

Федеральное государственное автономное
образовательное учреждение высшего
образования
«Национальный исследовательский университет
ИТМО»

Факультет Информационных технологий и программирования

Работа: Лабораторная работа по Git 2

Выполнил: Гонтарь Тимур Сергеевич
Проверил: Повышев Владислав Вячеславович

Санкт-Петербург
2022 г.

Система контроля версий Git

Git — это распределенная система контроля версий. Иначе говоря, это набор консольных утилит, которые отслеживают и фиксируют изменения в файлах (чаще всего речь идет об исходном коде программ, но вы можете использовать его для любых файлов). Изначально Git был создан Линусом Торвалдсом при разработке ядра Linux.

К базовым возможностям Git относятся:

- возврат к любой предыдущей версии кода;
- просмотр истории изменений;
- параллельная работа над проектом;
- backup кода.

Ядро Git представляет собой набор утилит командной строки с параметрами. Все настройки хранятся в текстовых файлах конфигурации. Такая реализация делает Git легко портируемым на любую платформу и даёт возможность легко интегрировать Git в другие системы (в частности, создавать графические git-клиенты с любым желаемым интерфейсом).

Репозиторий Git представляет собой каталог файловой системы, в котором находятся файлы конфигурации репозитория, файлы журналов, хранящие операции, выполняемые над репозиторием, индекс, описывающий расположение файлов, и хранилище, содержащее собственно файлы. Структура хранилища файлов не отражает реальную структуру хранящегося в репозитории файлового дерева, она ориентирована на повышение скорости выполнения операций с репозиторием. Когда ядро обрабатывает команду изменения (неважно, при локальных изменениях или при получении патча от другого узла), оно создаёт в хранилище новые файлы, соответствующие новым состояниям изменённых файлов. Существенно, что никакие операции не изменяют содержимого уже существующих в хранилище файлов.

Цикл работы в git:

- Редактирование, добавление, удаление файлов (собственно, работа).
- Индексация/добавление файлов в индекс (указание для git какие изменения нужно будет закоммитить).
- Коммит (фиксация изменений).
- Возврат к шагу 1 или отход ко сну.

Указатели:

- HEAD — указатель на текущий коммит или на текущую ветку (то есть, в любом случае, на коммит). Указывает на родителя коммита, который будет создан следующим.

- ORIG_HEAD — указатель на коммит, с которого вы только что переместили HEAD (командой `git reset ...`, например).

- Ветка (master, develop etc.) — указатель на коммит. При добавлении коммита, указатель ветки перемещается с родительского коммита на новый.

- Теги — простые указатели на коммиты. Не перемещаются.

Основные команды

Создание папок и файлов

`mkdir project` # создать папку с именем «project»

`touch index.html` # создать файл

Создать новый репозиторий

`git init` # создать новый проект в текущей директории

`git init folder-name` # создать новый проект в указанной директории

Клонирование репозитория

клонировать удаленный репозиторий в одноименную директорию

`git clone https://github.com/cyberspacedk/Git-commands.git`

клонировать удаленный репозиторий в директорию «FolderName»

`git clone https://github.com/cyberspacedk/Git-commands.git FolderName`

клонировать репозиторий в текущую директорию

`git clone https://github.com:nicothin/web-design.git .`

Просмотр изменений

`git status` # показать состояние репозитория (отслеживаемые, изменённые, новые файлы и пр.)

`git diff` # сравнить рабочую директорию и индекс

Добавление изменений в индекс

`git add .` # добавить в индекс все новые, изменённые, удалённые файлы из текущей директории и её поддиректорий

`git add text.txt` # добавить в индекс указанный файл (был изменён, был удалён или это новый файл)

`git add -p` # показать новые/изменённые файлы по очереди с указанием их изменений и вопросом об отслеживании/индексировании

Удаление изменений из индекса

`git reset` # убрать из индекса все добавленные в него изменения (в рабочей директории все изменения сохраняются), антипод `git add`

`git reset readme.txt` # убрать из индекса изменения указанного файла (в рабочей директории изменения сохраняются)

Отмена изменений

`git checkout text.txt` # ОПАСНО: отменить изменения в файле, вернуть состояние файла,

имеющееся в индексе

`git reset --hard` # ОПАСНО: отменить изменения; вернуть то, что в коммите, на который указывает HEAD (незаконченные изменения удалены из индекса и из рабочей директории, неотслеживаемые файлы останутся на месте)

