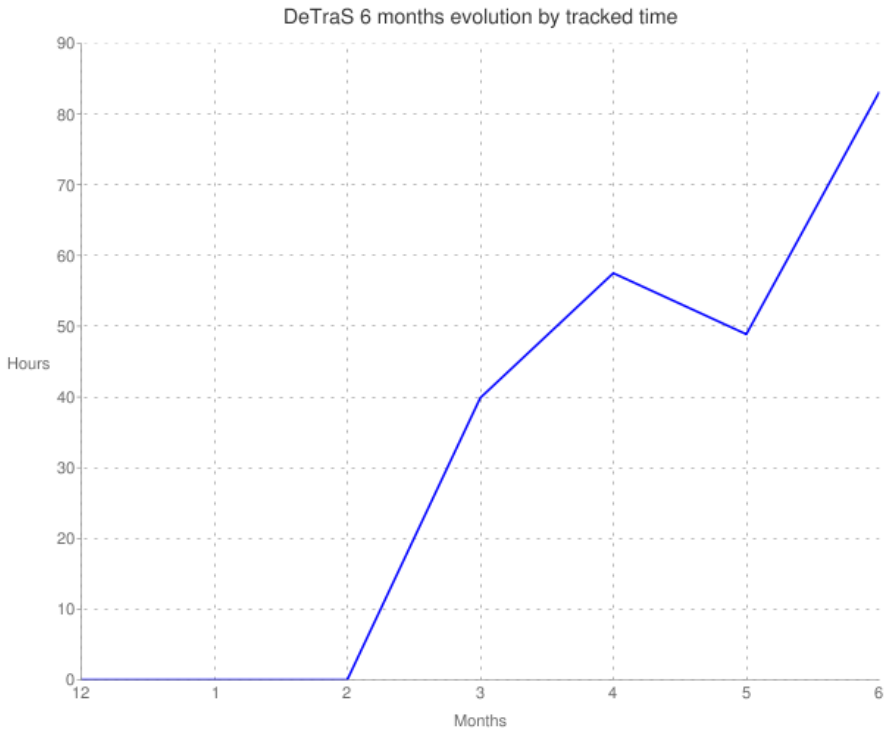


Figura 6.5: Gráfica de evolución de datos recogidos de Dazer (14/03/2011 – 19/06/2011).

6.1.2. CheckApp

En el caso de CheckApp es más complicado encontrar una manera de evaluar el trabajo realizado, ya que no disponemos de ninguna referencia anterior que permita realizar una comparación. En mi opinión, la mejor forma de comprobar el estado de CheckApp es realizar un caso de uso de cierto interés empleando la aplicación desarrollada, presentando capturas de pantalla del proceso. Para llevar a cabo esta prueba, vamos a efectuar el *check–app* de una aplicación sin haber sido identificados en el sistema previamente.

En primer lugar, es necesario acceder a la página de CheckApp. En esta prueba voy a emplear la página que está disponible en línea[1], aunque el funcionamiento en un servidor de prueba sería el mismo. Puede contemplar esta pantalla en la figura 6.6.

Una vez accedemos al portal, se nos redirigirá a nuestro perfil de usuario, tal y como puede observar en la figura 6.7. En esa página se muestra un conjunto de la información que hemos ido obteniendo tras utilizar la aplicación, como los *check–apps*

[1] `http://checkapp.net`

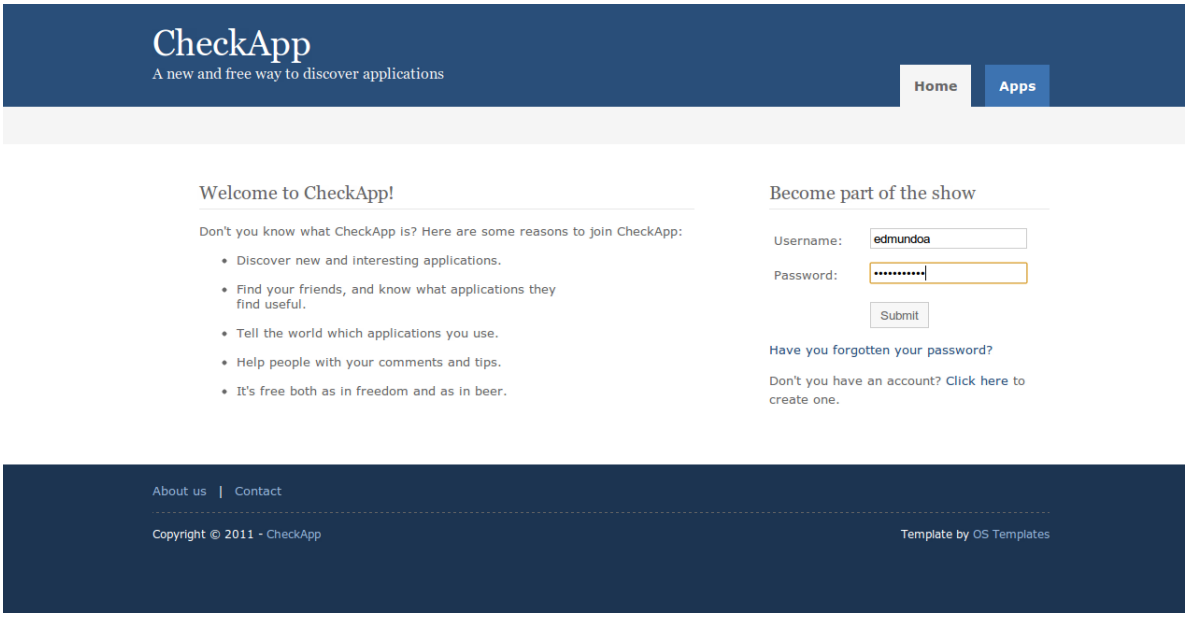


Figura 6.6: Página de identificación de CheckApp.

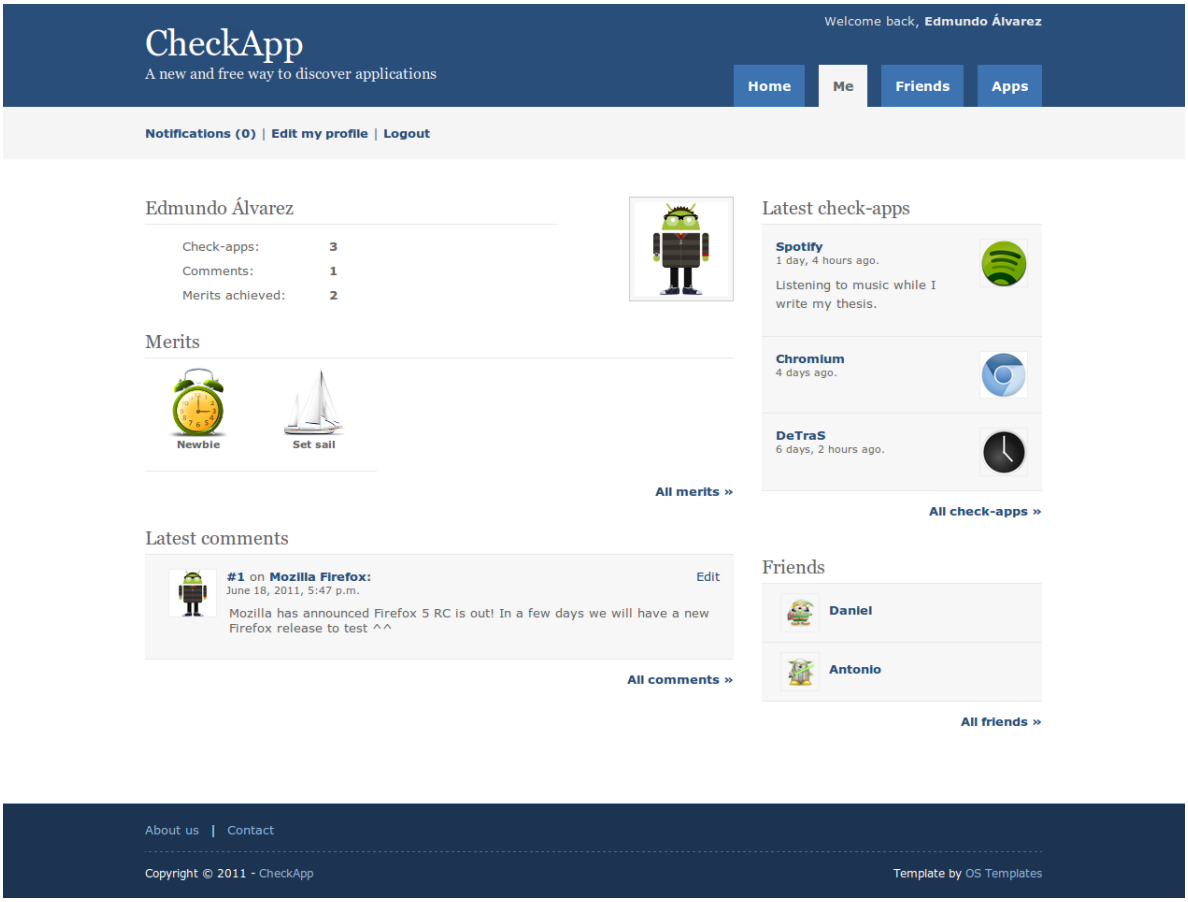


Figura 6.7: Perfil de usuario en CheckApp.

y comentarios realizados, los méritos alcanzados y los amigos que tenemos.

Para efectuar un *check–app* es preciso acceder al apartado de aplicaciones —ver figura 6.8—, donde podemos buscar aplicaciones que aparecen o simplemente seleccionar una de la lista paginada. En este caso, vamos a hacer *check–app* en la propia aplicación CheckApp, ya que estamos utilizándola en estos momentos. Por tanto, visitamos el perfil de la aplicación, que puede contemplar en la figura 6.9.

El perfil de la aplicación es muy similar al del usuario, reuniendo una serie de datos de la misma, así como los comentarios realizados si los hubiese. Como queremos hacer un *check–app*, pulsamos en el botón para ello, que nos redirigirá a una página donde podemos introducir un mensaje y confirmar que queremos continuar. Esta página es mostrada en la figura 6.10.

Una vez hemos confirmado el *check–app*, seremos dirigidos nuevamente a la página de la aplicación, indicando que la operación se ha llevado a cabo correctamente. También es posible apreciar que el contador de *check–apps* se ha incrementado en una unidad —ver figura 6.11—.

6.1.3. Conclusiones

Como habrá podido apreciar tras leer los apartados anteriores, los productos software desarrollados durante el presente proyecto están disponibles públicamente y, tras realizar numerosas pruebas, todo indica que funcionan correctamente. Por tanto, se considera que esta etapa se ha llevado a cabo satisfactoriamente.

6.2. Resultados de distribución

Quisiera destacar, antes de entrar en materia, que los resultados aquí detallados sólo hacen referencia a la distribución de DeTraS, no estando relacionados con el proyecto CheckApp.

