Bugtracking

Desarrollo Colaborativo de Aplicaciones

01: Introducción

02: Fundamentos de *bugtracking*

03: Aplicaciones de *bugtracking*

Introducción

101 del *software*

Cuando programamos aplicaciones…

Cometemos errores.

101 del *software*

¿Cuándo aparecen la mayoría de esos errores?

- En desarrollo.
- En producción.
- Días después de la salida de la última versión.
- Cuando menos te lo esperas.
- Cuando se te ha olvidado cómo se hizo la *app*.

```
// Dear programmer:
// When I wrote this code, only god and
// I knew how it worked.
// Now, only god knows it!
//
// Therefore, if you are trying to optimize
// this routine and it fails (most surely),
// please increase this counter as a
// warning for the next person:
//
// total_hours_wasted_here = 254
//

```

101 del *software*

¿Dónde aparecen la mayoría de esos errores?

- En la última actualización que hiciste.
- En una dependencia sin testear.
- En código que llevas años sin ver.
- En una librería que no programaste.

101 del *software*

Los ***bugs*** son un hecho inevitable del **desarrollo de *software***. La clave no está en hacer todo lo posible por evitarlos, sino en **saber gestionarlos inteligentemente** e integrarlos como parte del desarrollo.

Bugtracking

El *bugtracking* surge de una necesidad: **controlar y gestionar los *bugs* de nuestro *software***

Bugtracking

El *bugtracking* surge de una necesidad: **controlar y gestionar los *bugs* de nuestro *software***

Bugtracking

¿Es realmente importante?

Bugtracking

¿Es realmente importante?

Blog de Michael Meeks

> GNOME has ~45k open bugs the majority un-confirmed … it's somewhat encouraging to have a more tractable ~5k open with ~1k unconfirmed vs. LibreOffice

Bugtracking

¿Es realmente importante?

Es una inversión a **futuro**:

- Aporta trazabilidad y documentación.
- Evita duplicados.
- Permite priorizar y decidir.
- Evita duplicados.

Bugtracking

Cómo aplicar *bugtracking* es una decisión **entera del equipo** de desarrollo.

- Algunos prefieren usar su propio *software*.
- Otros prefieren usar herramientas estándar.

Fundamentos del *bugtracking*

¿Qué es el *bugtracking*?

Un sistema de *bugtracking* es una **base de datos** que almacena la información sobre los **erores** que surgen durante el uso de un *software* concreto.

¿Qué es el *bugtracking*?

La implementación de dicha B.D. es indiferente, siempre que proporcione la siguiente información:

- La gravedad del error.
- Cómo reproducir el error.
- El usuario que lo ha detectado.
- El *estado* en el que se encuentra.

¿Qué es el *bugtracking*?

Otras funcionalidades que puede incluir:

- Los programadores responsables.
- La prioridad del *bug* en el desarrollo.
- Relación en el sistema de control de versiones.

Severidad vs prioridad

Es importante distinguir entre **severidad** y **prioridad** al definir un bug.

Severidad vs prioridad

Es importante distinguir entre **severidad** y **prioridad** al definir un bug.

Severidad: Impacto técnico del *bug* en el *software*.

Severidad vs prioridad

Es importante distinguir entre **severidad** y **prioridad** al definir un bug.

Severidad: Impacto técnico del *bug* en el *software*.

Prioridad: Urgencia de resolución del *bug*.

Severidad vs prioridad

¿A mayor severidad, mayor prioridad?

Catalogación por severidad

Los bugs se **clasifican** en grupos principales. Por ejemplo, en Debian…

Catalogación por severidad

Los bugs se **clasifican** en cuatro grupos principales. Por ejemplo, en Debian…

Important

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

Catalogación por severidad

Los bugs se **clasifican** en cuatro grupos principales. Por ejemplo, en Debian…

Normal

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

Catalogación por severidad

Los bugs se **clasifican** en cuatro grupos principales. Por ejemplo, en Debian…

Minor

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

Catalogación por severidad

Los bugs se **clasifican** en cuatro grupos principales. Por ejemplo, en Debian…

Wishlist

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

Catalogación por severidad

También se definen algunas categorías específicas:

Crítico

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.

Catalogación por severidad

También se definen algunas categorías específicas:

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.

Catalogación por severidad

También se definen algunas categorías específicas:

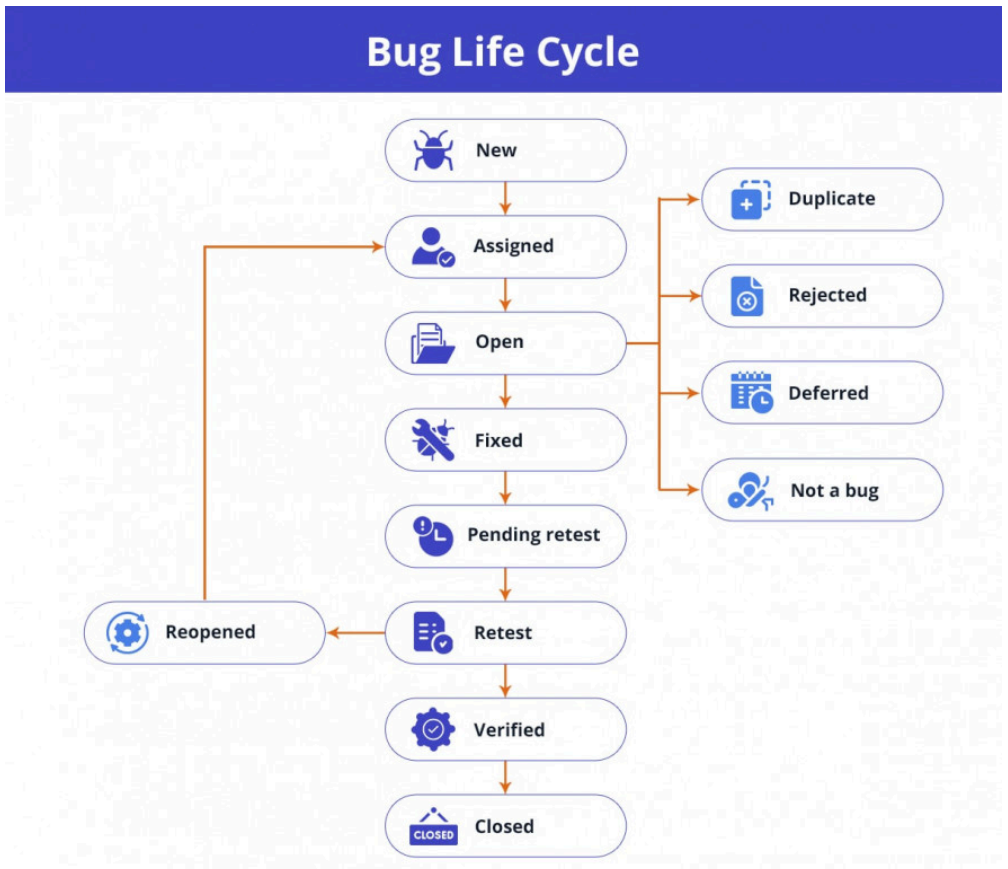
Serio

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.

Ciclo de vida de un *bug*

Ciclo de vida de un *bug*

- Un usuario lo notifica.
- Se realiza un **triaje**.
  - Evaluación y asignación del *bug*.
- Se arregla y *testea*.
- Se verifica y cierra.



La pregunta del millón

¿Y… cómo gestionamos el *bugtracking* de nuestra aplicación?

Aplicaciones de *bugtracking*

Las aplicaciones de *bugtracking*

El *bugtracking* se puede gestionar con *software* hecho a medida.

Sin embargo, **existen herramientas populares** para gestionarlo.

Beneficios de aplicaciones de *bugtracking*

- Recibir avisos de los usuarios y almacenarlos.
- Conocer el estado del tratamiento de los *bugs* al instante.
- Conocer cómo el equipo responde a los bugs y da soporte.

¿Qué aplicaciones de *bugtracking* conoces?

Aplicaciones de *bugtracking*

Existen múltiples aplicaciones de *bugtracking* en la actualidad:

Open-source: **Bugzilla**, Redmine, **Trac**, Fossil, Phorge, MantisBT.

Aplicaciones de *bugtracking*

Existen múltiples aplicaciones de *bugtracking* en la actualidad:

***Open-source*: Bugzilla**, Redmine, **Trac**, Fossil, Phorge, MantisBT.

Enfocadas a empresa: Jira, YouTrack, ClickUp, Asana, Monday.

Aplicaciones de *bugtracking*

Existen múltiples aplicaciones de *bugtracking* en la actualidad:

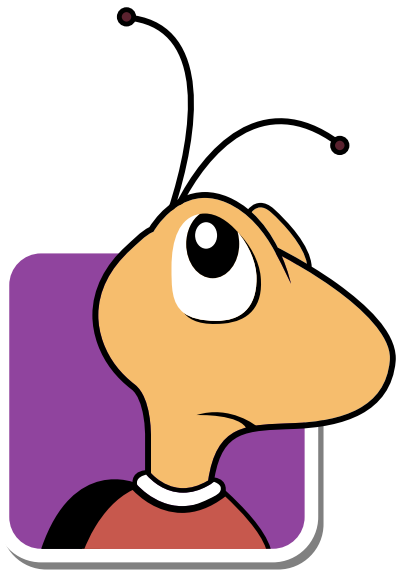
***Open-source*: Bugzilla**, Redmine, **Trac**, Fossil, Phorge, MantisBT.

Enfocadas a empresa: Jira, YouTrack, ClickUp, Asana, Monday.

Integradas en repositorios: GitHub Issues, GitLab Issues, BickBucket Issues.

BugZilla

- *Open Source*
  - Mozilla (1990).
- MariaDB + Apache + Perl.
- Robusto y altamente configurable.
- Usado en proyectos muy grandes de larga duración.
- Difícil de usar.



Trac

- *Bugtracker* minimalista
- Python.
- Wiki y *tickets* en un mismo sistema.
- Integra control de versiones.
- Simple y sencillo.

GitHub Issues

- Exclusivo de repositorios en GitHub.
- Enfocado en el trabajo colaborativo.
- Sencillo e intuitivo.
- Ideal para *open source*.

Jira

- Herramienta comercial de Atlassian.
  - Funciones gratuitas.
- Altamente configurable.
- Compatible con Scrum, Kanban y DevOps.
- Estándar en empresas.

¿Qué *bugtracker* escojo?

¿Qué *bugtracker* escojo?

No hay una respuesta única. Todo depende del proyecto y sus características.