`git clean -df` # удалить неотслеживаемые файлы и директории

Коммиты

`git commit -m "Name of commit"` # зафиксировать в коммите проиндексированные изменения (закоммитить), добавить сообщение

`git commit -a -m "Name of commit"` # проиндексировать отслеживаемые файлы (ТОЛЬКО отслеживаемые, но НЕ новые файлы) и закоммитить, добавить сообщение

Отмена коммитов и перемещение по истории

`git revert HEAD --no-edit` # создать новый коммит, отменяющий изменения последнего коммита без запуска редактора сообщения

`git revert b9533bb --no-edit` # то же, но отменяются изменения, внесённые коммитом с указанным хешем (b9533bb)

Временно переключиться на другой коммит

`git checkout b9533bb` # переключиться на коммит с указанным хешем (переместить HEAD на указанный коммит, рабочую директорию вернуть к состоянию, на момент этого коммита)

`git checkout master` # переключиться на коммит, на который указывает master (переместить HEAD на коммит, на который указывает master, рабочую директорию вернуть к состоянию на момент этого коммита)

Переключиться на другой коммит и продолжить работу с него

`git checkout -b new-branch 5589877` # создать ветку new-branch, начинающуюся с коммита с хешем 5589877 (переместить HEAD на указанный коммит, рабочую директорию вернуть к состоянию, на момент этого коммита, создать указатель на этот коммит (ветку) с указанным именем)

Восстановление изменений

`git checkout 5589877 index.html` # восстановить в рабочей директории указанный файл на момент указанного коммита (и добавить это изменение в индекс) (`git reset index.html` для удаления из индекса, но сохранения изменений в файле)

Копирование коммита (перенос коммитов)

`git cherry-pick 5589877` # скопировать на активную ветку изменения из указанного коммита, закоммитить эти изменения

Удаление файла

`git rm text.txt` # удалить отслеживаемый неизменённый файл и проиндексировать это изменение

`git rm -f text.txt` # удалить отслеживаемый изменённый файл и проиндексировать это изменение

Перемещение/переименование файлов

`git mv text.txt test_new.txt` # переименовать файл «text.txt» в «test_new.txt» и проиндексировать это изменение

`git mv readme_new.md folder/` # переместить файл readme_new.md в директорию folder/ (должна существовать) и проиндексировать это изменение

История коммитов

`git log master` # показать коммиты в указанной ветке

`git log -2` # показать последние 2 коммита в активной ветке

`git log -2 --stat` # показать последние 2 коммита и статистику внесенных ими изменений

`git log -p index.html` # показать историю изменений файла index.html (коммиты и изменения)

`git log master..branch_99` # показать коммиты из ветки branch_99, которые не влиты в master

`git show 60d6582` # показать изменения из коммита с указанным хешем

Ветки

`git branch` # показать список веток

`git branch -v` # показать список веток и последний коммит в каждой

`git branch new_branch` # создать новую ветку с указанным именем на текущем коммите

`git branch new_branch 5589877` # создать новую ветку с указанным именем на указанном коммите

`git branch -f master 5589877` # переместить ветку master на указанный коммит

`git checkout new_branch` # перейти в указанную ветку

`git checkout -b new_branch` # создать новую ветку с указанным именем и перейти в неё

`git merge hotfix` # влить в ветку, в которой находимся, данные из ветки hotfix

git branch --merged # показать ветки, уже слитые с активной

git branch --no-merged # показать ветки, не слитые с активной

git branch -a # показать все имеющиеся ветки (в т.ч. на удаленных репозиториях)

git branch -m old_branch_name new_branch_name # переименовать локально ветку

old_branch_name в new_branch_name

git push origin :old_branch_name new_branch_name # применить переименование в удаленном репозитории

Теги

git tag v1.0.0 # создать тег с указанным именем на коммите, на который указывает HEAD

git tag -a -m 'В продакшен!' v1.0.1 master # создать тег с описанием на том коммите, на который смотрит ветка master

git tag -d v1.0.0 # удалить тег с указанным именем(ами)

git tag -n # показать все теги, и по 1 строке сообщения коммитов, на которые они указывают

Временное сохранение изменений без коммита

git stash # временно сохранить незакоммиченные изменения и убрать их из рабочей директории

Удалённые репозитории

git remote -v # показать список удалённых репозиториях, связанных с локальным

git branch -a # показать все ветки(локальные и удаленные)

git remote rm origin # удалить привязку удалённого репозитория

git fetch origin # скачать все ветки с удаленного репозитория, но не сливать со своими ветками

git push origin master # отправить в удалённый репозиторий (с сокр. именем origin) данные своей ветки master

git pull origin master # влить изменения с удалённого репозитория (только указанная ветка)

