**Регламент работы с ветками, версиями и релизами**

Основная цель изменения процесса релизов – переход к коротким по длительности релизам, идущими один за другим. Что должно способствовать прогрнозируемости сроков выпуска нового функционала и исправлений существующих некритичных ошибок.

**Репозиторий**

Основные бранчи:

* master – соответствует последнему выпущенному релизу
* develop – соответствует текущей разрабатываемой версии
* release-candidate – соответствует текущему тестируемому релиз-кандидату

*Все большие изменения и новый функционал или группы связанного функционала(комплекс решений) разрабатываются в отдельных именованных бранчах. Решение о созданиии таких веток принимается коллективно и в целом является исключением из правил.*

Изменения в master, develop и release-candidate вливаются только через пулл-реквесты(PR). Перед слиянием необходимо, чтобы PR был одобрен ревьювером и прошёл все автоматические проверки: сборка, тесты хоста, тесты сервера, форматирование кода.

**Исправление ошибок.**

* Исправление ошибок текущей разрабатываемой версии начинаются от develop и сливаются в него же
* Критичные/срочные ошибки, исправление которых не может быть отложено до полноценного релиза, могут исправляться непосредственно в текущей версии. В таком случае бранч hotfix’а начинается от последнего коммита в master и сливается в master.
* Ошибки, выявленные в ходе тестирования релиз-кандидата, исправляются стартом бранча от последнего коммита в release-candidate и сливается соостветственно в release-candidate

**Новый функционал.**

* Разработка нового функционала стартует от develop и в последствии сливается в него же
* Разработка нового функционала, в рамках комплекса решений, стартует от именованного бранча комплекса и сливается в него же

**Релиз-кандидат.** Стартует от develop и после окончательного тестирования сливается в master. При выпуске очередного релиза стартует новый релиз-кандидат, включающий в себя весь набор изменений из текущей разрабатываемой версии. В случае отсутствия таких изменений, старт нового релиз-кандидата откладывается до появления изменений.

Для соответствия бранчей текущего релиз-кандидата и разрабатываемой версии бранчу актуальной версии релиза, после внесения изменения в master, эти изменения должны быть слиты в текущий релиз-кандидат и текущую разрабатываему версию. Соответствующее слияние так же должно происходить из релиз-кандидата в разрабатываемую версию, при внесении изменений в релиз-кандидат. Таким образом изменения должны каскадно сливаться: master->release-candidate->develop. Слияние актуальных изменений разрабатываемой версии в именованные бранчи комплескных решений производятся вручную ответственными за эти решения людьми.

В случаях, когда каскадное слияние на одном из этапов вызывает неразрешимые конфликты слияния, например когда исправления в одной ветке должны сливаться в переписанный функционал, создаётся отдельный бранч от конфликтующей ветки и конфликты слияния решаются в нём. После решения конфликтов создаётся отдельны PR на слияние таких изменений

В целом схема выглядит так (полностью см. Схема релизов (Анатолий).drawio):

Graphical user interface

Description automatically generated

**Пулл-реквест.** При окончании исправления ошибки или разработки нового функционала разработчик формирует PR в соответствующий бранч и назначает ответственного(ых) за его рассмотрение. У соответствующей задачи в jira проставляет статус Review и указывает ответственного за ревью в поле Reviewer. При окончании ревью ответственный сливает PR в указанный бранч (обычным простым **Merge** не используя Squash, для большей прозрачности).

**Правила именования бранчей.** Название бранча должно соответствовать номеру задачи в jira, например: DPA-1234. Исключение составляют именованые бранчи отдельных и комплексных решений. *\*Префиксы и раскладывание по папкам обсуждаемы*

**Комментарии к коммитам.** Комментарии к коммитам пишутся на английском языке и должны начинаться с номера задачи из jira, например: “DPA-1234 fix total sum in ABC report”

**Jira**

Основные версии:

* Номера версий вида(трёхзначные): **5.8.1, 5.8.2, 5.8.3**… - соответствуют выпущенным минорным релизам (соответствует коммитам в master)
* Номера версий вида(номер мажорного релиза+”X”): **5.8.X** – маркируют задачи, планируемые к выпуску минорным релизом, и попадающих в текущую разрабатываемую версию (соответствует коммитам в develop)
* Номера версий вида(“rc-“+номер мажорного релиза+”X”): **rc-5.8.X** – маркируют задачи, попадающие в текущий релиз-кандидат (соответствуют release-candidate в репозитории)
* Номера версий вида(двузначные): **5.8, 5.9**… - соответствуют выпущенным или планируемым к выпуску мажорным релизам. В случае планируемого мажорного релиза соответствует именованному бранчу связанному с группой разрабатываемого функционала)

При выпуске очередного мажорного релиза(например: 5.8) в jira создается новая версия с номером следующего мажорного релиза(например: 5.9), версия для текущего релиз-кандидата rc-5.8.X, версия текущей разработки 5.8.X. Все текущие задачи помечаются версией текущей разработки, если не планируются в другой мажорный релиз.

При выпуске минорного релиза(хотфикс) создается версия соответствующая текущему релизу(например 5.8.2) и все задачи попадающие в данных хотфикс помечаются данной версией.

При выпуске минорного/мажорного релиза из релиз-кандидата создается версия соответствующая текущему релизу(например 5.8.2) и все задачи помеченные   
версией rc-5.8.X меняют свою версию на версию выпускаемого минорного/мажорного релиза: 5.8.2