ТЕСТИРОВЩИК ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ

КУРС «РУЧНОЕ ТЕСТИРОВАНИЕ»

8. ЖИЗНЕННЫЙ ЦИКЛ ДЕФЕКТА. ОФОРМЛЕНИЕ БАГ-РЕПОРТА.

Жизненный цикл дефекта, статусы, стадии.

Баг-репорт, для чего он нужен, как его составлять.

Чтобы отлично понимать статус багов, уметь их быстро и правильно описывать, оформлять и закрывать в системах по отслеживанию ошибок, необходимо знать их **жизненный цикл (Defect Lifecycle).**

Жизненный цикл дефекта – это стадии, которые проходит ошибка с начала своего существования и до ее полного разрешения. Чтобы было проще воспринимать, жизненный цикл рисуют схематично, где отображаются все статусы и действия, которые эти статусы и сменяют.

Каждый этап работы с дефектом обозначается как **«Статус»**. Такой статус на любом этапе показывает, что нужно сделать с ошибкой и кто сейчас с ней работает. Когда один из разработчиков заканчивает работу с ошибкой, статус ошибки меняется и она переходит «под власть» следующего, который продолжает работу с ней.

ЖИЗНЕННЫЙ ЦИКЛ ДЕФЕКТА

«New» (новый) – описание дефекта записывается в систему управления впервые.

«Assigned» (назначен) – отчет об ошибке назначен на определенного пользователя.

«**Open» (открыт)** – пользователь начинает работу с отчетом (анализ и редактирование).

«Fixed» (исправлен) – пользователь внес нужные исправления в код и протестировал их самостоятельно. Отчет со статусом «Исправлен» снова возвращается тестировщику.

«Pending retest» (Тестирование в режиме ожидания) – разработчик исправил баг, предоставил новый код для тестирования.

«Retesting» (повторное тестирование) – тестировщик повторно проверяет код, измененный разработчиком, с целью посмотреть, исправлена ли ошибка.

«Verified» (проверен) – если дефект исправлен, тестировщик ставит данный статус.

«Reopened» (переоткрыт) – если ошибка снова появляется, тестировщик опять перенаправляет ее на разработчика. Дефект проходит те же стадии.

СТАТУСЫ В ЖИЗНЕННОМ ЦИКЛЕ ДЕФЕКТА И ИХ ОБОЗНАЧЕНИЕ

«Closed» (закрыт) – такой статус ставится тестировщиком, если тот уверен, что дефект исправлен и он больше не появляется.

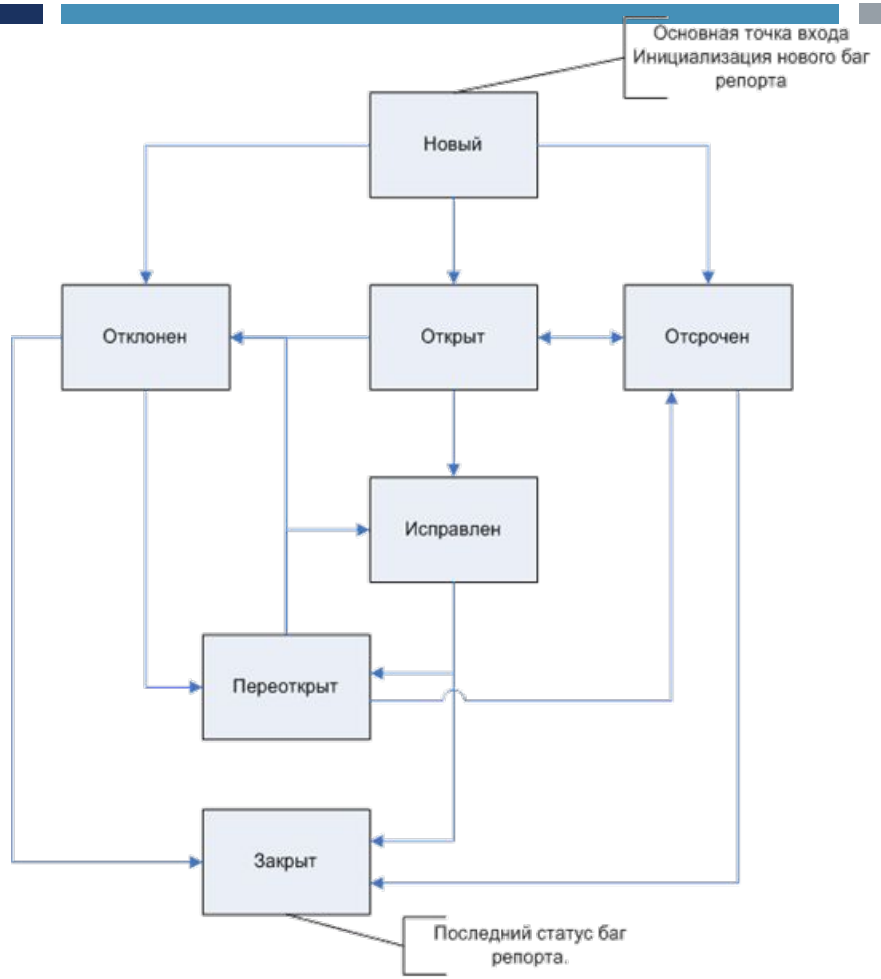
«Duplicate» (дубликат) – когда дефект встречается дважды или есть два дефекта, появившихся вследствие одной проблемы, один из них получает этот статус.

