**Разруливаем баги в Mantis**

В нашем мире существуют люди, которые могут найти баги даже в идеально написанной программе. Обороняться от такой несправедливости бессмысленно, поэтому лучше это дело упорядочить.

Для этого используются системы управления дефектами вроде “Mantis”.

* Да, можно называть дефекты багами. Главное не в названии, главное в том, что их надо чинить.

**Принципиальная схема работы с Mantis**

1. Если ты видишь баг – сделай об этом запись в Mantis. Не емайлом, не в скайпе, и не лично, и не молча, если на его починку нужно больше, чем десять секунд. Просто запиши это в Mantis.
2. Заполнить там надо всего лишь четыре поля:
   1. **Category** – это выпадающий список категорий в отдельно взятом проекте.
   2. **Summary** – заголовок репорта о проблеме. Опиши свой “баг?” кратко и без подробностей.
      * Пример неправильного заголовка: “Не могу сохранить пароль”.
      * Пример правильного заголовка: “Не могу сохранить пароль если использую заглавные буквы”.
   3. **Description** – а вот тут пиши подробности.
   4. **Steps To Reproduce** – пожалуй, самая важная часть репорта. Ведь программист, который получает сообщение о баге, но не понимает, что и как ты сделал и почему появился баг – это очень рассерженный программист…

**Пример шагов для воспроизведения:**

* + - 1. Залогиниться на сайте (tester101/tester101)
      2. Создать тикет в суппорт.
      3. На третьем экране в поле Label вписать слово с двоеточием - 'doc:doc'.

1. Приложи скриншот найдненного тобою бага. Иногда это важнее слов.
2. В Mantis появляется новая задача со статусом New.

А поскольку мы еще и знаем, на чье имя направить этот репорт, и укажем это в системе, то статус нового репорта будет Assigned. И это очень круто, ведь:

1. основное значение Mantis в том, чтобы хранить историю работы над отдельными задачами.
2. мы всегда знаем, на чьей совести сейчас находится эта задача
3. и мы всегда знаем, что об этой задаче думает сам программист, потому что комментарии программистов к репортам всегда важны и рулезны.

**Что такое issue?**

Все записи в Mantis называются issue.

Принципиально это переводится как “предмет спора, разногласие, проблема”. Обычно это переводят как “Задача”.

Если я сделал запись с сообщением о дефекте – это сообщение о баге.

Если я сделал запись о том, что надо бы сделать функцию сортировки – это уже задача на разработку.

И то, и другое можно комментировать, снабжать дополнительной информацией, указывать адрес и время изменений и менять общий статус работы над issue.

**Общие правила работы с Mantis**

1. Созданная задача получает уникальный номер в системе, и статус **New**.
2. Если при создании задачи был выбран человек, который будет за нее отвечать, статус задачи становится **Assigned**.
3. Конечный статус задачи **closed**. До тех пор, пока задача не получила статус **closed** – она все еще находится в работе.
4. За каждую отдельную задачу отвечает тот, на чье имя она записана. Для этого используется поле **Assigned To**. Если ты уверен, что твое участие больше не требуется – переведи задачу в статус **Resolved** и укажи имя того, кому ты ее передаешь.
5. Прекрасен принцип “One Bug – One Issue”. Например, существует задача, которая была источником разработки. А мы нашли баг, который касается непосредственно этой задачи.
6. Не надо вписывать **баг как комментарий к задаче** – надо сделать новое issue. Иначе потом будет очень сложно “управлять багами”, и будет сложно понять, сколько багов было найдено, и сложно объяснять, что “это задача, и она сделана, но в комментариях есть баг, и он не починен”. Все задачи, которые связаны между собой, можно слегко “линковать” посредством поля **Relationships**.
7. Не стесняйся адекватно и вовремя менять статус задачи, с которой работаешь.
8. Будь внимателен с типом задачи (Issue type). Каждый тип (“Feature Request”, “Change Request”, “Bug”, “Information”) обрабатывается по-разному в процессе разработки. Простейший пример: тестировщик написал *Change Request,* а программист, не разобравшись, принялся воплощать изменения, считая их задачей…
9. Прежде чем описывать баг или задачу, попытайся написать ее так, чтобы она была понятна всем, в том числе и твоей маме, которая опасается компьютера. Представь себе, что эту задачу назначут твоему коллеге. Представь себе, что он будет ее читать в конце рабочего дня, сильной уставшим. В кого полетит первый камень с вопросом “Ты тут вообще о чем пишешь?”
10. Если возможно, указывай не только то, что надо сделать, но и причину по которой это надо сделать. Это поможет понять приоритетность задачи. Иногда это не очевидно, или очевидно, но не всем.
11. Не используй Mantis как personal task list. Он не для того предназначен.
12. Указывай версию софта, в которой была обнаружена проблема, и указывай версию, в которой это проблема была/будет решена.
13. Есть глубокий смысл в том, чтобы акаунты уровня **Developer** и **Tester** (по-умолчанию роли тестировщика в Mantis нет) были лишены возможности ставить статус **Closed**. Принципиально это должен делать менеджер проекта. Менеджер должен получать прошедшую весь девелоперский цикл задачу только в статусе **Tested**.
14. Администратор должен всячески **связать Mantis с существующей в компании subversion**-системой, как бы она ни называлась. В крайнем случае, следует добавить новый цифровой **Custom Field**, который будет являться обязательным для заполнени при переводе issue в статус Resolved.

