# Основные Jira-проекты, используемые для разработки WoT:

**WOTD -** Проект в котором разрабатываются технические и геймплейные функциональности World of Tanks.

**WOTA -** Проект в котором создается арт контент: танки, карты, ангары, картинки, текстуры, кастомизации и т.д.

**WOTK -** Проект, в котором разрабатываются арт контент, технические и геймплейные функциональности World of Tanks на базе студии разработки в Киеве.

**WOTPRG -**Проект, в котором разрабатываются арт контент, технические и геймплейные функциональности World of Tanks на базе студии разработки в Праге.

**WOTCC -** Проект в котором проводятся работы над новыми версиями клиентского игрового движка. По своему назначению эти работы близки к проводимым в проекте WOTD. Но т.к. цикл разработки движка сильно отличается от цикла разработки продукта, работы по нему были вынесены в отдельный проект.

# Дополнительные Jira-проекты, в которых работает QA Game:

**WOTQC**(WOT Quality Control) **-**Ключевая особенность данного проекта в том, что в нем работают сотрудники внешних QC центров. В проекте WOTQC выполняются работы по контролю качества (отсюда и название проекта - quality control) различных областей World of Tanks: ежедневные проверки trunk/stable веток, проверки stability/performance, регресс тестирование отдельных веток в которых разрабатываются функциональности.

**BS**(Bug Support) - В данном проекте работает Центр Поддержки Пользователей (ЦПП) всех продуктов компании Wargaming. Сотрудники ЦПП принимют обращения игроков о замеченых проблемах и выставляют баги на проверку. Если BS баг валиден и не был ранее обнаружен/выставлен, он копируется в один из основных проектов, где и проводятся дальнейшие работы по его устранению.

**WGDP**(Global Deployment) - В данном проекте проводятся работы по разворачиванию и поддержке продакшн и локальных серверов. Сюда же относятся задачи по применению микропатчей на продакшн серверах.

**WTPOPS** и **WDO** (Web Development Operations) - В обоих проектах проводятся работы по разворачиванию, настройке и поддержке различных сервисов для продакшн и локальных серверов. Сервис - это дополнительный модуль подключаемый к серверу и включающий на нем ту или иную функциональность.

**WOTQA -** В данном проекте проводятся работы по развитию дисциплины QA продукта World of Tanks. Наиболее частые задачи проекта WOTQA - обучение новых сотрудников, тестовые задания, логгирование времени.

При разработке продукта WoT используются шесть типов ишью:

* Epic Story
* User Story
* Task
* Sub-Task
* Bug
* Improvement

**Epic Story** - контейнер разрабатываемой фичи или проекта, служаший для хранения ишью других типов. В зависимости от масштаба разрабатываемой фичи/проекта контейнер Epic Story может содержать от нескольких до десятков (а в некоторых случаях даже сотен) производственных задач.

**Epic Story передается в отдел тестирования для финального контроля завершения всего комплекса работ, поэтому данный тип issue относится к Extended Workflow.**

**Bug’и обнаруживаются в ходе проверки прилинкованных задач (User Story) и линкуются к ним же.**

**Epic Name -**Уникальный тип связи, присутствующий только у **Epic Story**. Используется для декомпозиции работ по проекту, в качестве альтернативы типу **include.** Данным типом связи может быть включен любой issue, кроме **Sub-Task** и **Epic Story**.Создаётся на основании названия фичи и указывается при создании **Epic Story**. Дочерние issue добавляются через ID Epic Name или при помощи "**+**" в строке Issues in Epic.Все созданные дочерние задачи с Epic Link отображаются, как **Issues in Epic** в самом **Epic Story**.

**User Story** - это производственная задача, в рамках которой осуществляется написание кода или создание арт контента (танков, карт, ангаров, стилей и т.д.). Используется в Extended Workflow и передается в отдел тестирования после закрытия всех Tasks и Sub-Tasks и перевода User Story в статус Resolved.

**Task** - это аналог **User Story**, служащая для поддержки разработки. Принципиальное различие в том, что Task принадлежит к Standard Workflow, следовательно, не передается в тестирование, и задача закрывается её заказчиком (Reporter). Например, в рамках тасков могут проводиться организационные работы, дополнительная регрессия фичи, исследования. issue

**Sub-Task** – минимальная единица workflow. используемый для разделения работ в рамках других производственых задач. Создаётся внутри issue. Standard Workflow.

**Bug** – это технические ошибки или отклонение от реализации согласно проектной документации (объективный критерий), а также, в отдельных случаях, противоречие общепринятой логики (субъективный критерий).