«Rejected» (отклонен) – если с точки зрения разработчика, ошибка не является весомой и она не требует рассмотрения и исправления, он ее отклоняет.

«Deferred» (отсрочен) – в режиме ожидания, что ошибку с таким статусом исправят в других версиях. В основном, дефект получает такой статус по нескольким причинам: низкий приоритет ошибки, недостаток времени, дефект не приведет к значительным сбоям в программе.

«Not a bug» (не является багом) – назначается в том случае, когда функциональные возможности программы меняться не будут. К примеру, заказчик хочет сменить габариты клавиш или цвет продукта — это не ошибка, а просто необходимость внесения поправок в дизайн программы.

СТАТУСЫ В ЖИЗНЕННОМ ЦИКЛЕ ДЕФЕКТА И ИХ ОБОЗНАЧЕНИЕ



. ЖИЗНЕННЫЙ ЦИКЛ ДЕФЕКТА

1. Тестировщик обнаруживает дефект.

2. Тестировщик пишет отчет об ошибке в систему управления дефектами (статус New (новый)) и перенаправляет его на разработчика (статус Assigned (назначен)).

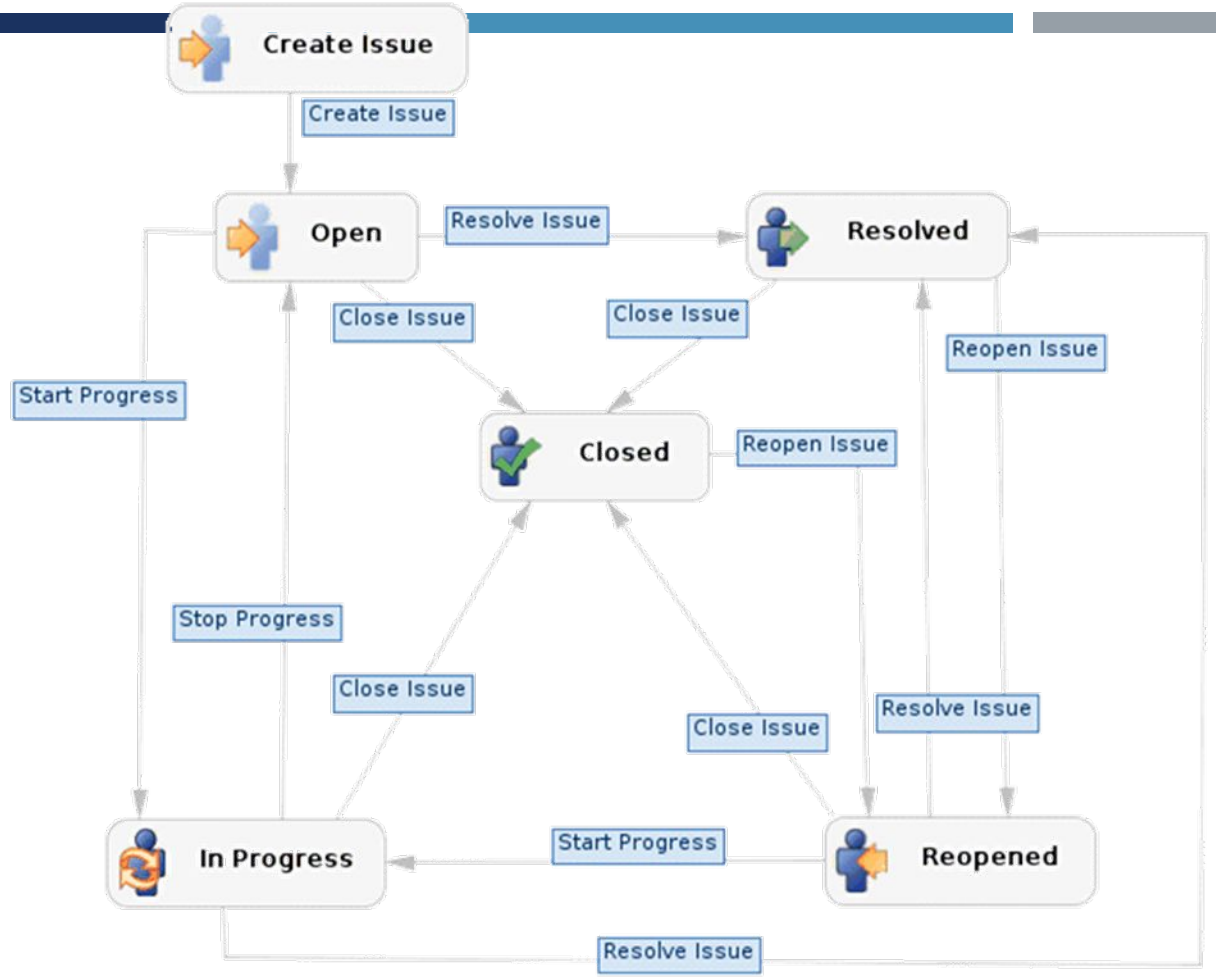
3. Разработчик изучает ошибку, ее возможности воспроизведения и по полученным результатам соотносит ее к одному из статусов: Duplicate (дубликат), Rejected (отклонен), Deferred (отсрочен), Not a bug (не баг),Open (открыт), Fixed (исправлен)