6.2.1. Número de descargas

Una de las mediciones que se pueden realizar para darse una idea de la notoriedad adquirida por un proyecto de software libre es contabilizar el número de descargas que se han producido del mismo. La tabla 6.5 muestra el número de descargas realizadas en Launchpad y SourceForge de los distintos archivos puestos a disposición

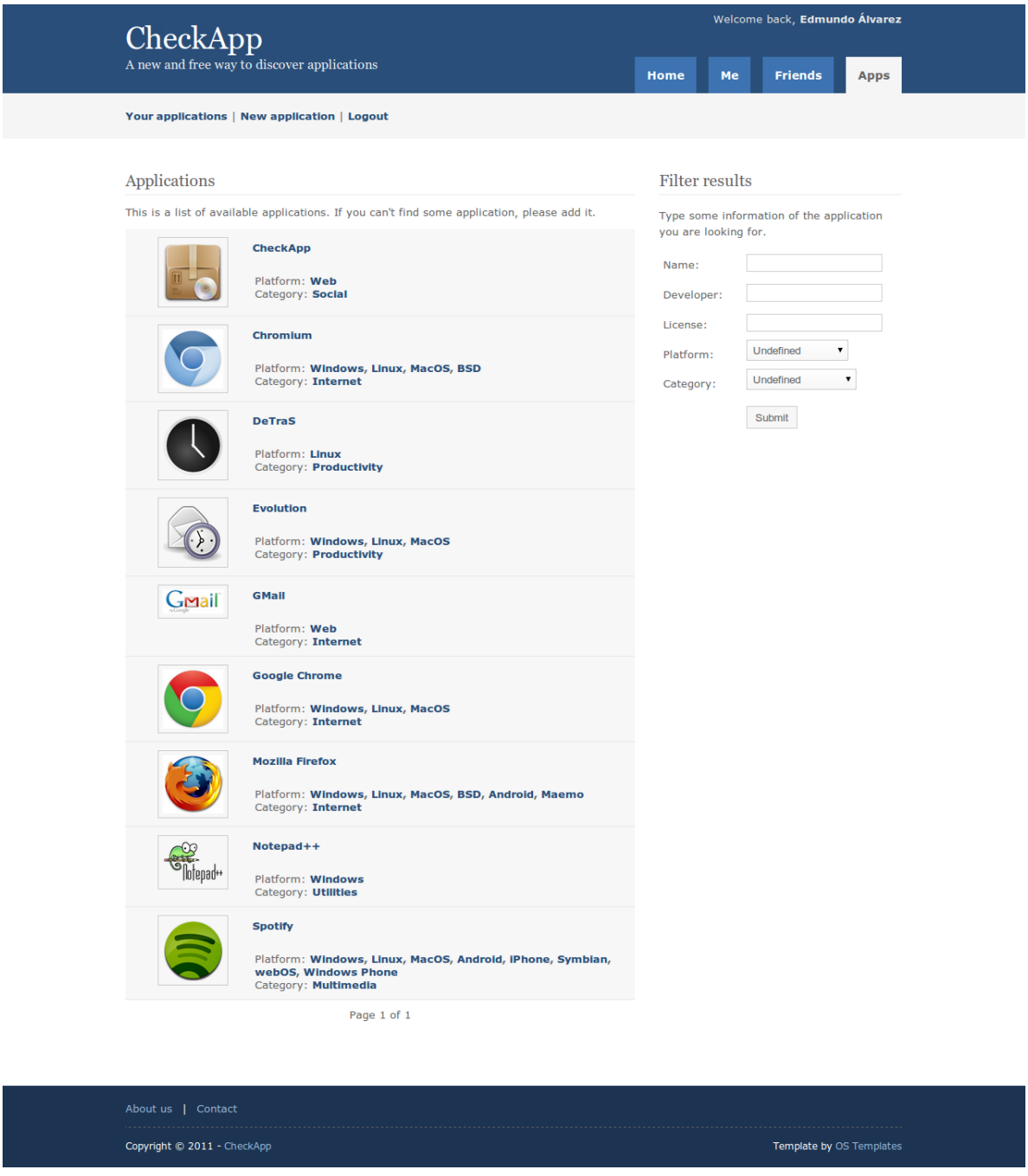


Figura 6.8: Listado de aplicaciones en CheckApp.

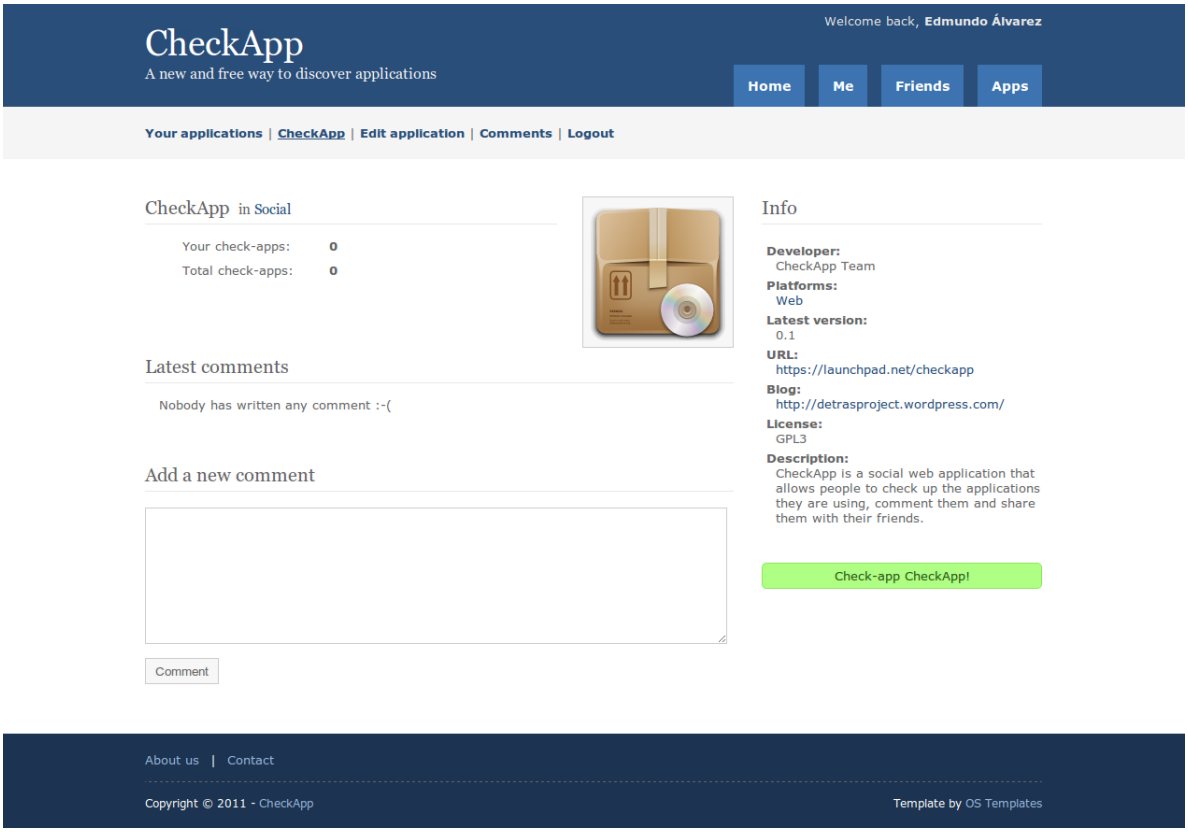


Figura 6.9: Perfil de aplicación en CheckApp.

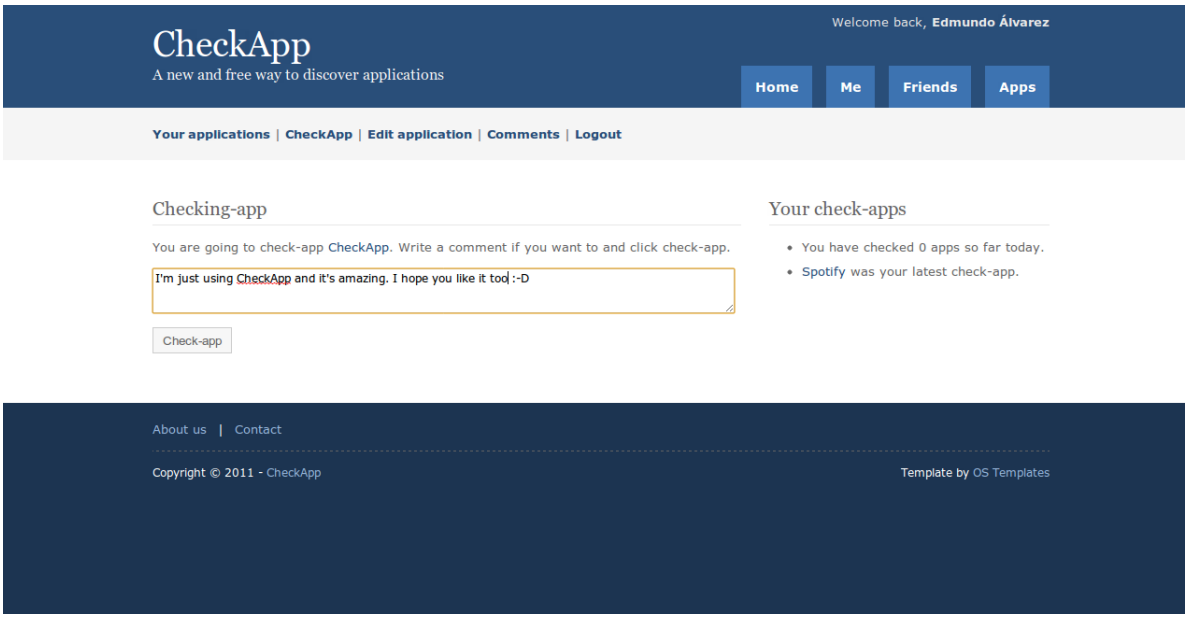


Figura 6.10: Página de confirmación de *check–app* en CheckApp.

