

## **Tema 3 - Bugtracking (DCA)**

HISTORIAL DE REVISIONES			
NÚMERO	FECHA	MODIFICACIONES	NOMBRE

## Índice

1. ¿Qué es el Bugtracking? (I)	1
2. ¿Qué es el Bugtracking? (II)	1
3. ¿Pero es necesario el Bugtracking?	1
4. Tipos de error (I)	2
5. Tipos de error (II)	2
6. Usos de las aplicaciones de Bugtracking	2
7. Algunas aplicaciones de Bugtracking	2
8. Ejemplos de uso (I): Bugzilla	3
9. Ejemplos de uso (II): Trac	3
10. Bugtracking y repositorios de código	3
11. Prácticas en grupo e individuales	3
12. Aclaraciones	3

Logo DLSI

**Tema 3 - Bugtracking**  
**Curso 2018-2019**

## 1. ¿Qué es el Bugtracking? (I)

- Una vez publicado un software, lo normal es recibir avisos de fallos.
- En función del número de usuarios el número de estos avisos puede llegar a ser una cantidad importante. . .
- En estos casos, deberíamos automatizar su tratamiento de algún modo, de lo contrario nuestros usuarios se podrían encontrar con situaciones donde sus avisos de fallos no son atendidos, se pierden o simplemente se ignoran.
- Algunos equipos de desarrolladores prefieren crear su propia herramienta de gestión y seguimiento de errores *LBT* (Local Bug Tracker).
- Hoy en día contamos con aplicaciones específicas para este cometido. A lo largo del tema comentaremos algunas de ellas.
- Por tanto un sistema de seguimiento de fallos o *bugtracking* es una aplicación que ayuda a los programadores a llevar un registro de los fallos (también de mejoras o añadidos nuevos -*wishlist*-) de su software que les son indicados por los usuarios del mismo.
- Como nos podemos imaginar, un componente importante de un sistema de *bugtracking* es la bb.dd. donde se guarda toda la información relacionada con un `fallo` en nuestro software.

## 2. ¿Qué es el Bugtracking? (II)

- En esta bb.dd. guardaremos toda la información relativa a un aviso de fallo y que puede ayudar en su resolución, p.e.:
  - La "gravedad" del fallo.
  - Cómo reproducir el fallo.
  - El usuario que lo ha detectado
  - Los programadores que intervienen en su resolución
  - Porqué el usuario piensa que es un fallo. . .
  - Su "estado" actual. . .
- Algunos sistemas de *bugtracking* están diseñados para trabajar en conjunto con el sistema de control de versiones que emplee el equipo de desarrolladores.

## 3. ¿Pero es necesario el Bugtracking?

- Juzga tu mismo, un par de ejemplos, según **Michael Meeks**:
  - . . . Went to a few talks, encouraged by a rather good talk on keeping bugzilla clean, GNOME has ~45k open bugs the majority un-confirmed. . . *Bug or Feature?*.
  - . . . it's somewhat encouraging to have a more tractable ~5k open with ~1k unconfirmed vs. LibreOffice - then again, GNOME has a longer history and ~500k bugs total.

## 4. Tipos de error (I)

- Es conveniente fijar unos tipos de error predefinidos para facilitárselos al usuario que reporta un fallo.
- Es conveniente disponer de distintos tipos de error, pero tampoco demasiados, tomemos como ejemplo el seguimiento de fallos del s.o. **Debian**:

### **Important**

"Un fallo que tiene un efecto importante en la usabilidad de un paquete, sin hacerle completamente inútil para todo el mundo."

### **Normal**

"El valor por omisión, aplicable a la mayoría de los fallos."

### **Minor**

"Un problema que no afecta a la utilidad del paquete, y presumiblemente es trivial de arreglar."

### **Wishlist**

"Para la petición de cualquier característica, y también para cualquier fallo que sea muy difícil de arreglar debido a consideraciones de diseño mayores."