4.Тестировщик повторно проверяет ошибку (статус «Retesting» (повторное тестирование)).

5. Если дефект исправлен, тестировщик его закрывает (статусы «Verified» (проверен), затем «Closed» (закрыт).

6. Если дефект проявляется и дальше, он опять передается на редактирование разработчику (статусы «Reopened» (переоткрыт), «Assigned» (назначен)) и вновь проходит через каждую стадию цикла.

СТАДИИ ЖИЗНЕННОГО ЦИКЛА ОШИБКИ



ЖИЗНЕННЫЙ ЦИКЛ ДЕФЕКТА НА БАЗЕ JIRA

Баг-репорт (bug report) — это технический документ, который подробно описывает ошибку в работе программы, приложения или другого ПО. Его составляет тестировщик, чтобы разработчикам было понятно, что работает неправильно, насколько дефект критичен и что нужно исправить.

Баг-репорты — часть рабочего процесса. В них фиксируют наличие ошибки, назначают ответственного за исправление. Если сообщить об ошибке в рабочем чате, о ней скорее всего забудут. Каждый член команды подумает, что ошибку исправит другой, и в итоге она так и останется в коде.

Если кратко, то хороший баг-репорт позволяет:

- воспроизвести проблему;
- понять, в чем проблема, и какова ее важность.