Конфликт слияния

git merge feature # влить в активную ветку изменения из ветки feature

GIT FLOW

- 1) Клонировать репозиторий партнера для работы с ним и последующего pull requesta

```
C:\LabsGit\Lab2>git clone https://github.com/YuraGafurov/Laba2
Cloning into 'Laba2'...
warning: You appear to have cloned an empty repository.

C:\LabsGit\Lab2>cd Laba2
```

- 2) Инициализировать gitflow, создаётся ветка develop

```
C:\LabsGit\Lab2\Laba2>git flow init
No branches exist yet. Base branches must be created now.
Branch name for production releases: [master]
Branch name for "next release" development: [develop]

How to name your supporting branch prefixes?
Feature branches? [feature/]
Bugfix branches? [bugfix/]
Release branches? [release/]
Hotfix branches? [hotfix/]
Support branches? [support/]
Version tag prefix? []
Hooks and filters directory? [C:/LabsGit/Lab2/Laba2/.git/hooks]

C:\LabsGit\Lab2\Laba2>
```

- 3) Создать ветку ReadMeFeature, туда я добавлю файл ReadMe

```
C:\LabsGit\Lab2\Laba2>git flow feature start ReadMeFeature
Switched to a new branch 'feature/ReadMeFeature'

Summary of actions:
- A new branch 'feature/ReadMeFeature' was created, based on 'develop'
- You are now on branch 'feature/ReadMeFeature'

Now, start committing on your feature. When done, use:

    git flow feature finish ReadMeFeature

C:\LabsGit\Lab2\Laba2>git branch
  develop
* feature/ReadMeFeature
  master

C:\LabsGit\Lab2\Laba2>
```

- 4) Создать файл ReadMe, добавляю его в репозиторий и делаю commit

Этот компьютер > Acer (C:) > LabsGit > Lab2 > Laba2

	Имя	Дата изменения	Тип	Размер
оступ	.git	09.10.2022 17:48	Папка с файлами	
стол	README.md	09.10.2022 17:53	Исходный файл Маг...	1 КБ
пы				
ения				
зг				
-form				
Personal				
ютер				

Hello, my name is Timur

and I'm contributing to Yura's repo

```
C:\LabsGit\Lab2\Laba2>git status
On branch feature/ReadMeFeature
Untracked files:
  (use "git add <file>..." to include in what will be committed)

    README.md

nothing added to commit but untracked files present (use "git add" to track)

C:\LabsGit\Lab2\Laba2>git add README.md

C:\LabsGit\Lab2\Laba2>git commit -m "added README"
[feature/ReadMeFeature 98d9c4e] added README
 1 file changed, 2 insertions(+)
 create mode 100644 README.md

C:\LabsGit\Lab2\Laba2>
```

- 5) Завершаю работу с веткой feature – ветка сливается в develop и после этого удаляется

```
C:\LabsGit\Lab2\Laba2>git flow feature finish ReadMeFeature
Switched to branch 'develop'
Updating 465899f..98d9c4e
Fast-forward
 README.md | 2 ++
 1 file changed, 2 insertions(+)
 create mode 100644 README.md
Deleted branch feature/ReadMeFeature (was 98d9c4e).

Summary of actions:
- The feature branch 'feature/ReadMeFeature' was merged into 'develop'
- Feature branch 'feature/ReadMeFeature' has been locally deleted
- You are now on branch 'develop'
```


