Git Workflow

Инструкция по работе с git

Перед началом реализации фичи

1. Переключиться на ветку `master` (если вы уже не находитесь на ней)

   Текущая ветка отображается в пункте **Git Commit - > "ветка"...** в контекстном меню.

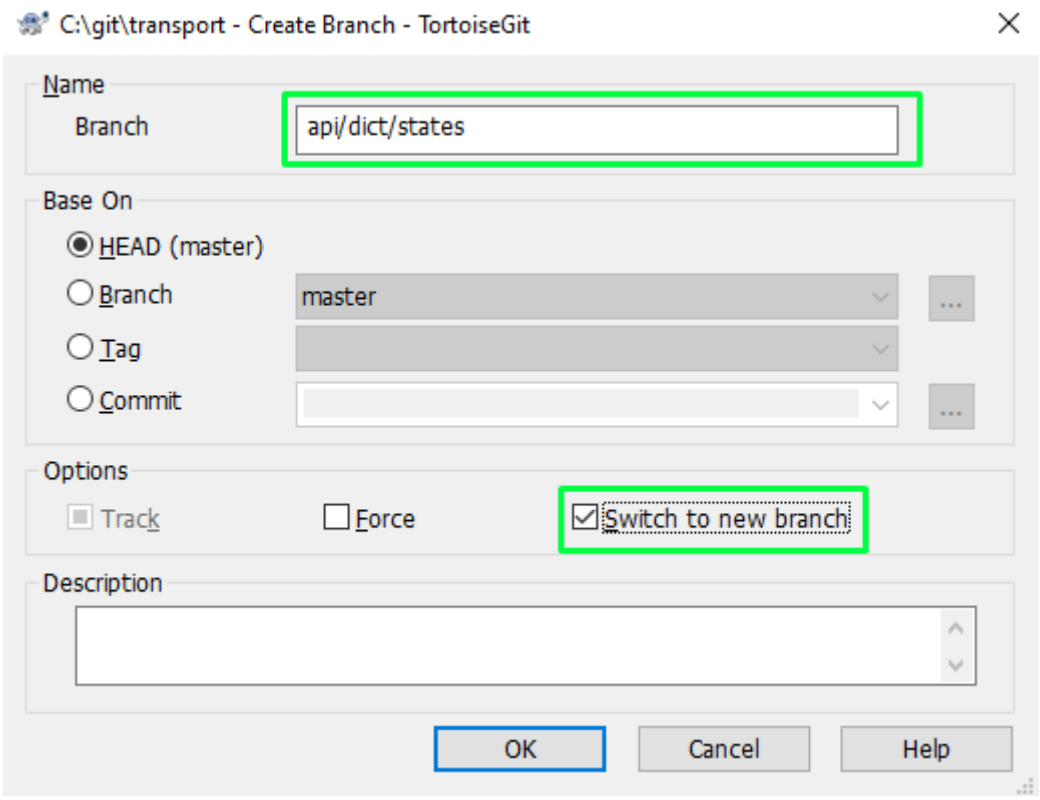
   ▼ **Переключение ветки**

   1. В контекстном меню выбрать **TortoiseGit → Switch/Checkout...**
   2. Если в выпадающем списке в поле **Branch** нет нужной ветки, нажмите на **[...]** и в появившемся диалоге выберите нужную ветку из списка **remotes**

2. Сделать **pull**
3. Создать ветку для фичи (feature branch) и переключиться на нее

   ▼ **Создание ветки**

   1. В контекстном меню выбрать **TortoiseGit → Create Branch...**
   2. В появившемся диалоге в поле **Branch** ввести название ветки
   3. Поставить галочку **Switch to new branch**



Создание ветки в TortoiseGit

Название ветки должно быть по шаблону: `<папка в src>/<краткое название фичи>`.
Например, `api/dict/states`, `api/dict/org-types`, `web/orders`.

Если в ветке реализуется как фронтенд, так и бэкенд, вместо префиксов *api* и *web* следует использовать префикс ***fs*** (full stack).