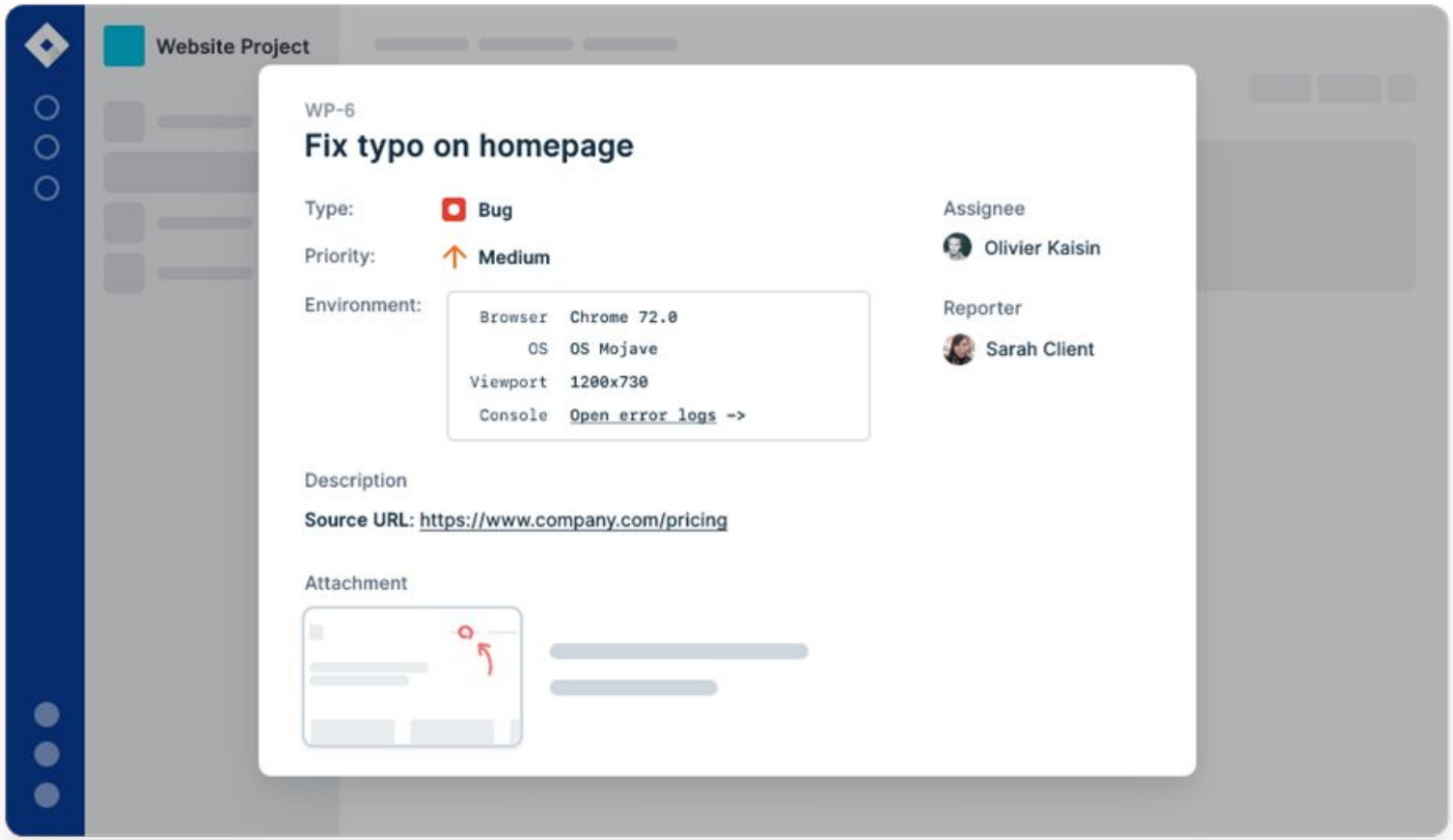
БАГ-РЕПОРТ

- **Функциональные.** Возникают, когда фактический результат работы не соответствует ожиданиям: не получается опубликовать комментарий на сайте, добавить товар в корзину или открыть страницу.
- **Визуальные.** Это случаи, когда приложение выглядит иначе, чем задумано: кнопка накладывается на текст, не отображаются картинки или текст выходит за пределы окна.
- **Логические.** Баг, при котором что-то работает неправильно с точки зрения логики, — например, когда можно указать несуществующую дату (31 февраля) или поставить дату рождения из будущего (2077 год).
- **Дефекты UX.** Приложение или программа неудобны в использовании: при просмотре ленты новостей пользователя постоянно отбрасывает к началу, слишком близко расположены кнопки и вместо одной нажимается другая.
- **Дефекты безопасности.** Случаи, когда из-за ошибки в коде данные пользователей (почты, пароли, фото, информация о платежах) могут быть доступны третьим лицам.