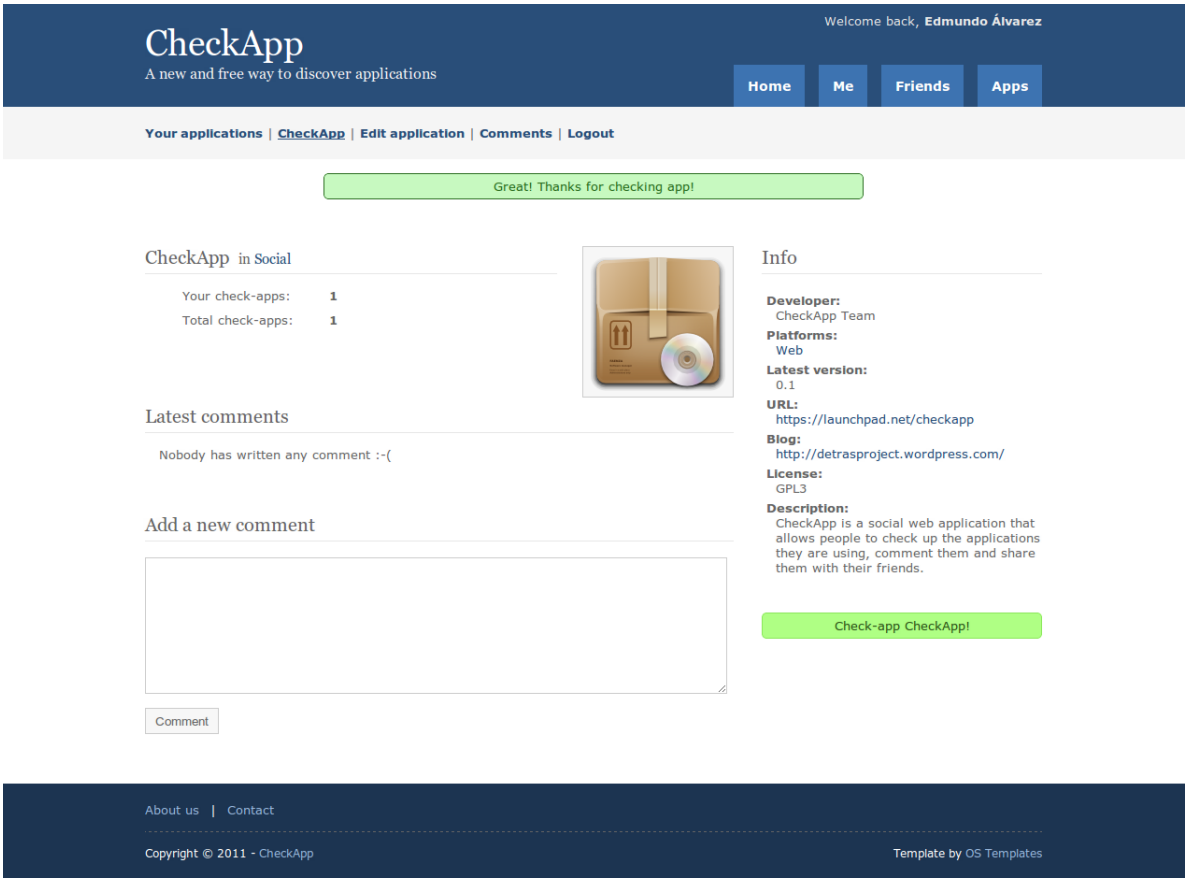


Figura 6.11: Perfil de aplicación tras realizar el *check–app* en CheckApp.

Versión	Fecha de lanzamiento	Número de descargas
0.1.1	19/02/2010	8
0.2	21/03/2010	11
0.3.0	11/04/2010	20
0.3.5	08/05/2010	25
0.4.0	05/06/2010	18
0.4.1	24/06/2010	51
0.4.2	11/12/2010	43
0.5.0	22/04/2011	42

Tabla 6.5: Número de descargas por versión.

de los usuarios por cada versión. Desgraciadamente, no es posible conocer el número de descargas que se han realizado desde el repositorio de paquetes que nos facilita Launchpad, por lo que no han sido incluidas en la tabla mencionada.

Como puede apreciarse en los datos de la tabla 6.5, el número de descargas ha aumentado considerablemente, pasando de las 8 de la versión inicial hasta estabilizarse en torno a unas 40 descargas en las últimas versiones liberadas.

6.2.2. Estadísticas de la bitácora

Desde que se abrió la bitácora en mayo de 2010, se han publicado un total de 19 artículos, es decir, un promedio de casi un artículo y medio por mes.

El total de visitas a la bitácora es de 1768, teniendo un máximo de 154 visitas en un día. El número total de comentarios es de 6, siendo 5 de ellos de páginas que han agregado algún artículo escrito.

6.2.3. Estadísticas de la documentación

Desde la aparición del manual de usuario en la versión 0.3.5 de DeTraS y hasta la creación de la *wiki* del proyecto a finales de abril de 2011, las descargas de documentación han representado en torno al 50 por cierto del total de descargas del proyecto.

Mientras tanto, la página principal de la *wiki* de DeTraS ha sido visitada un total de 250 veces hasta el momento de escribir este texto.

6.2.4. Colaboraciones de usuarios

Como se mencionó en el apartado 6.2.2, sólo se ha producido un comentario en la bitácora por parte de un usuario.

La participación en el desarrollo del proyecto no ha sido mucho mejor. Únicamente dos errores de los detectados han sido reportados por usuarios de la aplicación y sólo un par de usuarios han realizado algún tipo de sugerencia.

El número total de usuarios que han aportado algún dato al servidor de DeTraS desde su puesta en marcha asciende a un total de ocho.

6.2.5. Conclusiones

A la vista de los resultados, tan sólo se puede concluir que la promoción de DeTraS no ha dado los resultados esperados, obteniendo una participación mucho más baja de la que se pretendía.

No es preciso agregar que con la cantidad de datos recogidos no merece la pena llevar a cabo un estudio de comportamiento de usuarios, ya que no sería representativo.

Capítulo 7

Conclusiones

Tras haber descrito las labores realizadas en el presente trabajo, ha llegado el momento de realizar un repaso de los logros alcanzados, así como de extraer conclusiones del proceso descrito en este documento.

7.1. Logros alcanzados

El trabajo aquí detallado ha tenido como parte de sus objetivos el desarrollo de diversas mejoras en DeTraS, que han permitido conseguir corregir diversos defectos detectados, así como mejorar y hacer más útil el sistema.

Como puede comprobarse, se han llevado a cabo todos los objetivos marcados en el capítulo 3 del presente trabajo. Aquellos objetivos que estaban más relacionados con tareas de desarrollo han sido concluidos de forma exitosa, mientras que los que estaban más orientados a la investigación no han corrido la misma suerte. A continuación se presentan los distintos hitos alcanzados en el presente trabajo.

Simplificación de las opciones de privacidad. Se ha diseñado, desarrollado y puesto a disposición de los usuarios una versión mucho más sencilla de las opciones de privacidad. De esta manera, los usuarios podrán proteger su privacidad de una manera más cómoda, clara y útil a la vez.

Mejoras en la presentación de datos locales. Tal y como se ha podido comprobar, se ha desarrollado la funcionalidad para ver los datos recogidos por el sistema DeTraS localmente, complementando esos datos con información y gráficas de utilidad.

Corrección de errores y desarrollo de mejoras en DeTraS. Se han corregido diversos errores y realizado varias mejoras en la aplicación con el fin de ofrecer un producto lo más pulido posible. Pese a la escasa intervención de los usuarios en el proyecto, algunos de estos cambios han sido fruto de sugerencias e informes realizados por ellos.

Desarrollo de aplicación social. Con el fin de experimentar en el campo de las redes sociales, se ha desarrollado una aplicación que permite interactuar a los usuarios entre sí gracias a las aplicaciones. Además, se han propuesto varias vías de negocio que podrían hacer que la aplicación no sólo fuese viable técnicamente —que se ha demostrado que es así—, sino que a la vez podría conseguir obtener una remuneración económica.

Difusión de DeTraS. Se han utilizado diversas plataformas y estrategias para aumentar el número de usuarios de DeTraS. Pese a que algunos datos indican que el número de usuarios de la aplicación ha aumentado, el incremento no ha sido suficiente.

Creación de una comunidad en torno a DeTraS. No se ha logrado el objetivo de implicar a los usuarios de la aplicación en su desarrollo y futuro a pesar de los esfuerzos realizados en ese sentido. Este fracaso está en parte relacionado con el punto anterior.

7.2. Conocimientos adquiridos

El desarrollo del presente trabajo me ha permitido adquirir conocimientos en varios campos, entre los que destaco:

Gestión de un proyecto. El desarrollo de los proyectos presentados me ha permitido comprender lo complicado que es gestionar un proyecto, la gran cantidad de decisiones que hay que tomar ante cualquier problema que surge y la importancia de las mismas en el futuro del proyecto.

Ampliación de conocimientos en Python. Pese a que había desarrollado previamente con Python, en este trabajo he podido obtener una visión más amplia de sus

bibliotecas, emplear herramientas pertenecientes al ecosistema del lenguaje y conocer aspectos de optimización y de sintaxis más idiomáticos.

Ampliación de conocimientos en Django. El desarrollo de CheckApp me ha proporcionado conocimientos en diversos aspectos de Django con los que no había podido trabajar hasta la fecha. Por ejemplo, he utilizado la autenticación y herramientas de administración que proporciona este *framework* y he descubierto la herramienta South para migrar los datos ante cambios producidos en el modelo.

Creación de gráficas usando GTK. El presente trabajo me ha permitido conocer de una forma mucho más detallada la manera en que las aplicaciones de escritorio usan gráficos.

Funcionamiento de notificaciones usando Libnotify, conociendo cómo utilizar una biblioteca ampliamente usada en las aplicaciones de GNU/Linux.

Despliegue de aplicaciones web en un servidor LAMP, adquiriendo conocimientos de instalación y configuración de herramientas como Apache y MySQL entre otros.

Conocimientos de herramientas usadas en el desarrollo. La realización del presente trabajo me ha permitido conocer una gran cantidad de herramientas que se emplean en el desarrollo de software en general, y en el desarrollo de software libre en particular.

7.3. Lecciones aprendidas

Estas son algunas de las lecciones aprendidas en el desarrollo del presente trabajo:

1. La distribución de una aplicación es francamente difícil. Aún más cuando la realiza un desarrollador desconocido y tiene que abrirse paso en un mercado con tanta competitividad como el de las aplicaciones.

2. La primera idea que se hace un potencial usuario sobre una aplicación es fundamental. Aunque se ha indicado continuamente que el objetivo de DeTraS no es espiar ni comprobar cuánto tiempo se trabaja, me queda la sensación de que algunos usuarios han pensado que ese era el objetivo real del sistema realizado, algo que puede haber pesado mucho a la hora de utilizar la aplicación.

3. Las labores de desarrollo y distribución de una aplicación deben mantenerse a lo largo del tiempo. He podido comprobar que los periodos de inactividad en el proyecto, provocados por diversos motivos, han supuesto un lastre a la hora de convencer a los usuarios de descargar y usar el código de la aplicación.

4. Es preciso escoger las tecnologías de desarrollo de un proyecto con mucha cautela. Con el reciente lanzamiento de GNOME 3 se han realizado profundos cambios en el escritorio de este entorno, especialmente en la forma de trabajar con *applets*. Esto ha provocado que el *applet* desarrollado para DeTraS necesite cambios de cierto calado para poder funcionar en la nueva plataforma, lo que ha podido reducir el número de usuarios potenciales de DeTraS. En este caso, no obstante, era complicado evitar el problema, ya que las tecnologías usadas en este nuevo entorno no eran compatibles con las versiones antiguas de GNOME, por lo que, o se desarrollaban y mantenían dos *applets* distintos, o se dejaba a una de las versiones sin soporte. Aún así, se ha aprendido que es mejor invertir tiempo escogiendo con cautela las tecnologías a emplear, que perderlo posteriormente reescribiendo código.