**Подробная схема работы с Mantis**

[](https://testitquickly.files.wordpress.com/2009/06/mantis_development_processes__bug_processing.jpg)

Разруливание багов через Mantis

На картинке весьма подробная, но все-таки **принципиальная** схема.

Подобные схемы разнятся от конторы к конторе, но это именно “принципиальный подход”.

Предупреждение: для того, чтобы работать с Mantis в таком вот режиме, следует повозиться с настройкой статусов.

**Управление проектом через Mantis**

Это фантастика. Дело в том, что Mantis сделан **для разработчиков**, и вовсе не предназначался для раздачи задач и составления графиков успеваемости.

Но менеджерам системы типа Mantis нравятся тем, что они могут в режиме нереального времени видеть статус каждой задачи в отдельности – она в работе, или по ней требуется больше информации, или она передана в тестирование, или она уже вообще закрыта. Поэтому все менеджеры на планете Земля пытаются использовать Mantis и как систему управления задачами. Иногда это получается успешно. Иногда нет.

Однако некоторые товарищи умудряются прикрутить к Mantis [систему учета времени](http://www.tiutiun.com/projects/time-tracking-table-mantis), которое было затрачено бравыми разработчиками на решение существующих проблем.

Mantis это инструмент для разработчиков.

Записи в подобной системе помогают основательно сказать “*Эта проблема была решена еще в пятницу, 13-го числа – смотри логи*, *я там оставил комментарий о том, что изменения в коде зачекинены в ревизии 1478*…”

Записи в подобной системе помогают сортировать список задач, которые войдут в очередной релиз.

Записи в подобной системе помогают не упустить из виду какие-то проблемы и/или задачи, не забыть и не спрятать.

Дефолтный пароль администратора при первой установке Mantis:  
Username: **administrator**  
Password: **root**

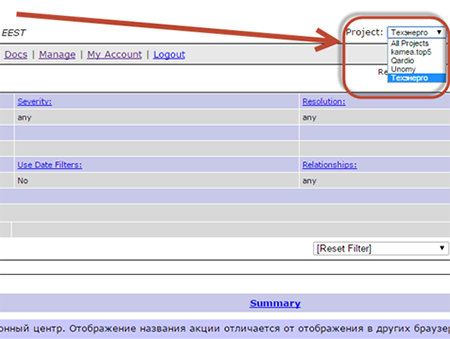
# Для каких проектов подходит багтрекер Mantis?

Известно, что красиво заведенный баг радует глаз тестировщика. А багтрекер является одним из инструментов, которые помогают успешно справиться с этой задачей.

Ни одно [**тестирование ПО**](http://ru.qatestlab.com/services/We-Are-Professionals-in/software-product-testing/), тестирование веб-сайтов и **тестирование мобильных приложений** не обходится без системы отслеживания ошибок.

Новички в тестировании часто начинают знакомство с системами отслеживания ошибок с работы в одном из самых простых и удобных багтрекеров – Mantis. Система разработана на PHP и распространяется свободно.

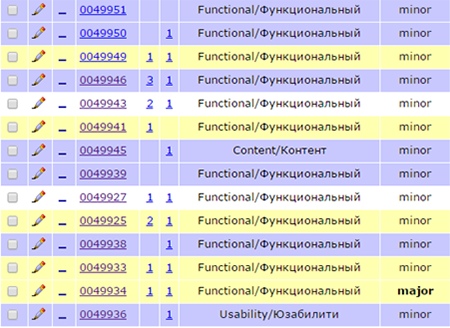
Mantis предоставляет возможность работы с большим количеством проектов. Проект меняется в меню программы.



Интерфейс программы – «на любителя». Но, как говорят, «Все гениальное – просто», и если Вы поклонник всего простого и удобного, то Вам он придется по вкусу. Здесь нет ничего лишнего, все действия могут быть выполнены с помощью минимального количества кликов, что бережет чуткие пальцы тестировщика от чрезмерного напряжения и усталости.

Основные функции стандартны для багтрекера.  Mantis позволяет добавлять, редактировать, просматривать баги. Очень радует возможность поиска с использованием фильтров, которые значительно упрощают работу со списком ошибок.

Эстеты оценят цветовую индикацию статусов багов, которая не только добавляет красок в унылые будни тестировщика, но и облегчает работу.



## Недостатки багтрекера Mantis:

1. Интерфейс. Как упоминалось выше, он очень удобный и простой, но вместе с этим довольно угрюмый. Видимо, разработчики багтрекера очень суровы и неприхотливы, но не все отличаются подобными качествами, поэтому дизайн системы можно отнести скорее к недостаткам.
2. В программе не предусмотрена возможность составления отчетов, что весьма печалит руководителей проектов.
3. Программа медленно работает с большими объемами информации, поэтому совершенно не подходит для долгосрочных проектов и обработки огромного количества данных.
4. Любителей иллюстрированного материала огорчает возможность прикрепления к багу только один скриншот. Уже после сохранения можно добавить новые скриншоты, но тоже по одному.

Mantis нельзя назвать идеальной системой отслеживания ошибок. Но простота его использования делает этот багтрекер замечательным инструментом. Тем, кто недавно начал выполнять [**мобильное тестирование**](http://mobile.qatestlab.com/?lang=ru), тестирование десктопных приложений и **тестирование сайтов**, он очень хорошо подходит.

Специалисты, проводящие ручное и [**автоматизированное тестирование**](http://ru.qatestlab.com/services/We-Are-Professionals-in/automated-testing/), утверждают, что Mantis подойдет для тех, кто не сильно придирчив к дизайну, работает над краткосрочными проектами и не нуждается в системе управления проектом.