ВИДЫ БАГОВ

ID	**Идентификационный номер, который присваивается конкретному багу**
Краткое описание (также может называться Title или Summary)	Раздел, который кратко передает суть бага одним предложением. Отвечает на вопросы «Что?», «Где?», «При каких обстоятельствах?» Неправильно: «Проблемы с кнопкой» Правильно: «При нажатии на кнопку “Зарегистрироваться” на главной странице не открывается форма регистрации»
Проект	Название проекта, программы или приложения, в котором выявлен баг
Версия	Точная версия ПО, содержащая баг. Например, последняя версия iOS (операционной системы iPhone) — V14.7.1
Серьезность бага	Параметр, который определяет влияние бага на работу программы по шкале от S0 до S4
Приоритет	Параметр, который определяет срочность исправления бага по шкале от P1 до P3
Статус	Определяется в зависимости от того, на какой стадии находится баг: открыт, в работе, исправлен, отклонен, отсрочен и т.д.
Автор	Создатель баг-репорта
Исполнитель	Разработчик, который будет устранять баг
Шаги к воспроизведению	Точная последовательность действий, которая приводит к тому, чтобы воспроизвести баг
Фактический результат	К чему приводит воспроизведение шагов, как сейчас работает программа
Ожидаемый результат	Как должна работать программа на самом деле и к чему должны приводить действия, описанные в пункте «Шаги к воспроизведению»
Дополнения	Ссылки, скриншоты, видео и другие материалы, которые помогут исполнителю лучше понять суть проблемы

СТРУКТУРА БАГ-РЕПОРТА



ПРИМЕР БАГ-РЕПОРТА В JIRA

СЕРЬЕЗНОСТЬ И ПРИОРИТЕТ БАГОВ

- **S0 Trivial** (Тривиальный) — баг не влияет на работу программы, поэтому для его исправления могут не выделить отдельную задачу, а исправить попутно при исправлении других, похожих ошибок. Например, при заполнении анкеты в поле «Дата рождения» по умолчанию отображается не актуальный год, а 1999-й.
- **S1 Minor** (Незначительный) — баг почти не нарушает логику процессов, поэтому с ним программа может нормально работать. Например, неудобная навигация в интерфейсе.
- **S2 Major (**Серьезный) — баг создает неудобства в использовании, но еще не нарушает функционал программы.
- **S3 Critical** (Критический) — баг мешает приложению выполнять основные функции: калькулятор расходов неправильно считает бюджет или в текстовом редакторе невозможно вводить текст.
- **S4 Blocker** (Блокирующий) — ситуация, когда программа не работает в принципе: сайт выдает «ошибку 404» или не запускается приложение.

Приоритет — это срочность выполнения задачи. Всего выделяется три уровня приоритетов:

- **P1 Высокий** — исправляется в первую очередь, так как баг ломает работу приложения.
- **P2 Средний** — обязательный к исправлению баг после критического.
- **P3 Низкий** — не требует немедленного решения.

СЕРЬЕЗНОСТЬ И ПРИОРИТЕТ БАГОВ

- **Выявить причину возникновения.** Например, если на сайте не получается восстановить пароль, то проблема может быть как в бэкенде, так и во фронтенде. Задача тестировщика — разобраться в ней, так как от этого зависит, кому из разработчиков отдавать баг на исправление.
- **Провести проверку на разных устройствах.** Если проблема есть в десктоп-версии, то она может возникнуть и на мобильных устройствах, поэтому стоит проверить.
- **Провести проверку в разных версиях ПО.** Баг может не воспроизводиться в старой версии ПО, но появится в новой.
- **Описать несоответствие ожидаемому результату.** Чтобы сопоставить то, как работает программа сейчас, с ожидаемым результатом, начинающим специалистам лучше свериться с технической документацией и техническим заданием, где подробно описано, как все работает в идеале.

НА ЧТО СТОИТ ОБРАТИТЬ ВНИМАНИЕ ПРИ ОПИСАНИИ ДЕФЕКТА?

1. Для начала нужно убедиться, что найденный баг ещё не был оформлен. Следует провести поиск его в соответствующем проекте по всем подходящим ключевым словам и\или полям. Если баг уже есть, следует обновить его описание.
2. Если баг не найден – нажимаем на кнопку создания бага. Не стоит забывать важное правило: один дефект - один баг в трекере.
3. Далее нужно постараться кратко описать, что не работает - это и будет заголовок баг-репорта.
4. После этого перейти к подробному описанию бага:

- Указать шаги к воспроизведению.
- Указать ожидаемый результат. Можно добавить ссылку на спецификацию.
- Указать полученный результат.
- Указать версию ПО, также указать версию окружения.
- Если необходимо, приложить соответствующие артефакты: логи, скриншоты, дампы и т.д.