- 6) Создаю ветку Release1, где будет релиз версии 1.0 нашего проекта

```
C:\LabsGit\Lab2\Laba2>git branch
* develop
  master

C:\LabsGit\Lab2\Laba2>git flow release start Release1
Switched to a new branch 'release/Release1'

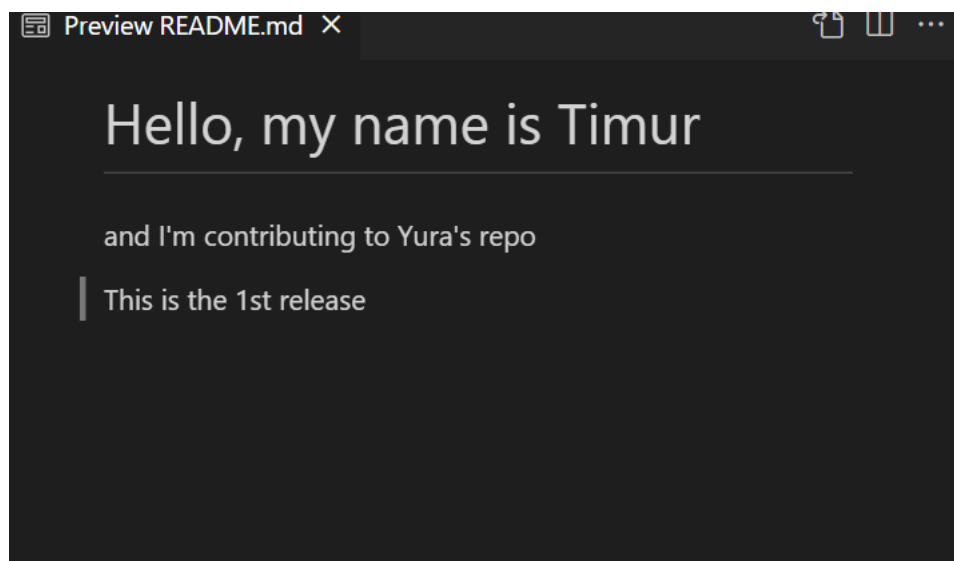
Summary of actions:
- A new branch 'release/Release1' was created, based on 'develop'
- You are now on branch 'release/Release1'

Follow-up actions:
- Bump the version number now!
- Start committing last-minute fixes in preparing your release
- When done, run:

    git flow release finish 'Release1'

C:\LabsGit\Lab2\Laba2>_
```

- 7) Изменяю файл ReadMe и коммичу это. Также ставлю тег на этот коммит, сообщаящий что это версия 1.0



```
C:\LabsGit\Lab2\Laba2>git branch
  develop
  master
* release/Release1

C:\LabsGit\Lab2\Laba2>git add .

C:\LabsGit\Lab2\Laba2>git commit -m "1st release commit"
[release/Release1 1659140] 1st release commit
1 file changed, 3 insertions(+), 1 deletion(-)

C:\LabsGit\Lab2\Laba2>git tag -a v1.0 -m "release version 1.0"

C:\LabsGit\Lab2\Laba2>_
```

8) Сливаю ветку релиза и мастер вручную

```
C:\LabsGit\Lab2\Laba2>git checkout master
Switched to branch 'master'
Your branch is based on 'origin/master', but the upstream is gone.
  (use "git branch --unset-upstream" to fixup)

C:\LabsGit\Lab2\Laba2>git merge release/Release1
Updating 465899f..1659140
Fast-forward
 README.md | 4 ++++
 1 file changed, 4 insertions(+)
 create mode 100644 README.md

C:\LabsGit\Lab2\Laba2>
C:\LabsGit\Lab2\Laba2>git log --all --graph
* commit 1659140bdf886b667421678641258ca6c6988140 (HEAD -> master, tag: v1.0, release/Release1)
  Author: TGontar <gontar.ts@mail.ru>
  Date:   Sun Oct 9 18:18:53 2022 +0300

    1st release commit

* commit 98d9c4e77cf75be5c5d09d3c3b57d05701b7df46 (develop)
  Author: TGontar <gontar.ts@mail.ru>
  Date:   Sun Oct 9 17:55:06 2022 +0300

    added README

* commit 465899f74fc786b141694418165a50fbd3e07579
  Author: TGontar <gontar.ts@mail.ru>
  Date:   Sun Oct 9 17:40:38 2022 +0300

    Initial commit
```

9) Завершаю работу с веткой релиз, результат сливается в develop

```
C:\LabsGit\Lab2\Laba2>git chekout develop
git: 'chekout' is not a git command. See 'git --help'.

The most similar command is
  checkout

C:\LabsGit\Lab2\Laba2>git checkout develop
Switched to branch 'develop'

C:\LabsGit\Lab2\Laba2>git flow release finish Release1
Switched to branch 'master'
Your branch is based on 'origin/master', but the upstream is gone.
  (use "git branch --unset-upstream" to fixup)
Switched to branch 'develop'
Merge made by the 'recursive' strategy.
 README.md | 4 +++-
 1 file changed, 3 insertions(+), 1 deletion(-)
Deleted branch release/Release1 (was 1659140).

Summary of actions:
- Release branch 'release/Release1' has been merged into 'master'
- The release was tagged 'Release1'
- Release tag 'Release1' has been back-merged into 'develop'
- Release branch 'release/Release1' has been locally deleted
- You are now on branch 'develop'

C:\LabsGit\Lab2\Laba2>
```