**Improvement**– это запрос о необязательном улучшении функциональности. В рамках импрувментов не проводится разработка, а только принимается решение реализовывать предложеное улучшение или нет.

|  |  |  |
| --- | --- | --- |
|  | Bug | Improvement |
| Определение | Различие текущего поведения от ожидаемого, описанного в техническом задании. | Предложение по необязательному улучшению существующей функциональности, в рамках которого принимается решение о его реализации. |
| [Тип workflow](https://confluence.wargaming.net/pages/viewpage.action?pageId=1101228881) | Extended workflow | Standard workflow |
| Коммиты | Разрешены | Запрещены |
| Решение по issue | 1. Баг исправлен | 1. Импрувмент не будет реализован |
| 2. Баг не будет исправляться | 2. Импрувмент будет реализован |
|  | 1. Конвертация в User Story |
|  | 2. Выставление User Story со ссылкой на Improvement |
|  |  |
| Severity | Используется | Отсутствует |
| Priority | Любой | Не используется Very High и High приоритет |
| Description | Развернутое описание бага, раскрывающее детали выявленной проблемы, предусловие, шаги воспроизведения, а также ожидаемый и фактический результат. | Описывается способ улучшения существующей функциональности, инструментация или контента. |
| Visible for Player | Используется | Не используется |
| Reproducibility | Используется | Не используется |
| Affects Version | Используется | Не используется |
| Уникальные резолюции | Not a Bug | Passed, Cancelled |

Workflow - это схема рабочего процесса, поток задач и связанные с ним подпроцессы.

В Standard workflow участвует 2 роли:

* **Заказчик**
* **Исполнитель**

Standard workflow используется для следующих типов issue:

* **Task**
* **Sub-Task**
* **Improvement**

В Extended workflow участвует 3 роли:

* **Заказчик**
* **Исполнитель**
* **Проверяющий (QA)**

Extended workflow используется для следующих типов issue:

* **Epic Story**
* **User Story**
* **Bug**

>>>[Диаграмма процесса тестирования](https://confluence.wargaming.net/plugins/gliffy/viewer.action?inline=false&pageId=59289017&name=test%20balance%20of%20tanks&lastPage=%2Fpages%2Fviewpage.action%3FpageId%3D59289017&imageUrl=%2Fdownload%2Fattachments%2F59289017%2Ftest%2520balance%2520of%2520tanks.png%3Fversion%3D25%26modificationDate%3D1383567417000%26api%3Dv2&gonUrl=%2Fdownload%2Fattachments%2F59289017%2Ftest%2520balance%2520of%2520tanks%3Fapi%3Dv2)

**Виды линков:**

**К горизонтальным относятся:** relates/is related to; duplicates/is duplicated by; blocks/is blocked by; clones/is cloned by; has to be done before/has to be done after. **К вертикальным относятся:** includes/is included in, is child task of/is parent task of.

**Relates/is related** отражает смысловую связь задач, без привязки к очередности их выполнения. Применяется для связи задач не находящихся в иерархическом подчинении и указывает на их отношение.

**Duplicates/is duplicated by** отражает дублирование уже имеющейся задач. (одинаковые баги)

**Blocks/is blocked by** отражает невозможность выполнения задачи связанной данным типом линка. (blocking bugs)

**Clones/is cloned by** используется для копирования задачи в несколько проектов. (wotd слонирует баги wotqc)

**Has to be done before/has to be done after** отражает хронологическую последовательность

**Includes/is includes in –** отражает вложенность задач и включенные в них bug’и.

**is child task of / is parent task of –** Отражает родительскую связь задач, когда одна готовая задача "порождает" другую. **(Например Imrovement, ставший основанием для модернизации, в результате чего будет создан массивный комплекс задач)**

|  |  |
| --- | --- |
| **Open** | Первоначальный статус задачи. В этом статусе она поступает на исполнителя. |
| **InProgress** | Обуславливает начало работ по данной задаче. Выставляется Исполнителем в момент начала реализации. |
| **Resolved** | Выставляется после выполнения всех запланированных работ для перенаправления задачи в отдел тестирования. |
| **Verified** | Выставляется тестировщиком после проверки задачи в основной ветке разработки (trunk) с целью её перенаправления для "подлива" в необходимый бранч. |
| **Uploaded** | Присваивается разработчиком после переноса производственной задачи в необходимый бранч. В этом статусе задача возвращается на тестировщика для проведения финального тестирования в соответствующем бранче. |
| **Closed** | Присваивается после завершения всех предусмотреных тестовых сценариев по задаче, выставлении найденных багов, и окончательного закрытия. |
| **Reopened** | Статус предусматривающий процедуру переоткрытия задачи. Отделом QA выставляется лишь в следующих случаях: |
| а) функциональность невозможно протестировать по причине отсутствия в ветке. Как пример это случаи ошибочного подлива в другую ветку, или подлива лишь части файлов. |
| б) при обнаружении Blocking бага не позволяющего осуществлять дальнейшее тестирование и делающего задачу полностью неработоспособной. |
| в) при процессуальных ошибках производства, как примеры: передача задачи в тестирование с недейтсивтельной резолюцией, статусом и иными обязательными полями issue, исходная отправка задачи на проверку не на руководителя отдела QA, и т.д. |
| г) недостаток исходной информации для начала проведения проверок, - переоткрывается с прямым запросом на предоставление комментариев о особенностях реализации, типов необходимых проверок и т.д. |