КАК ПРАВИЛЬНО ОФОРМИТЬ БАГ-РЕПОРТ

- *Заголовок не понятен.* Есть риск, что ни разработчик, ни коллеги не обратят внимания на довольно критичную проблему.
- *Отсутствуют шаги для воспроизведения.* Есть риск, что разработчик, не поняв как повторить проблему, вернёт баг со статусом «Не воспроизводится».
- *Неправильно назначен баг.* Возможно, баг по ошибке был назначен не на того разработчика или вообще остался в статусе “не назначен”. Есть риск, что багу долгое время не будет уделено внимание.
- *Недостаточность предоставленных данных.* Не всегда одна и та же проблема проявляется при всех вводимых значениях и под любым вошедшим в систему пользователем, поэтому настоятельно рекомендуется вносить все необходимые данные в баг-репорт. Иначе баг будет отклонён разработчиком, и придётся потратить время на его детальное описание.

ОШИБКИ ПРИ СОЗДАНИИ БАГ-РЕПОРТА

ОШИБКИ ПРИ СОЗДАНИИ БАГ-РЕПОРТА

- *Отсутствие ожидаемого или полученного результата*. В случаях, если вы не указали, что же должно быть ожидаемым поведением системы, вы тратите время разработчика на поиск данной информации, тем самым замедляете исправления дефекта. Рекомендуется указать ссылку на пункт в требованиях, написанный тест кейс или же ваше личное мнение, если эта ситуация не была задокументирована.

Если ваш баг-репорт составлен правильно, то шансы на быстрое исправление этих багов выше. Таким образом, исправление ошибки зависит от того, насколько качественно вы о ней сообщите. Смысл написания баг-репорта состоит в том, чтобы устранять проблемы. Составление правильных баг-репортов - не что иное, как навык, и его необходимо сформировать.

Если тестировщик не сообщает об ошибке правильно, программист, скорее всего, отклонит эту ошибку, заявив, что она не воспроизводится.

Поэтому, чем лучше тестировщики будут писать баг-репорты, тем дешевле обойдётся компании исправление этих дефектов.

ЗАКЛЮЧЕНИЕ

ТЕСТИРОВЩИК ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ

КУРС «РУЧНОЕ ТЕСТИРОВАНИЕ»

8. ПРАКТИЧЕСКОЕ ЗАНЯТИЕ

Изучение инструментов баг-трекинга

Оформление баг-репорта.

ИЗУЧЕНИЕ ИНСТРУМЕНТОВ БАГ-ТРЕКИНГА

Каждый день тестировщики создают десятки и сотни баг-репортов (отчетов об ошибках). Эти отчеты копятся, копятся и копятся.

Где лучше всего их хранить?

С этой задачей нам помогут справиться баг-трекинговые системы, которых на рынке сейчас достаточно много. Рассмотрим самые популярные и удобные из них.

JIRA

Jira представляет собой интерактивную доску (Дашборд), с помощью которой можно следить за выполнением поставленных задач. Все задачи классифицируются различными видами функций, подзадач, багов и т.д. Они могут редактироваться, назначаться на различных исполнителей или просто изменять статус с «открыт» на «закрыт». Все изменения по задаче записываются в журнал.

Плюсы:

— Широкий функционал, который можно дополнительно расширить с помощью плагинов.

— Интеграция с различными системами (Git, Zephyr, Trello, Slack, Google Drive & Docs, draw.io и так далее).

— Есть возможность строить диаграмму Ганта.

— Рабочие столы можно настроить под себя.

— Позволяет составлять план работы.

— Возможность искать задачи по гибким фильтрам.

— Наличие мобильного приложения.

— Связывание задач/ошибок.

— Уведомления по электронной почте.

— Пользователи получают последние обновления о ходе выполнения проектов.

JIRA

Минусы

— Сложность настройки и обслуживания, особенно для малого бизнеса и небольших команд.

— Иногда тяжело найти то, что нужно (из-за огромного количества функций).

— Требуется много времени, чтобы научиться эффективно использовать.

REDMINE

Redmine — решение для управления проектами с открытым исходным кодом и существует он уже более десяти лет. Написан на Ruby и совместим с MySQL, PostgreSQL, Microsoft SQL и SQLite.

Баг-репорт может отслеживаться любым сотрудником, который добавлен в проект и отмечен как наблюдатель.

MANTIS

Бесплатный инструмент. По сравнению с другими баг-трекинговыми системами, это довольно простой инструмент. Он доступен как в виде web-приложения, так и в мобильной версии. Баг-репорт можно назначить на любого пользователя, который работает в проекте.