## 5. Tipos de error (II)

*El BTS de Debian admite otros tipos de fallos, pero para un usuario inicial, quizás más tipos de fallos le hagan echarse atrás a la hora de reportar un fallo.*

### **critical**

Hace que software no relacionado entre sí en el sistema (o el sistema entero) falle, o cause serias pérdidas de datos, o introduzca un agujero de seguridad en el sistema donde se instale el paquete.

### **grave**

Hace que el paquete en cuestión no se pueda utilizar o no se pueda casi nunca, o cause pérdida de datos, o introduce un agujero de seguridad que permita el acceso a las cuentas de los usuarios que usen el paquete.

### **serious**

Es una violación severa de la política de Debian (en pocas palabras, viola una directiva debe (must) o requerida (required)) o, en opinión del responsable del paquete o del responsable de la publicación de una versión de debian, hace que el paquete no se pueda publicar.

## 6. Usos de las aplicaciones de Bugtracking

- Recibir los avisos de los usuarios. Almacenarlos y evitar que se pierdan.
- Conocer cuáles eran avisos de errores "reales" y cuáles no.
- Conocer cuál es el estado del tratamiento de cada error real en un instante dado.
- Saber cómo están respondiendo nuestros programadores a esos avisos de error, p.e.: a los usuarios que reportan un error no les gusta que se resuelva con un **"WONTFIX"** o con un **"It's not an error but a feature"**.

## 7. Algunas aplicaciones de Bugtracking

- **Bugzilla**
- Original del proyecto **Mozilla**

- Lo emplean diversos proyectos ([wine](#), [linux kernel](#), [Gnome](#), [Kde](#), etc. . . )
- [Trac](#). Trac también dispone de una importante base de [usuarios](#).
- [Redmine](#). Al igual que los anteriores, también dispone de una amplia base de [usuarios](#).
- [Jira](#), más información en la [entrada de Jira en la wikipedia](#).

## 8. Ejemplos de uso (I): Bugzilla

## 9. Ejemplos de uso (II): Trac

## 10. Bugtracking y repositorios de código

- La mayoría de proyectos hoy en día emplean servicios como [GitHub](#) o similares.
- Parece poco efectivo entonces separar la gestión de bugs de la gestión del código fuente.
- Es por eso que servicios como *github* o similares integran la gestión de bugs (*issues*) en su flujo de trabajo.

## 11. Prácticas en grupo e individuales

- **Grupo:** Compara entre si dos de estos sistemas de control de fallos. ¿Qué resaltarías de cada uno de ellos?, ¿Porqué lo recomendarías?. ¿Porqué no lo recomendarías?
- **Grupo:** El desarrollador Andre Klapper mantiene un [blog con trucos sobre Bugzilla](#), repasadlos y preparad una presentación/resumen de los mismos.
- **Grupo:** Compara la gestión de bugs de [github](#) con la de [gitlab](#). ¿Qué ventajas e inconvenientes le ves a cada una?
- **Individual:** Crea tu propio *LBT*. Con las herramientas que quieras, aunque sea sencillo. Debe permitirte crear un nuevo aviso de fallo, poder seguirlo (añadir comentarios) y darlo por cerrado (como mínimo).
- **Opcional:** Instala bugzilla (`sudo su; pacman -Sy bugzilla; exit`), da de alta un proyecto, reporta algunos fallos.

ENTREGA:

- La práctica se entregará en [pracdlsi](#) en las fechas allí indicadas.

## 12. Aclaraciones

EN NINGÚN CASO ESTAS TRANSPARENCIAS SON LA BIBLIOGRAFÍA DE LA ASIGNATURA.

- Debes estudiar, aclarar y ampliar los conceptos que en ellas encuentres empleando los enlaces web y bibliografía recomendada que puedes consultar en la página web de la [ficha de la asignatura](#) y en la [web propia de la asignatura](#).
-