5. Si convencer a un usuario para que utilice tu aplicación es complicado, hacerlo para que colabore contigo es aún más difícil. La escasa participación de usuarios en el proyecto así lo demuestra.

6. Para alcanzar un buen número de usuarios de una aplicación en GNU/Linux hay que llegar a los repositorios oficiales. Puede que esto no sea así en todos los casos, pero muchos usuarios prefieren confiar en el entorno controlado que les facilitan los repositorios de su distribución, evitando instalar programas externos.

7.4. Trabajos futuros

A continuación, se ofrece una lista de líneas de desarrollo e investigación que permitirían extender y mejorar el trabajo aquí realizado.

1. Llevar a cabo mejoras y ampliaciones en CheckApp. Estas mejoras pueden ser muy variadas, siendo algunas de ellas:

a) Utilizar JavaScript y AJAX para mejorar la experiencia de usuario en formularios y ciertas páginas como la de notificaciones.

b) Permitir a los usuarios identificarse utilizando OpenId u otros servicios similares que ofrecen Google, Facebook y Twitter, entre otros.

c) Asociar cuentas de otras redes sociales, para permitir propagar las acciones realizadas en CheckApp a las otras redes.

d) Dado el carácter distribuido de CheckApp, podría desarrollarse un protocolo de comunicación entre instancias de la misma utilizando, por ejemplo, XMPP.

e) Crear una versión móvil de la página, así como aplicaciones para las distintas plataformas móviles existentes.

f) Ofrecer una API para que desarrolladores externos puedan utilizar CheckApp.

g) Mejorar la presentación de comentarios, permitiendo agregar ciertos elementos HTML y mostrar dinámicamente comentarios mencionados en el texto.

2. Conseguir que el *applet* de escritorio soporte el nuevo GNOME 3, así como otras plataformas, como pueden ser Unity o KDE.

3. Integrar el *applet* de DeTraS con CheckApp, permitiendo hacer un *check–app* de una aplicación directamente desde el escritorio.

4. Ofrecer un servicio que permita a otras aplicaciones emplear ciertos datos recogidos por DeTraS posibilitando, por ejemplo, desarrollar un menú de aplicaciones que ordene sus resultados en función de las aplicaciones más utilizadas por el usuario.

5. Buscar otras alternativas para conseguir que DeTraS obtenga un mayor número de usuarios. La mejor opción para ello sería trabajar para incluir el sistema en los repositorios oficiales de alguna distribución, aunque también podrían redactarse textos para leer en conferencias de software libre y buscar otras alternativas para dar a conocer la aplicación.

Apéndice A

DeTraS 0.5.0 User Manual

About this document

A.1. Introduction

DeTraS (Development Tracking System) is a set of tools to track developers activities by registering its applications' usage. It can also be used as a personal time manager, allowing you to know how much time you spend performing tasks on the computer. As a lot of networking applications, it has two parts: a client (part of the application that runs on your computer) and a server (usually runs in a remote computer). This manual describes utils under client side of DeTraS (so I will use DeTraS referring to client side of DeTraS from now on). DeTraS includes these utilities:

TempusFugit. It is a tool to track your applications' usage under X11 Window.

Squealer. A tool to filter and upload your data to a machine running DeTraS server.

DeTraS CLI. It is a Command Line Interface to view your events and statistics from a shell.

DeTraS applet. A Gnome applet that allows you to configure and use DeTraS' tools more easily.

You can download a PDF version of this document here: `http://launchpad.net/detras/0.5/0.5.0/+download/user_manual.pdf`.

A.2. Getting started

A.2.1. Installation

In order to use DeTraS, you have to download it from this page[1] and install it on your system. You can find more information about installing DeTraS on Installation guide[2].

A.2.2. What information tracks DeTraS?

DeTraS will save the following information about your system's usage:

- Application name and window title of the current window focused. It will also take the time when a window gains focus.
- Time when your session becomes active or inactive.
- Your name or nickname (optional).
- Projects you are working on (optional).

A.2.3. What information uploads DeTraS?

DeTraS does not upload any information if you don't do it manually. In any case, you can select a privacy level to control uploaded information:

Weak. All information detailed in section A.2.2 section is uploaded.

Medium. This option will upload the same information as weak level, but it will filter tracked window titles by using as many regular expressions as you want.

Strong. Using this option DeTraS will only upload tracked application names.

[1] `https://launchpad.net/detras/+download`

[2] `http://sourceforge.net/apps/mediawiki/detras/index.php?title=Client_installation_guide`

A.2.4. Problems, questions or bugs

I would really appreciate if you take the time to report any problem, question or bug you may find using or installing DeTraS. In first place, you should look in our FAQ[3] or in out Launchpad page[4] to see if your question has already been answered. If you don't find it in DeTraS documentation, you can report your problems or questions here[5] and your bugs here[6].

Alternatively, you can also write me on Twitter[7] or send an E-Mail to `e.alvarezj@gmail.com`.

A.3. TempusFugit

You can interact with TempusFugit by using a command line interface or DeTraS applet.

A.3.1. Run TempusFugit on a shell

To execute TempusFugit on a shell, you have to run `tempusfugit` command. By default, it will store all tracked events on a database called `detras.db` under `\$HOME/.detras` folder.

As you will see, TempusFugit run as a daemon, so you have to send a `SIGTERM` signal (Ctrl+C) to stop its execution. Be sure you do not send a SIGKILL signal because it can break events database.

To see more details about TempusFugit execution please check `tempusfugit` manual page.

A.3.2. Run TempusFugit using DeTraS applet

Please refer to A.7.3 section.

[3] `http://sourceforge.net/apps/mediawiki/detras/index.php?title=FAQ`
[4] `https://launchpad.net/detras`
[5] `https://answers.launchpad.net/detras`
[6] `https://bugs.launchpad.net/detras`
[7] `http://twitter.com/detrasproject`

A.4. Squealer

Squealer can be executed to upload tracked information from a shell or from DeTraS applet.

A.4.1. Run Squealer on a shell

To execute Squealer from a terminal, you must execute `squealer` command. It will apply any filter defined on your preferences and will send data to the server of your choice. By default, Squealer will not show any output unless it finds a problem.

You can find more details on `squealer` manpage.

A.4.2. Run Squealer from DeTraS applet

Please refer to A.7.4 section.

A.5. Preferences

DeTraS stores your preferences in a file called `detras.ini` under `\$HOME/.detras` folder. If this file does not exist, DeTraS will get its preferences from `/usr/share/detras/detras.ini`, which contains system default preferences.

Take into account that you can configure your preferences more easily using DeTraS applet. Anyway, if you want to modify these files manually, you can read `detras.ini` manpage to get more information.

A.6. DeTraS CLI

You can execute DeTraS CLI (Command Line Interface) by typing `detras-cli` on a shell.

This tool offers you a view of your tracked information in two different ways: showing a list of tracked events on the current week (passing `events` action), or showing some statistics generated using your data (giving `stats` action).

For more information, please read `detras-cli` manual page.

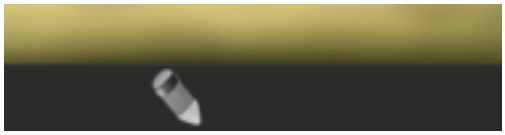


Figura A.1: DeTraS applet icon (tracking off).

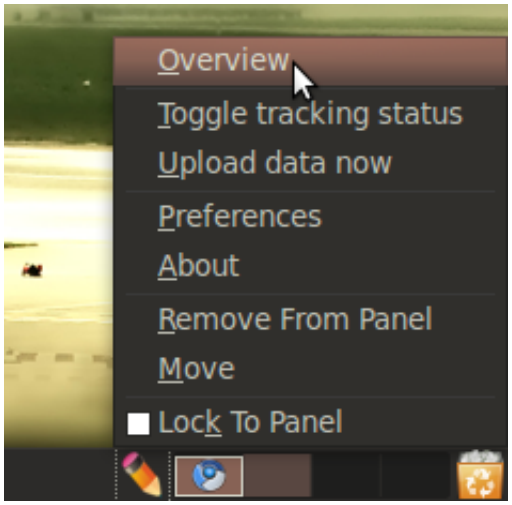


Figura A.2: DeTraS applet menu.

A.7. Using DeTraS applet

A.7.1. Adding DeTraS applet to panel

To add DeTraS applet to panel, right-click on the panel and choose *Add to panel*. Then select DeTraS applet from applets list and click on *Add* button. You should see a pencil icon in the panel that represents DeTraS applet.

A.7.2. DeTraS applet menu

You can perform many actions right clicking on DeTraS applet. As you can see in figure number A.2, it allows you to see an overview of events tracked, toggle tracking status, upload data and change your preferences.

A.7.3. Work with TempusFugit

You can start or stop TempusFugit in two ways: by clicking on DeTraS applet or by choosing *Toggle tracking status* from DeTraS applet menu. DeTraS applet icon indicates you TempusFugit's status. A grey icon implies that TempusFugit is stopped, so it is not tracking your data. A coloured icon indicates that TempusFugit is tracking your data.