Инструмент построен на PHP и совместим с базами данных MySQL и PostgreSQL. Его также можно настроить для управления проектами.

GOOGLE ТАБЛИЦЫ

Универсальный бесплатный инструмент. Если проект небольшой и в нем не так много баг-репортов, то Google Таблицы подойдут.

Плюсы:

— Бесплатно.

— Полная свобода творчества.

Минусы:

— Нет возможности интеграция с различными программами.

— Нет возможности учитывать время.

TRELLO

Веб-приложение, которое изначально было предназначено для управления проектами небольших групп. Платная система, но есть бесплатный тариф с определенными ограничениями.

Проекты распределяются на доски, которые выглядят очень наглядно. На досках задачи можно распределять по колонкам по принципу доски в канбан: новые задачи, задачи в очереди, задачи в работе, завершенные задачи и так далее.

Атрибуты, как таковые, в данной системе не предусмотрены. Но для присвоения серьезности или приоритета можно использовать цветовые маркеры. На доски добавляются определенные пользователи, которые закрепляются за задачами, где пишут комментарии по ходу выполнения.

YOUTRACK

Платный баг-трекер с возможностью пользоваться бесплатной ограниченной версией. Поддерживает Scrum и Kanban, а также работу по собственной (свободной) методике. Обеспечивает контроль просроченных задач, диаграммы «выгорания задач» и кумулятивного потока исполнения, поддержку вложенных задач, а также возможность обслуживания нескольких проектов в одной контрольной панели.

Доступен в виде облачного сервиса, либо в виде веб-приложения для установки на собственный веб-сервер.

TRAC

Открытая бесплатная веб-система для контроля багов и разработки софтверных продуктов. Есть русская локализация.

Trac специально создан для проектов разработки и отслеживания проблем, но также может использоваться для управления документами. Он имеет минималистский дизайн, встроенную вики и интегрируется с Apache Subversion и GitHub.

Можно связать ошибки с различными задачами, файлами, страницами вики или ошибками. Trac написан на Python и совместим с SQLite, MySQL и PostgreSQL.

.

ОСНОВНЫЕ ПОЛЯ БАГ РЕПОРТА И РОЛЬ РАБОТНИКА, ОТВЕТСТВЕННОГО ЗА ЗАПОЛНЕНИЕ ДАННОГО ПОЛЯ

Название графы	**Содержание графы**	**Кто заполняет**
Название баг-репорта	Что, где и когда происходит?	Тестировщик
Краткое описание ошибки	Расширенная версия названия	Тестировщик
Проект	Название проекта, в котором существует баг	Тестировщик
Компонент приложения	Часть или функция проекта, в которой находится баг	Тестировщик
Номер версии	Точный номер версии софта, в которой происходит ошибка	Тестировщик
Серьезность бага	От S1 до S5. Подробнее об этом в следующем разделе статьи.	Тестировщик или менеджер проекта
Приоритет бага	От P1 до P3. Подробнее об этом в следующем разделе статьи.	Менеджер проекта
Статус бага	Подробнее об этом в разделе «Жизненный цикл бага».	Тестировщик
Автор	Создатель баг-репорта	Тестировщик
Назначен на	Сотрудник, которому поручено исправление бага	Менеджер проекта
Окружение	В каких условиях был найден баг: версия ОС, версия браузера и тд.	Тестировщик
Шаги воспроизведения	Четкая последовательность действия, разбитая по шагам, которая позволит воспроизвести баг.	Тестировщик
Фактический результат	Результат, к которому приводит воспроизведение шагов из предыдущей графы таблицы.	Тестировщик
Ожидаемый результат	Результат, к которому должно приводить воспроизведение шагов из предыдущей графы таблицы.	Тестировщик
Дополнения	Все, что вы считаете нужным добавить к баг-репорту, но для этого нет соответствующей графы в таблице: дополнительная информация, файл с логами, скриншоты и т.п.	Тестировщик

УПРАЖНЕНИЕ

Представьте, что вы тестируете калькулятор Windows, который выдает: 1+1=2, 2+2=5, 3+3=6, 4+4=9, 5+5=10 и 6+6=13. Напишите отчет о дефекте, который бы эффективно описывал проблему.

- 5 мин на размышления
- 10 мин на разбор