* + [О процессах цикла производственных задач-багов, их статусах, резолюциях, и нормах переоткрытия](https://confluence.wargaming.net/pages/viewpage.action?pageId=19411526)
* Если возникает необходимость переоткрытия задачи\бага на этапе проверки подлива совершённого в стэйбл, то это означает, что проблема произошла именно на этапе модификации стабильной ветки. В таком случае присваивая статус **Reopen** мы искажаем действительное состояние ишью, ведь в транке с ним всё хорошо. Поэтому рекомендуется не переоткрывать ишью присваивая ей статус **Reopen**, а возвращать статус "на шаг раньше", а это уже **Verified (Test Resolution: Verified at Trunk**) с комментарием что перенос в стэйбл произошёл с ошибкой, и требуется это сделать еще раз.
* Отдельно стоит обратить внимание на значение статусов **Reopen**и**Verified**. Их можно назвать "юридически" значимыми статусами, поскольку в случае если тестировщик переоткрывает баг в связи с низким качеством работы, это учитывается на общей оценке специалиста реализовывавшего эту задачу. Если тестировщик перенаправляет ишью для подлива в стэйбл, то это значит, что качество данного подлива на достаточном уровне, чтобы использовать её в качестве формирующего версию звена.
* Работая с багами и задачами в проектах **WOTD** и **WOTA** предназначенных для подлива в стэйбл ветку, стоит учитывать, что за их перенос могут отвечать разные люди. В отличие от проекта **WOTD**, где подилв в стэйбл ветку осуществляет тот же разработчик, который занимался этой задачей-багом, в проекте **WOTA** это делают отдельные, назначенные люди. Такми образом, если в проекте **WOTD**, после проверки в транке, вы отправляете баг в на подлив тому же разработчику, который исправлял его в транке, то в **WOTA** адресат может быть иным.

[Особенности жизненного цикла задач и багов, относящихся к клиентскому интерфейсу/серверной или геймплейной код-базе](https://confluence.wargaming.net/pages/viewpage.action?pageId=1376907823)

Авторский контроль:

UI-программист: обязательно привлекает UX-дизайнера для проведения АК перед отправкой на ревью кода, если в рамках задачи (или бага) были осуществлены любые изменения интерфейсов

UX-дизайнер: оставляет однозначный комментарий в задаче о прохождении АК либо указывает, что требуется исправить.

QA-инженер: осуществляет контроль проведения АК для задач. Свидетельством о проведении работ является заключение в виде комментария от UX-дизайнера. Если такого комментария нет ни в задаче, ни в относящихся к ней дочерних задачах (Task, Sub-task и т.д.), то задача должна быть переоткрыта на исполнителя.

Авторский контроль выполняется для следующих типов задач:

* Производственные задачи (User story), в рамках которых создаётся новый интерфейс либо изменяется текущий.
* Баги, в рамках которых прорабатываются UX-решения. Как правило, это баги, имеющие двойной компонент - UX/UI.

Code Review

Code Review - это проверка реализации задачи одним разработчиком за другим разработчиком с целью повышения ее технического качества, а также с целью обмена знаниями о ее реализации.

Примеры ситуаций, когда проходит процедура Code Review

* Ведущий программист делает Review написанного кода младшим сотрудником;
* Опытный программист отдает свой код на проверку таким же опытным коллегам, потому что реализованная функциональность сложна и шанс упустить некоторые нюансы велик.

Для прохождения Code Review используется специальный инструмент - FishEye.

Code Review это не обязательная процедура, QA не контролируют факт ее прохождения, следовательно могут принимать в работу задачи без пройденного Review, а отсутствие пройденного Code Review не может являться основанием для переоткрытия задачи.

[Code Review Process](https://confluence.wargaming.net/display/DEV/Code+Review+Process)

Unit-tests

Unit-test - это участок кода, который проверяет другой участок кода на предмет присутствия в нем функциональных или стилистических ошибок. Помогают достаточно быстро проверить, не появились ли ошибки после последних изменений кода.

test\_clan\_request\_processor.py.

QA контролируют факт покрытия задачи unit-test'ами, но не проверяют их количество, содержание на наличие ошибок или другие характеристики. На текущий момент, только issue типа **User Story**, которые имеют JIRA-компонент**Server**или **Gameplay** должны покрываться unit-test'ами. **Исключение:** в задаче явным образом в виде комментария к ней от разработчика указано, что тестами покрываться она не должна (в силу небольших правок кода или других причин). Иногда, unit-test'ы могут создавать и в рамках Bug'ов, но покрытие ими всех багов не является обязательным условием для их резолва, в отличие от User Story.

Unit-test'ы являются обязательным атрибутом функциональных задач. Их отсутствие – основание для переоткрытия такой задачи на разработчика с соответствующим комментарием.