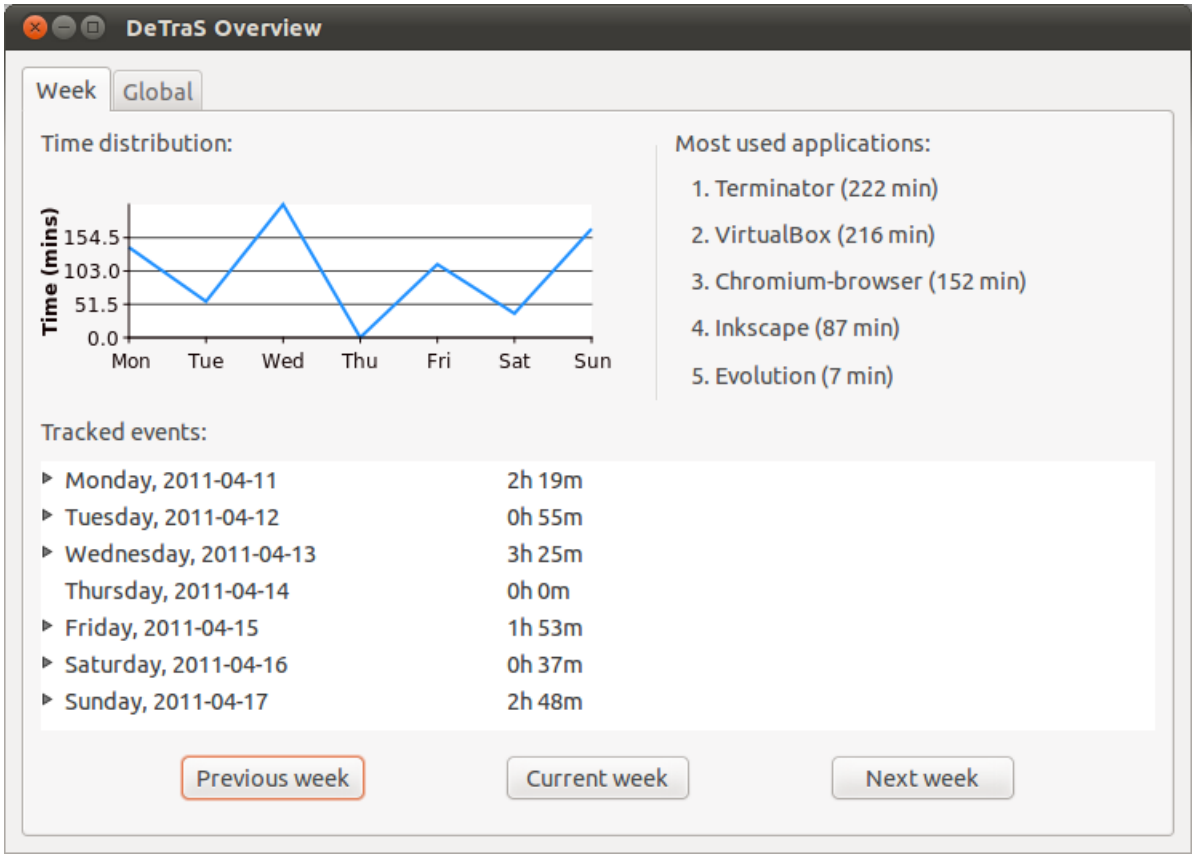


Figura A.3: DeTraS overview – week view.

DeTraS will also show a Libnotify message every time you start or stop TempusFugit, to give you feedback of performed action.

A.7.4. Work with Squealer

To upload data tracked by TempusFugit you have to select *Upload data now* option from DeTraS applet menu. DeTraS applet will show up a message to inform you if the operation was performed successfully or not.

A.7.5. Overview window

Selecting *Overview* option on DeTraS applet menu you will see a window with events tracked on your system and some useful information about them.

Week view

Once you open DeTraS overview, you'll see week view. As shown in A.3 figure, in that window you can see events tracked in a week, total tracked time per day and most

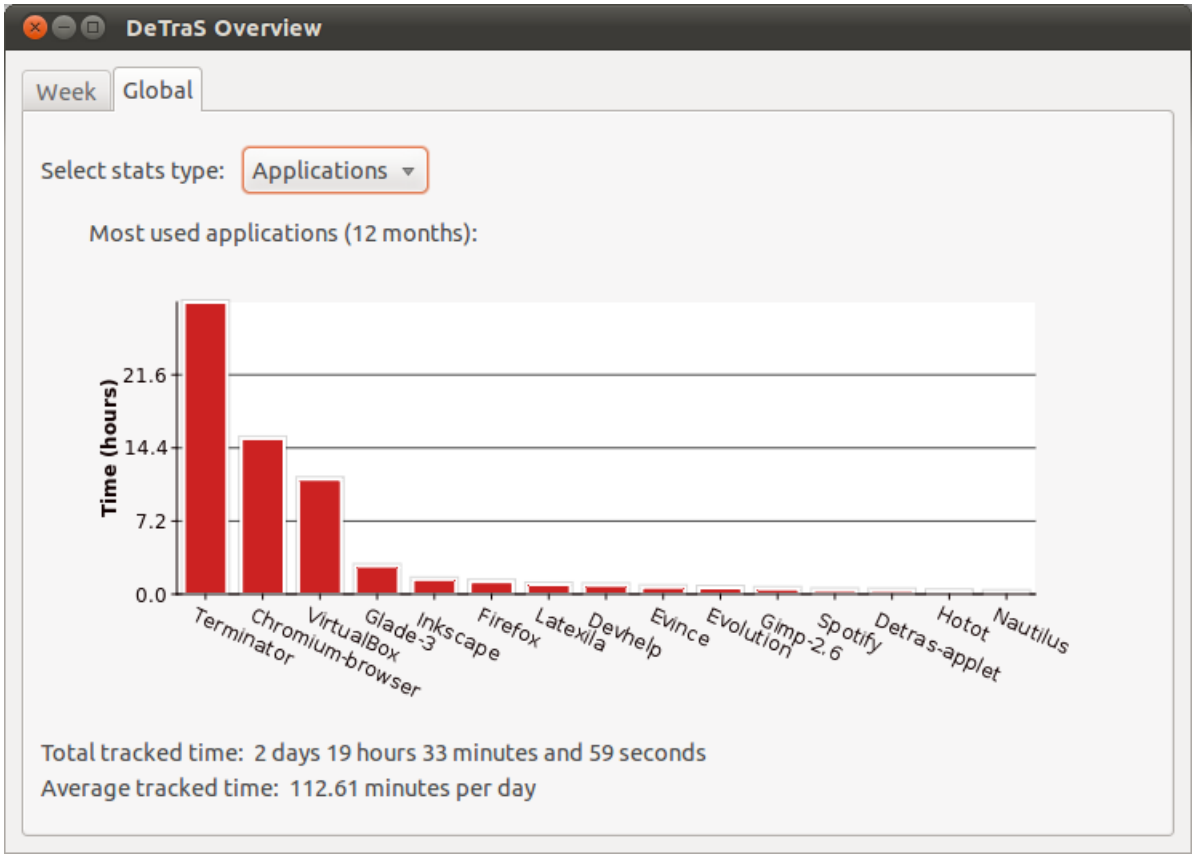


Figura A.4: DeTraS overview — global view.

used applications in that week.

You can move among weeks by clicking on *Previous week* or *Next week*. To return to current week, you can click on *Current week* button.

Global view

Overview window allows you to check events in a wider period of time. Among more information, this screen basically offers you a graph of most used applications on the lastest twelve months or monthly time distribution of your work from twelve months ago.

A.7.6. Preferences dialog

When you click on *Preferences* item on DeTraS applet menu, you can edit DeTraS preferences in an easier way than editing `\$HOME/.detras/detras.ini` file manually.

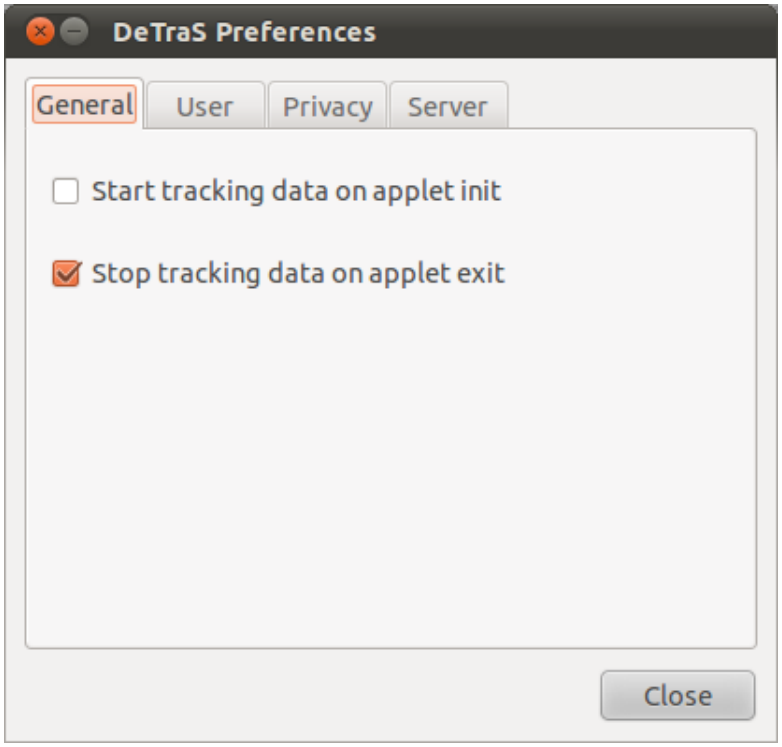


Figura A.5: DeTraS general preferences.

General tab

Here you can configure general preferences:

Start tracking data on applet init. If you check this box, TempusFugit will start automatically to track your activity when the applet is loaded. (Default: disabled).

Stop tracking data on applet exit. If you check this box, TempusFugit will stop tracking your activity automatically when the applet is unloaded. (Default: enabled).

User tab

This window allows you to change your personal information:

User name. Type your name or nickname on this text box.

Projects you are working on. Configure projects you are working on. Click on *Add* button to add a new project, select a project and click *Delete* button to remove a project, or double-click on a project to edit it.

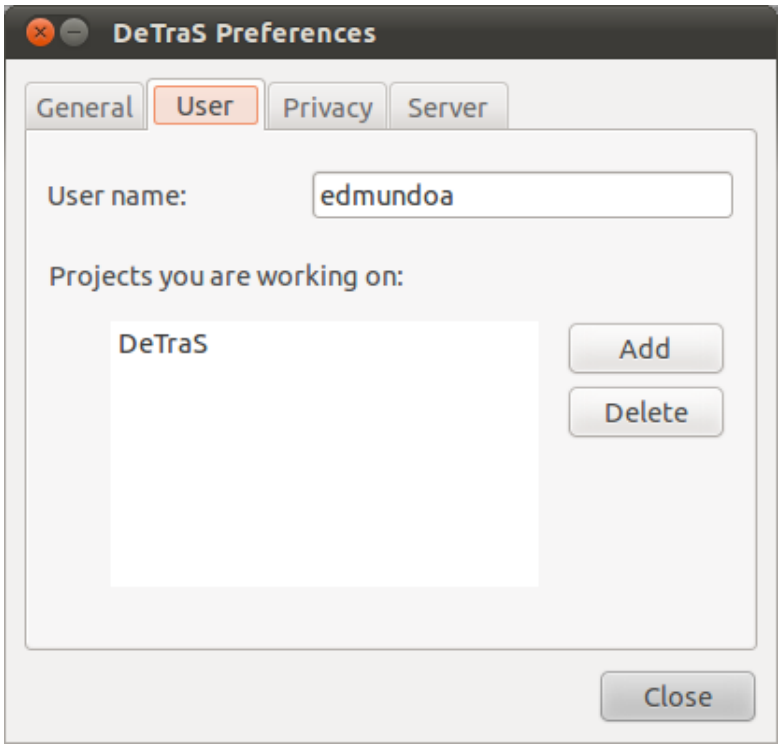


Figura A.6: DeTraS user preferences.