4. Начать пилить фичу

В процессе реализации фичи

1. Регулярно делать коммиты (как только появляется хоть небольшая завершенная часть задачи)
2. Сообщение коммита должно быть содержательным
3. У сообщения коммита должен быть префикс (*api, web*, *db*)
4. Перед каждым элементом в списке изменений тоже должен быть префикс

- **[+]** - фича (или ее часть) добавлена
- **[-]** - фича удалена
- **[*]** - изменение фичи
- **[!]** - важные изменения фичи
- **[wip]** - незавершенные изменения (work in progress; но так коммитить нежелательно)

Примеры хороших сообщений коммитов:
web: [+] Добавлен компонент основного меню
api: [+] Реализованы методы контроллера и сервиса для получения, создания и удаления заявок; [*] Метод GenericRepository.RemoveByIdAsync возвращает false, если не удалось найти сущность для удаления

Примеры плохих сообщений коммитов:
правил много всего
api: [*] Доработки справочников *// какие доработки? каких справочников?*

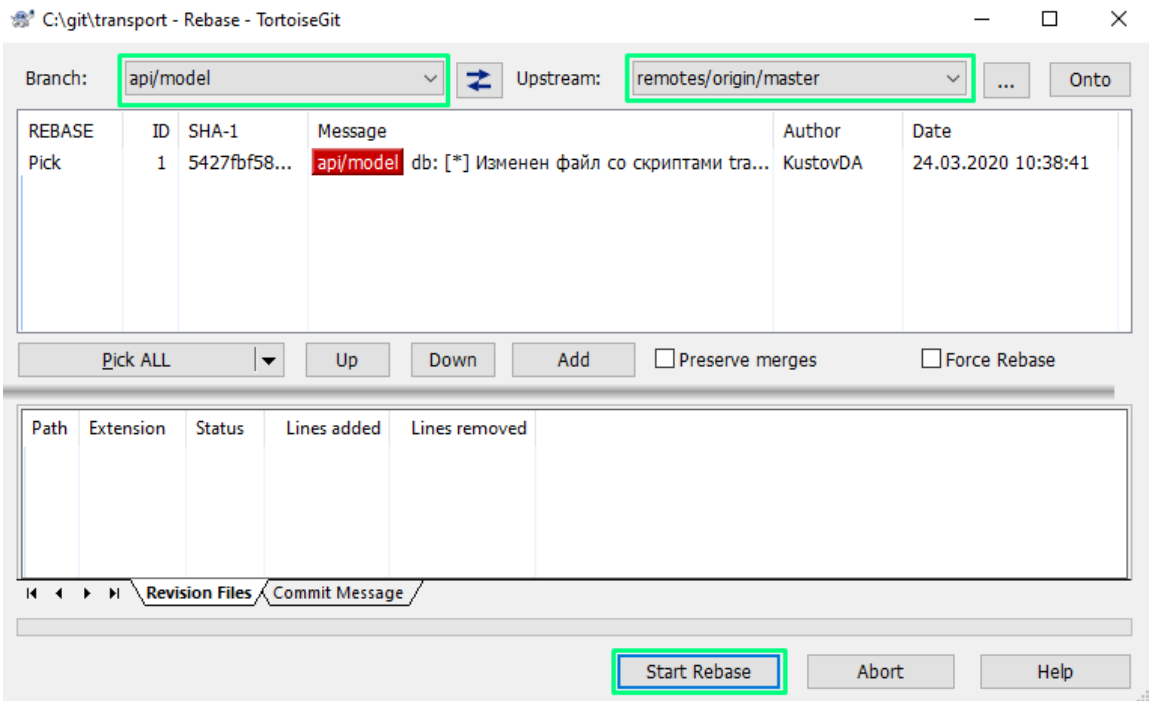
5. Регулярно выполнять rebase ветки master в ветку фичи

Чтобы проверить, есть ли изменения в `master` в удаленном репозитории:
1. выполнить **fetch** через меню **TortoiseGit → Fetch...**
2. открыть лог через меню **TortoiseGit → Show log** и ИЗМЕНИТЬ ВЕТКУ в левом верхнем углу на `remotes/origin/master` (для вызова диалога выбора нужно кликнуть по названию ветки);
3. жирным будет выделен коммит, который является последним в локальной ветке `master`, а все, что выше - новые коммиты из удаленного репозитория.

6. Изменения в PgDBInterface нужно коммитить отдельно (делать коммит из этой папки, т.к. это submodule), затем в родительском репозитории нужно сделать коммит, фиксирующий изменение PgDBInterface (эта папка будет иметь статус Modified)
7. Хранимые процедуры и функции тоже нужно коммитить (сохраняя в папку db)

После завершения реализации фичи

1. Сделать **fetch**
2. Сделать **rebase на** ветку `remotes/origin/master`, чтобы ветка фичи включала все актуальные изменения из удаленного репозитория



Выполнение rebase через TortoiseGit

3. Переключиться на ветку `master`
4. Сделать **merge** `master` с `remotes/origin/master`
5. Сделать **merge** `master` с веткой фичи (через меню **TortoiseGit → Merge...**)
6. Сделать **push**