10) Запускаю ветку hotfix

```
C:\LabsGit\Lab2\Laba2>git flow hotfix start ReadMeFix
Switched to a new branch 'hotfix/ReadMeFix'

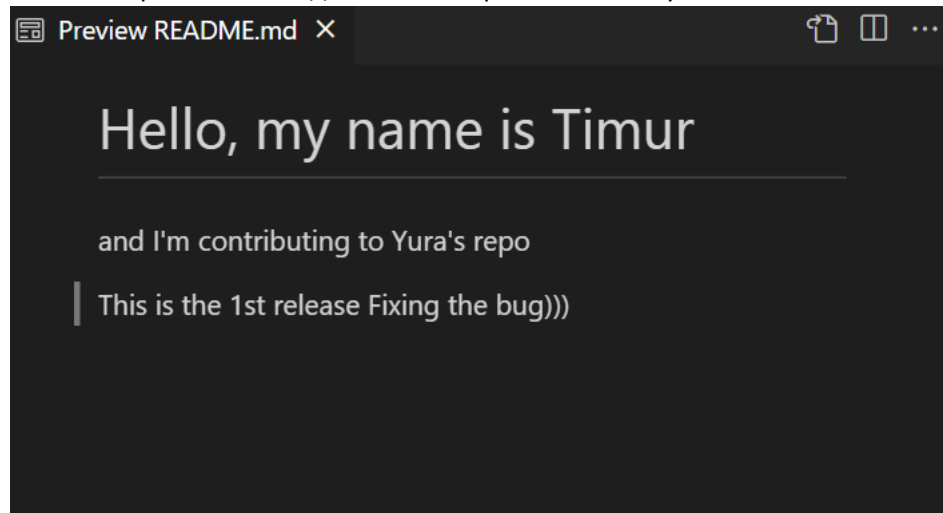
Summary of actions:
- A new branch 'hotfix/ReadMeFix' was created, based on 'master'
- You are now on branch 'hotfix/ReadMeFix'

Follow-up actions:
- Start committing your hot fixes
- Bump the version number now!
- When done, run:

    git flow hotfix finish 'ReadMeFix'

C:\LabsGit\Lab2\Laba2>
```

11) Изменяю файл ReadMe для ветки хотфикс и коммичу это



The screenshot shows a code editor window titled "Preview README.md". The content of the README file is as follows:

```
Hello, my name is Timur
```

```
and I'm contributing to Yura's repo

| This is the 1st release Fixing the bug)))
```

Below the editor window, a terminal window shows the following commands and output:

```
C:\LabsGit\Lab2\Laba2>git branch
develop
* hotfix/ReadMeFix
master

C:\LabsGit\Lab2\Laba2>git add .

C:\LabsGit\Lab2\Laba2>git commit -m "fixed README bug"
[hotfix/ReadMeFix fa70bd9] fixed README bug
1 file changed, 2 insertions(+), 1 deletion(-)

C:\LabsGit\Lab2\Laba2>
```

- 12) Завершаю работу с хотфиксом, результат сливается в master и develop

```
C:\LabsGit\Lab2\Laba2>git flow hotfix finish ReadMeFix
Switched to branch 'master'
Your branch is based on 'origin/master', but the upstream is gone.
(use "git branch --unset-upstream" to fixup)
Merge made by the 'recursive' strategy.
 README.md | 3 ++-
 1 file changed, 2 insertions(+), 1 deletion(-)
Switched to branch 'develop'
Merge made by the 'recursive' strategy.
 README.md | 3 ++-
 1 file changed, 2 insertions(+), 1 deletion(-)
Deleted branch hotfix/ReadMeFix (was fa70bd9).

Summary of actions:
- Hotfix branch 'hotfix/ReadMeFix' has been merged into 'master'
- The hotfix was tagged 'ReadMeFix'
- Hotfix tag 'ReadMeFix' has been back-merged into 'develop'
- Hotfix branch 'hotfix/ReadMeFix' has been locally deleted
- You are now on branch 'develop'
```