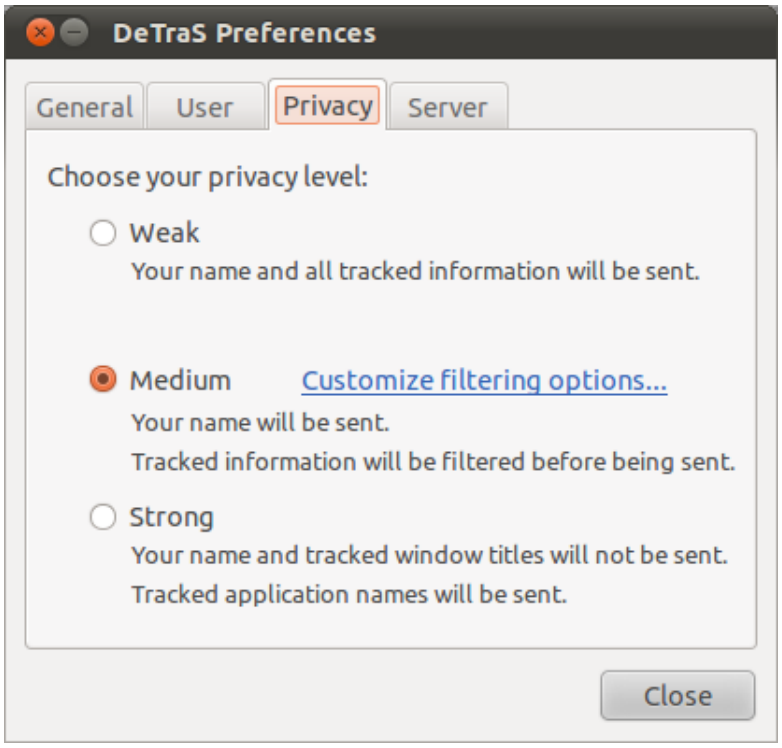


Figura A.7: DeTraS privacy preferences.

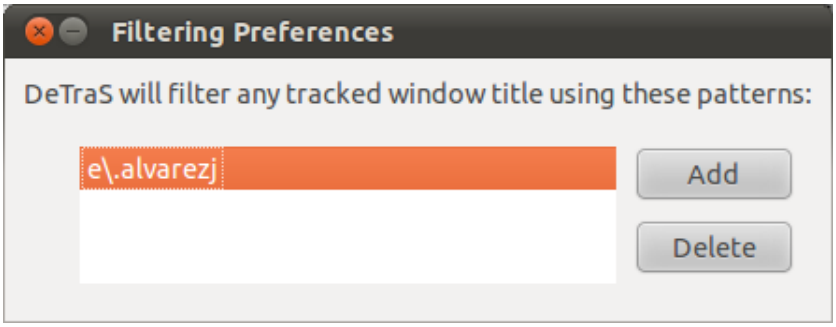


Figura A.8: DeTraS filtering preferences.

Privacy tab

Please take into account that DeTraS will not send any information unless manually click on *Upload data now* or execute Squealer by yourself. Anyway, you can choose among some filtering levels to maintain your privacy. These options are available:

Weak. Setting this level, your name (or nickname) and all tracked information will be uploaded without any filtering.

Medium. By choosing this level, your name will be sent but you can set filters to apply to tracked information.

Customize filtering options. Clicking on this link a popup will show up. There you can manage your filtering regular expressions. If you need help to build regular expressions, you can take a look to this guide[8].

Strong. Select this level if you don't want to send your name or any window title tracked on your system.

Server tab

Configure server parameters from this window. By default, this window contains parameters of a server that is tracking data to help me doing my thesis. I really appreciate if you send your data there.

Address. Type server URL. (Default: https://193.147.51.204).

Port. Type port where the server is listening. (Default: 8080).

[8] `http://docs.python.org/library/re.html/#regular-expression-syntax`

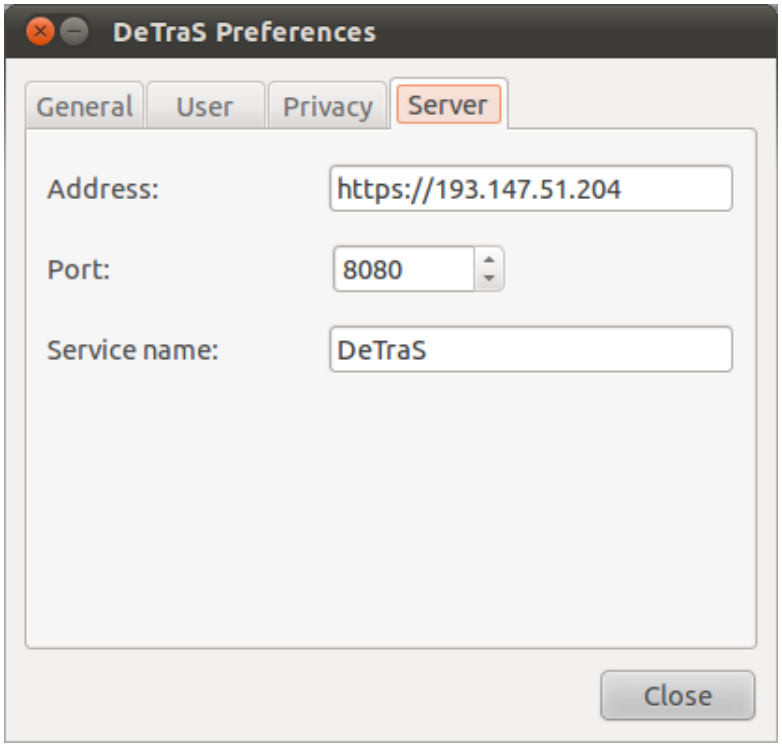


Figura A.9: DeTraS server preferences.

Service name. Connection service name. Please don't change this preference unless you know what you are doing. (Default: DeTraS).

A.8. Screencast

Here[9] you can watch a screencast to take a quick view of DeTraS installation and its applet.

[9] https://www.youtube.com/watch?v=MrViyy8Kzf4

Apéndice B

Contenido del disco compacto

B.1. Licencia del código fuente

El código fuente desarrollado en este trabajo se distribuye bajo la licencia pública GPL. En el fichero `licencia.txt` dentro de la carpeta `/codigo/` del disco compacto adjunto a este trabajo se encuentra el fichero con la licencia completa utilizada. Asimismo, en todos los ficheros de código incluidos puede encontrarse el preámbulo de esta licencia:

```
This program is free software: you can redistribute it and/or modify
it under the terms of the GNU General Public License as published by
the Free Software Foundation, either version 3 of the License, or
(at your option) any later version.

This program is distributed in the hope that it will be useful,
but WITHOUT ANY WARRANTY; without even the implied warranty of
MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
GNU General Public License for more details.

You should have received a copy of the GNU General Public License
along with this program.  If not, see <http://www.gnu.org/licenses/>.
```

B.2. Contenido del disco compacto

El disco compacto adjunto a este documento incluye todo el código fuente desarrollado para el trabajo, los paquetes creados para Debian, parte de la documentación creada, así como una versión de este documento —incluyendo sus fuentes en LaTeX—.

B.2.1. Código fuente del trabajo

El código fuente de DeTraS y CheckApp se incluye en el directorio `/codigo/` del disco compacto. Allí se encuentran todos los ficheros necesarios para su compilación y ejecución.

B.2.2. Paquetes de instalación

Los paquetes Debian y RPM de DeTraS para las arquitecturas *x86* y *amd64* se encuentran en el directorio `/binarios/` del disco.

B.2.3. Documentación

La documentación de la versión de DeTraS incluida en el CD está disponible en la carpeta `/documentacion/`.

B.2.4. Memoria del proyecto

Este documento, junto a sus fuentes en LaTeX, se incluyen en el directorio `/memoria/`.

Apéndice C

Licencia de este documento

LA OBRA O LA PRESTACIÓN (SEGÚN SE DEFINEN MÁS ADELANTE) SE PROPORCIONA BAJO LOS TÉRMINOS DE ESTA LICENCIA PÚBLICA DE CREATIVE COMMONS (**CCPL O LICENCIA**). LA OBRA O LA PRESTACIÓN SE ENCUENTRA PROTEGIDA POR LA LEY ESPAÑOLA DE PROPIEDAD INTELECTUAL Y/O CUALESQUIERA OTRAS NORMAS QUE RESULTEN DE APLICACIÓN. QUEDA PROHIBIDO CUALQUIER USO DE LA OBRA O PRESTACIÓN DIFERENTE A LO AUTORIZADO BAJO ESTA LICENCIA O LO DISPUESTO EN LA LEY DE PROPIEDAD INTELECTUAL.

MEDIANTE EL EJERCICIO DE CUALQUIER DERECHO SOBRE LA OBRA O LA PRESTACIÓN, USTED ACEPTA Y CONSIENTE LAS LIMITACIONES Y OBLIGACIONES DE ESTA LICENCIA, SIN PERJUICIO DE LA NECESIDAD DE CONSENTIMIENTO EXPRESO EN CASO DE VIOLACIÓN PREVIA DE LOS TÉRMINOS DE LA MISMA. EL LICENCIADOR LE CONCEDE LOS DERECHOS CONTENIDOS EN ESTA LICENCIA, SIEMPRE QUE USTED ACEPTE LOS PRESENTES TÉRMINOS Y CONDICIONES.

1. **Definiciones**

   a) La **obra** es la creación literaria, artística o científica ofrecida bajo los términos de esta licencia.

   b) En esta licencia se considera una **prestación** cualquier interpretación, ejecución, fonograma, grabación audiovisual, emisión o transmisión, mera fotografía u otros objetos protegidos por la legislación de propiedad intelectual vigente aplicable.

c) La aplicación de esta licencia a una **colección** (definida más adelante) afectará únicamente a su estructura en cuanto forma de expresión de la selección o disposición de sus contenidos, no siendo extensiva a éstos. En este caso la colección tendrá la consideración de obra a efectos de esta licencia.

d) El **titular originario** es:

1) En el caso de una obra literaria, artística o científica, la persona natural o grupo de personas que creó la obra.
2) En el caso de una obra colectiva, la persona que la edite y divulgue bajo su nombre, salvo pacto contrario.
3) En el caso de una interpretación o ejecución, el actor, cantante, músico, o cualquier otra persona que represente, cante, lea, recite, interprete o ejecute en cualquier forma una obra.
4) En el caso de un fonograma, el productor fonográfico, es decir, la persona natural o jurídica bajo cuya iniciativa y responsabilidad se realiza por primera vez una fijación exclusivamente sonora de la ejecución de una obra o de otros sonidos.
5) En el caso de una grabación audiovisual, el productor de la grabación, es decir, la persona natural o jurídica que tenga la iniciativa y asuma la responsabilidad de las fijaciones de un plano o secuencia de imágenes, con o sin sonido.
6) En el caso de una emisión o una transmisión, la entidad de radiodifusión.
7) En el caso de una mera fotografía, aquella persona que la haya realizado.
8) En el caso de otros objetos protegidos por la legislación de propiedad intelectual vigente, la persona que ésta señale.

e) Se considerarán **obras derivadas** aquellas obras creadas a partir de la licenciada, como por ejemplo: las traducciones y adaptaciones; las revisiones, actualizaciones y anotaciones; los compendios, resúmenes y extractos; los arreglos musicales y, en general, cualesquiera transformaciones de una obra literaria, artística o científica. Para evitar la duda, si la obra consiste en una composición musical o grabación de sonidos, la sincronización temporal de

la obra con una imagen en movimiento (synching) será considerada como una obra derivada a efectos de esta licencia.

f) Tendrán la consideración de **colecciones** la recopilación de obras ajenas, de datos o de otros elementos independientes como las antologías y las bases de datos que por la selección o disposición de sus contenidos constituyan creaciones intelectuales. La mera incorporación de una obra en una colección no dará lugar a una derivada a efectos de esta licencia.

g) El **licenciador** es la persona o la entidad que ofrece la obra o prestación bajo los términos de esta licencia y le concede los derechos de explotación de la misma conforme a lo dispuesto en ella.

h) **Usted** es la persona o la entidad que ejercita los derechos concedidos mediante esta licencia y que no ha violado previamente los términos de la misma con respecto a la obra o la prestación, o que ha recibido el permiso expreso del licenciador de ejercitar los derechos concedidos mediante esta licencia a pesar de una violación anterior.

i) La **transformación** de una obra comprende su traducción, adaptación y cualquier otra modificación en su forma de la que se derive una obra diferente. La creación resultante de la transformación de una obra tendrá la consideración de obra derivada.

j) Se entiende por **reproducción** la fijación directa o indirecta, provisional o permanente, por cualquier medio y en cualquier forma, de toda la obra o la prestación o de parte de ella, que permita su comunicación o la obtención de copias.

k) Se entiende por **distribución** la puesta a disposición del público del original o de las copias de la obra o la prestación, en un soporte tangible, mediante su venta, alquiler, préstamo o de cualquier otra forma.

l) Se entiende por **comunicación pública** todo acto por el cual una pluralidad de personas, que no pertenezcan al ámbito doméstico de quien la lleva a cabo, pueda tener acceso a la obra o la prestación sin previa distribución de ejemplares a cada una de ellas. Se considera comunicación pública la puesta a disposición del público de obras o prestaciones por procedimientos alám-

bricos o inalámbricos, de tal forma que cualquier persona pueda acceder a ellas desde el lugar y en el momento que elija.

m) La **explotación** de la obra o la prestación comprende la reproducción, la distribución, la comunicación pública y, en su caso, la transformación.

n) Los **elementos de la licencia** son las características principales de la licencia según la selección efectuada por el licenciador e indicadas en el título de esta licencia: Reconocimiento, CompartirIgual.

ñ) Una **licencia equivalente** es:

1) Una versión posterior de esta licencia de Creative Commons con los mismos elementos de licencia.
2) La misma versión o una versión posterior de esta licencia de cualquier otra jurisdicción reconocida por Creative Commons con los mismos elementos de la licencia (ejemplo: Reconocimiento-CompartirIgual 3.0 Japón).
3) La misma versión o una versión posterior de la licencia de Creative Commons no adaptada a ninguna jurisdicción (Unported) con los mismos elementos de la licencia.
4) Una de las licencias compatibles que aparece en `http://creativecommons.org/compatiblelicenses` y que ha sido aprobada por Creative Commons como esencialmente equivalente a esta licencia porque, como mínimo:

a′ Contiene términos con el mismo propósito, el mismo significado y el mismo efecto que los elementos de esta licencia.

b′ Permite explícitamente que las obras derivadas de obras sujetas a ella puedan ser distribuidas mediante esta licencia, la licencia de Creative Commons no adaptada a ninguna jurisdicción (Unported) o una licencia de cualquier otra jurisdicción reconocida por Creative Commons, con sus mismos elementos de licencia.

2. **Límites de los derechos**. Nada en esta licencia pretende reducir o restringir cualesquiera límites legales de los derechos exclusivos del titular de los derechos de propiedad intelectual de acuerdo con la Ley de propiedad intelectual o cuales-

quiera otras leyes aplicables, ya sean derivados de usos legítimos, tales como la copia privada o la cita, u otras limitaciones como la resultante de la primera venta de ejemplares (agotamiento).

3. **Concesión de licencia**. Conforme a los términos y a las condiciones de esta licencia, el licenciador concede, por el plazo de protección de los derechos de propiedad intelectual y a título gratuito, una licencia de ámbito mundial no exclusiva que incluye los derechos siguientes:

   a) Derecho de reproducción, distribución y comunicación pública de la obra o la prestación.

   b) Derecho a incorporar la obra o la prestación en una o más colecciones.

   c) Derecho de reproducción, distribución y comunicación pública de la obra o la prestación lícitamente incorporada en una colección.

   d) Derecho de transformación de la obra para crear una obra derivada siempre y cuando se incluya en ésta una indicación de la transformación o modificación efectuada.

   e) Derecho de reproducción, distribución y comunicación pública de obras derivadas creadas a partir de la obra licenciada.

   f) Derecho a extraer y reutilizar la obra o la prestación de una base de datos.

   g) Para evitar cualquier duda, el titular originario:

      1) Conserva el derecho a percibir las remuneraciones o compensaciones previstas por actos de explotación de la obra o prestación, calificadas por la ley como irrenunciables e inalienables y sujetas a gestión colectiva obligatoria.

      2) Renuncia al derecho exclusivo a percibir, tanto individualmente como mediante una entidad de gestión colectiva de derechos, cualquier remuneración derivada de actos de explotación de la obra o prestación que usted realice.

Estos derechos se pueden ejercitar en todos los medios y formatos, tangibles o intangibles, conocidos en el momento de la concesión de esta licencia. Los derechos mencionados incluyen el derecho a efectuar las modificaciones que sean precisas

técnicamente para el ejercicio de los derechos en otros medios y formatos. Todos los derechos no concedidos expresamente por el licenciador quedan reservados, incluyendo, a título enunciativo pero no limitativo, los derechos morales irrenunciables reconocidos por la ley aplicable. En la medida en que el licenciador ostente derechos exclusivos previstos por la ley nacional vigente que implementa la directiva europea en materia de derecho sui generis sobre bases de datos, renuncia expresamente a dichos derechos exclusivos.

4. **Restricciones**. La concesión de derechos que supone esta licencia se encuentra sujeta y limitada a las restricciones siguientes:

   a) Usted puede reproducir, distribuir o comunicar públicamente la obra o prestación solamente bajo los términos de esta licencia y debe incluir una copia de la misma, o su Identificador Uniforme de Recurso (URI). Usted no puede ofrecer o imponer ninguna condición sobre la obra o prestación que altere o restrinja los términos de esta licencia o el ejercicio de sus derechos por parte de los concesionarios de la misma. Usted no puede sublicenciar la obra o prestación. Usted debe mantener intactos todos los avisos que se refieran a esta licencia y a la ausencia de garantías. Usted no puede reproducir, distribuir o comunicar públicamente la obra o prestación con medidas tecnológicas que controlen el acceso o el uso de una manera contraria a los términos de esta licencia. Esta sección 4.a también afecta a la obra o prestación incorporada en una colección, pero ello no implica que ésta en su conjunto quede automáticamente o deba quedar sujeta a los términos de la misma. En el caso que le sea requerido, previa comunicación del licenciador, si usted incorpora la obra en una colección y/o crea una obra derivada, deberá quitar cualquier crédito requerido en el apartado 4.c, en la medida de lo posible.

   b) Usted puede distribuir o comunicar públicamente una obra derivada en el sentido de esta licencia solamente bajo los términos de la misma u otra licencia equivalente. Si usted utiliza esta misma licencia debe incluir una copia o bien su URI, con cada obra derivada que usted distribuya o comunique públicamente. Usted no puede ofrecer o imponer ningún término respecto a la obra derivada que altere o restrinja los términos de esta licencia o el ejercicio de sus derechos por parte de los concesionarios de la misma. Us-

ted debe mantener intactos todos los avisos que se refieran a esta licencia y a la ausencia de garantías cuando distribuya o comunique públicamente la obra derivada. Usted no puede ofrecer o imponer ningún término respecto de las obras derivadas o sus transformaciones que alteren o restrinjan los términos de esta licencia o el ejercicio de sus derechos por parte de los concesionarios de la misma. Usted no puede reproducir, distribuir o comunicar públicamente la obra derivada con medidas tecnológicas que controlen el acceso o uso de la obra de una manera contraria a los términos de esta licencia. Si utiliza una licencia equivalente debe cumplir con los requisitos que ésta establezca cuando distribuya o comunique públicamente la obra derivada. Todas estas condiciones se aplican a una obra derivada en tanto que incorporada a una colección, pero no implica que ésta tenga que estar sujeta a los términos de esta licencia.

c) Si usted reproduce, distribuye o comunica públicamente la obra o la prestación, una colección que la incorpore o cualquier obra derivada, debe mantener intactos todos los avisos sobre la propiedad intelectual e indicar, de manera razonable conforme al medio o a los medios que usted esté utilizando:

1) El nombre del autor original, o el seudónimo si es el caso, así como el del titular originario, si le es facilitado.
2) El nombre de aquellas partes (por ejemplo: institución, publicación, revista) que el titular originario y/o el licenciador designen para ser reconocidos en el aviso legal, las condiciones de uso, o de cualquier otra manera razonable.
3) El título de la obra o la prestación si le es facilitado.
4) El URI, si existe, que el licenciador especifique para ser vinculado a la obra o la prestación, a menos que tal URI no se refiera al aviso legal o a la información sobre la licencia de la obra o la prestación.
5) En el caso de una obra derivada, un aviso que identifique la transformación de la obra en la obra derivada (p. ej., "traducción castellana de la obra de Autor Original,.º "guión basado en obra original de Autor Original").

Este reconocimiento debe hacerse de manera razonable. En el caso de una obra derivada o incorporación en una colección estos créditos deberán aparecer como mínimo en el mismo lugar donde se hallen los correspondientes a otros autores o titulares y de forma comparable a los mismos. Para evitar la duda, los créditos requeridos en esta sección sólo serán utilizados a efectos de atribución de la obra o la prestación en la manera especificada anteriormente. Sin un permiso previo por escrito, usted no puede afirmar ni dar a entender implícitamente ni explícitamente ninguna conexión, patrocinio o aprobación por parte del titular originario, el licenciador y/o las partes reconocidas hacia usted o hacia el uso que hace de la obra o la prestación.

d) Para evitar cualquier duda, debe hacerse notar que las restricciones anteriores (párrafos 4.a, 4.b y 4.c) no son de aplicación a aquellas partes de la obra o la prestación objeto de esta licencia que únicamente puedan ser protegidas mediante el derecho sui generis sobre bases de datos recogido por la ley nacional vigente implementando la directiva europea de bases de datos.