- 13) Добавляю подмодуль (submodule), им будет выступать репозиторий из предыдущей лабораторной

```
C:\LabsGit\Lab2\Laba2>git submodule add https://github.com/smartiqaorg/geometric_lib
Cloning into 'C:/LabsGit/Lab2/Laba2/geometric_lib'...
remote: Enumerating objects: 35, done.
remote: Counting objects: 100% (18/18), done.
remote: Compressing objects: 100% (12/12), done.
remote: Total 35 (delta 8), reused 6 (delta 6), pack-reused 17
Unpacking objects: 100% (35/35), done.
warning: LF will be replaced by CRLF in .gitmodules.
The file will have its original line endings in your working directory

C:\LabsGit\Lab2\Laba2>git status
On branch develop
Changes to be committed:
  (use "git reset HEAD <file>..." to unstage)

    new file:   .gitmodules
    new file:   geometric_lib

C:\LabsGit\Lab2\Laba2>
```

- 14) Ставлю LFS – расширение будет отслеживать и хранить в отдельном хранилище все файлы типа md

```
C:\LabsGit\Lab2\Laba2>git lfs track "*.md"
Tracking "*.md"

C:\LabsGit\Lab2\Laba2>

C:\LabsGit\Lab2\Laba2>git add .gitattributes
```

15) Делаю fork проекта партнера, чтобы залить создать pull request

The image consists of two screenshots of the GitHub web interface, illustrating the process of forking a repository.

Top Screenshot: Creating a new fork

The browser address bar shows `github.com/YuraGafurov/Laba2/fork`. The page title is "Create a new fork". Below the title, it says: "A fork is a copy of a repository. Forking a repository allows you to freely experiment with changes without affecting the original project."

The form for creating a fork includes:

- Owner:** A dropdown menu showing "TGontar".
- Repository name:** A text input field containing "Laba2" with a green checkmark.
- Description (optional):** An empty text input field.
- Copy the `develop` branch only:** A checked checkbox.

Below the form, it says: "Contribute back to YuraGafurov/Laba2 by adding your own branch. [Learn more.](#)"

Bottom Screenshot: Forked repository page

The browser address bar shows `github.com/TGontar/Laba2`. The page title is "TGontar / Laba2". Below the title, it says: "forked from YuraGafurov/Laba2".

The page shows the repository details for "Laba2":

- Branches:** A dropdown menu showing "develop", "1 branch", and "0 tags".
- Files:** A section showing "This branch is up to date with YuraGafurov/Laba2:develop." and a "Sync fork" button.
- Commits:** A section showing "YuraGafurov Initial commit" with a commit hash "e2c897c" and a timestamp "15 hours ago".
- README:** A section with the text "Help people interested in this repository understand your project by adding a README." and an "Add a README" button.
- About:** A section with the text "No description, website, or topics provided." and statistics: "0 stars", "0 watching", and "1 fork".
- Releases:** A section with the text "No releases published" and a "Create a new release" button.
- Packages:** A section with the text "No packages published".

16) Пушу все изменения в этот fork и создаю pull request

The screenshot displays two GitHub pages. The top page shows the repository **TGontar / Laba2**, which is a fork of **YuraGafurov / Laba2**. The repository has 2 branches, 3 tags, and 1 commit behind the parent. The README.md file is visible, containing the text: "Hello, my name is Timur", "and I'm contributing to Yura's repo", and "This is the 1st release Fixing the bug)))".

The bottom page shows a pull request titled **Changing Yura's repo #1** from **TGontar** to **YuraGafurov / Laba2**. The pull request is open and shows 8 commits. The commit history includes: "Initial commit", "added README", and "1st release commit". The pull request is currently in progress, with no reviews or assignees.

17) Партнёр принимает pull request. Работа с репозиторием окончена

The screenshot shows a GitHub interface for the repository **YuraGafurov / Laba2**. The top navigation bar includes links for Pull requests, Issues, Marketplace, and Explore. The repository page shows a pull request titled "Changing Yura's repo" by TGontar, which has been merged. The repository page also displays the README content: "Hello, my name is Timur" and "This is the 1st release Fixing the bug)))".

Filters: is:pr is:closed

0 Open 1 Closed

Author Label Projects Milestones Reviews Assignee Sort

Changing Yura's repo
#1 by TGontar was merged in 2 hours

ProTip! Add no:assignee to see everything that's not assigned.

© 2022 GitHub, Inc. Terms Privacy Security Status Docs Contact GitHub Pricing API Training Blog About

develop 2 branches 0 tags

Go to file Add file Code

YuraGafurov Merge pull request #1 from TGontar/develop 77af6e2 in 2 hours 10 commits

README.md fixed README bug 1 hour ago

README.md

Hello, my name is Timur

and I'm contributing to Yura's repo

This is the 1st release Fixing the bug)))

About

No description, website, or topics provided.

Readme

0 stars

1 watching

1 fork

Releases

No releases published

Packages

No packages published