5. **Exoneración de responsabilidad** A MENOS QUE SE ACUERDE MUTUAMENTE ENTRE LAS PARTES, EL LICENCIADOR OFRECE LA OBRA O LA PRESTACIÓN TAL CUAL (ON AN .ᴬS-IS"BASIS) Y NO CONFIERE NINGUNA GARANTÍA DE CUALQUIER TIPO RESPECTO DE LA OBRA O LA PRESTACIÓN O DE LA PRESENCIA O AUSENCIA DE ERRORES QUE PUEDAN O NO SER DESCUBIERTOS. ALGUNAS JURISDICCIONES NO PERMITEN LA EXCLUSIÓN DE TALES GARANTÍAS, POR LO QUE TAL EXCLUSIÓN PUEDE NO SER DE APLICACIÓN A USTED.

6. **Limitación de responsabilidad**. SALVO QUE LO DISPONGA EXPRESA E IMPERATIVAMENTE LA LEY APLICABLE, EN NINGÚN CASO EL LICENCIADOR SERÁ RESPONSABLE ANTE USTED POR CUALESQUIERA DAÑOS RESULTANTES, GENERALES O ESPECIALES (INCLUIDO EL DAÑO EMERGENTE Y EL LUCRO CESANTE), FORTUITOS O CAUSALES, DIRECTOS O INDIRECTOS, PRODUCIDOS EN CONEXIÓN CON ESTA LICENCIA O EL USO DE LA OBRA O LA PRESTACIÓN, INCLUSO SI EL LICENCIADOR HUBIERA SIDO INFORMADO DE LA POSIBILIDAD DE TALES DAÑOS.

7. **Finalización de la licencia**

   a) Esta licencia y la concesión de los derechos que contiene terminarán automáticamente en caso de cualquier incumplimiento de los términos de la misma. Las personas o entidades que hayan recibido de usted obras derivadas o colecciones bajo esta licencia, sin embargo, no verán sus licencias finalizadas, siempre que tales personas o entidades se mantengan en el cumplimiento íntegro de esta licencia. Las secciones 1, 2, 5, 6, 7 y 8 permanecerán vigentes pese a cualquier finalización de esta licencia.

   b) Conforme a las condiciones y términos anteriores, la concesión de derechos de esta licencia es vigente por todo el plazo de protección de los derechos de propiedad intelectual según la ley aplicable. A pesar de lo anterior, el licenciador se reserva el derecho a divulgar o publicar la obra o la prestación en condiciones distintas a las presentes, o de retirar la obra o la prestación en cualquier momento. No obstante, ello no supondrá dar por concluida esta licencia (o cualquier otra licencia que haya sido concedida, o sea necesario ser concedida, bajo los términos de esta licencia), que continuará vigente y con efectos completos a no ser que haya finalizado conforme a lo establecido anteriormente, sin perjuicio del derecho moral de arrepentimiento en los términos reconocidos por la ley de propiedad intelectual aplicable.

8. **Miscelánea**

   a) Cada vez que usted realice cualquier tipo de explotación de la obra o la prestación, o de una colección que la incorpore, el licenciador ofrece a los terceros y sucesivos licenciatarios la concesión de derechos sobre la obra o la prestación en las mismas condiciones y términos que la licencia concedida a usted.

   b) Cada vez que usted realice cualquier tipo de explotación de una obra derivada, el licenciador ofrece a los terceros y sucesivos licenciatarios la concesión de derechos sobre la obra objeto de esta licencia en las mismas condiciones y términos que la licencia concedida a usted.

   c) Si alguna disposición de esta licencia resulta inválida o inaplicable según la Ley vigente, ello no afectará la validez o aplicabilidad del resto de los térmi-

nos de esta licencia y, sin ninguna acción adicional por cualquiera las partes de este acuerdo, tal disposición se entenderá reformada en lo estrictamente necesario para hacer que tal disposición sea válida y ejecutiva.

d) No se entenderá que existe renuncia respecto de algún término o disposición de esta licencia, ni que se consiente violación alguna de la misma, a menos que tal renuncia o consentimiento figure por escrito y lleve la firma de la parte que renuncie o consienta.

e) Esta licencia constituye el acuerdo pleno entre las partes con respecto a la obra o la prestación objeto de la licencia. No caben interpretaciones, acuerdos o condiciones con respecto a la obra o la prestación que no se encuentren expresamente especificados en la presente licencia. El licenciador no estará obligado por ninguna disposición complementaria que pueda aparecer en cualquier comunicación que le haga llegar usted. Esta licencia no se puede modificar sin el mutuo acuerdo por escrito entre el licenciador y usted.

Bibliografía

[1] GARCÍA CAMPOS, C. "DeTraS: sistema de seguimiento de actividad de desarrollo". Director: Gregorio Robles Martínez. Universidad Rey Juan Carlos de Madrid, 2005/2006.

[2] ÁLVAREZ JIMÉNEZ, E. "Mejoras a DeTraS: describiendo la actividad humana frente al ordenador". Director: Gregorio Robles Martínez. Universidad Rey Juan Carlos de Madrid, 2009/2010.

[3] PRESSMAN, R.S. *Ingeniería del Software, un enfoque práctico*. 5a ed. Madrid: McGraw-Hill, 2002.

[4] SOMMERVILLE, I. *Software Engineering*. 8a ed. Harlow, England: Addison-Wesley, 2007.

[5] LÓPEZ, J.E.; DOLADO COSÍN, J.J. "Estimación del Esfuerzo Software: Factores vinculados a la aplicación a desarrollar". *Actas de los Talleres de las Jornadas de Ingeniería del Software y Bases de Datos*. Vol. 2, No. 1, 2008.

[6] GARZÁS PARRAS, J. *Introducción a la estimación* [en línea]. <http://jgarzas.googlepages.com/Jgarzas_Estimacion_Putnam.pdf> [Consulta: noviembre de 2009].

[7] KEYES, J. *Software Engineering Handbook*. 1a ed. Florida: CRC Press, 2003.

[8] MORENO CAPUCHINO, A.M. *COCOMO II* [en línea]. <http://trevinca.ei.uvigo.es/~cfajardo/Nueva_carpeta/presentaciones/cocomo2k.pdf> [Consulta: diciembre de 2009].

[9] AMOR, J.J.; ROBLES, G.; GONZÁLEZ–BARAHONA, J.M. *Effort Estimation by Characterizing Developer Activity*. 2006.

[10] *Administración de tiempo* [en línea]. Wikilibros. <http://es.wikibooks.org/wiki/Administraci%C3%B3n_de_tiempo/Versi%C3%B3n_para_imprimir> [Consulta: noviembre de 2009].

[11] *The Free Software Definition* [en línea]. Free Software Foundation. <http://www.gnu.org/philosophy/free-sw.html> [Consulta: diciembre de 2009].

[12] *Social software* [en línea]. Wikipedia. <http://en.wikipedia.org/wiki/Social_software> [Consulta: junio de 2011].

[13] *Social networking service* [en línea]. Wikipedia. <http://en.wikipedia.org/wiki/Social_networking_service> [Consulta: junio de 2011].

[14] KERNIGHAN, B.W; RITCHIE, D.M. *The C programming language*. 2a ed. [s.l.]: Prentice Hall, 1988.

[15] *Lenguaje de programación C* [en línea]. Wikipedia. <http://es.wikipedia.org/wiki/Lenguaje_de_programaci%C3%B3n_C> [Consulta: febrero de 2010].

[16] *GLib Reference Manual* [en línea]. GNOME Project. <http://library.gnome.org/devel/glib/2.22/> [Consulta: diciembre de 2009 – mayo de 2010].

[17] *GObject Reference Manual* [en línea]. GNOME Project. <http://library.gnome.org/devel/gobject/2.22/> [Consulta: diciembre de 2009 – mayo de 2010].

[18] *Libnotify Reference Manual* [en línea]. GNOME Project. <http://developer.gnome.org/libnotify/> [Consulta: junio de 2011].

[19] *Extensible Markup Language* [en línea]. Wikipedia. <http://es.wikipedia.org/wiki/XML> [Consulta: junio de 2010].

[20] GONZALEZ DUQUE, R. *Python para todos* [en línea]. <http://mundogeek.net/tutorial-python/> [Consulta: noviembre de 2009].

[21] *About Python* [en línea]. Python Software Foundation. <http://python.org/about/> [Consulta: junio de 2010].

[22] *Django (web framework)* [en línea]. Wikipedia. <http://en.wikipedia.org/wiki/Django_(web_framework)> [Consulta: junio de 2010].

[23] *HTML* [en línea]. Wikipedia. <http://es.wikipedia.org/wiki/Html> [Consulta: junio de 2010].

[24] *Cascading Style Sheets* [en línea]. Wikipedia. <http://en.wikipedia.org/wiki/Cascading_Style_Sheets> [Consulta: junio de 2010].

[25] *Python Google Chart* [en línea]. <http://pygooglechart.slowchop.com/> [Consulta: mayo – junio de 2010].

[26] *SQLite* [en línea]. Wikipedia. <http://es.wikipedia.org/wiki/SQLite> [Consulta: junio de 2010].

[27] *About SQLite* [en línea]. SQLite. <http://www.sqlite.org/about.html> [Consulta: junio de 2010].

[28] *GTK+* [en línea]. Wikipedia. <http://es.wikipedia.org/wiki/GTK%2B> [Consulta: junio de 2010].

[29] *PyGTK* [en línea]. Wikipedia. <http://en.wikipedia.org/wiki/PyGTK> [Consulta: junio de 2010].

[30] *Glade* [en línea]. Wikipedia. <http://es.wikipedia.org/wiki/Glade> [Consulta: junio de 2010].

[31] *Bazaar (software)* [en línea]. Wikipedia. <http://es.wikipedia.org/wiki/Bazaar_(software)> [Consulta: junio de 2010].

[32] *Launchpad (website)* [en línea]. Wikipedia. <http://en.wikipedia.org/wiki/Launchpad_(website)> [Consulta: junio de 2010].

[33] *Pycha (PYthon CHArts)* [en línea]. Wikipedia. <https://bitbucket.org/lgs/pycha/wiki/Home> [Consulta: junio de 2011].

[34] *MySQL* [en línea]. Wikipedia. <http://en.wikipedia.org/wiki/MySQL> [Consulta: junio de 2011].

[35] *SourceForge Support* [en línea]. Wikipedia. <http://sourceforge.net/apps/trac/sourceforge/wiki/Support> [Consulta: junio de 2011].

[36] *Apache HTTP server* [en línea]. Wikipedia. <http://en.wikipedia.org/wiki/Apache_HTTP_Server> [Consulta: junio de 2011].

[37] *WordPress* [en línea]. Wikipedia. <http://en.wikipedia.org/wiki/WordPress> [Consulta: junio de 2011].

[38] *About Freshmeat* [en línea]. Wikipedia. <http://freshmeat.net/about> [Consulta: junio de 2011].

[39] *pbuilder User's Manual* [en línea]. Wikipedia. <http://www.netfort.gr.jp/~dancer/software/pbuilder-doc/pbuilder-doc.html> [Consulta: junio de 2011].