**01 Git: branches & Postman**

**Создание репозитория, клонирование на локальный ПК, экспорт коллекций из Postman, push, работа с ветками.**

Выполнил: Герман Буйницкий

**Содержание:**

[**Задача:** 1](#_Toc70434585)

[**Предварительная настройка:** 2](#_Toc70434586)

[**Решение:** 2](#_Toc70434587)

[**Приложение 1. Описание всех эндпоинтов Postman из задания** 12](#_Toc70434588)

# **Задача:**

1) Выполнить в Postman endpoints 1,2,7,8.

endpoint\_1: object\_info\_1

endpoint\_2: object\_info\_2

endpoint\_7: user\_info\_3

endpoint\_8: user\_info\_4

2) Сделать репозиторий в Гит

3) Склонировать репозиторий на локальный компьютер.

4) Експортировать тесты endpoints 1,2,7,8 и окружение в папку git склонированного репозитория.

5) Запушить тесты на внешний репозиторий

6) Сделать в GIT отдельную ветку.

7) Сделать тесты ендроинтов 3, 6 в Postman

endpoint\_3: object\_info\_3

endpoint\_6: user\_info\_2

8) Экспортировать тесты в новую, созданную ветку GIT.

9) Запушить новую ветку на внешний репозиторий. Тесты должны появиться на внешнем репозитории созданной в ветке

10) Вмержить изменения новой ветки в основную.

11) Запушить изменения основной ветки на внешний репозиторий.

12) Сделать в GIT отдельную ветку.

13) Сделать тесты ендпоинтов 4,5 в Postman

endpoint\_4: object\_info\_4

endpoint\_5: user\_info\_1

14) Экспортировать тесты в созданную ветку GIT.

15) Запушить новую ветку на внешний репозиторий. Тесты должны появиться на внешнем репозитории созданной в ветке.

16) Вмержить изменения ветки в основную ветку.

17) Запушить изменения основной ветки на внешний репозиторий.

# **Предварительная настройка:**

В Git Bash вести команду

ssh-keygen -t rsa -C "Username"

Далее просто жмем Enter и получаем ssh-ключ.

SSH ключ лежит в файле id\_rsa.pub, который лежит в папке .ssh

Открыть файл id\_rsa.pub и скопировать из него ключ весь от начала до конца

Открыть наш GitHub

Кликнуть по иконке аккаунта справа в углу -> Settings -> SSH and GPG keys

Придумать любое имя для ключа и ввести его в соответствующее поле.

Также нужно поставить на локальный гит несколько настроек:

git config --global user.name "Username"

git config --global user.email "Usermail@gmail.com"

# **Решение:**

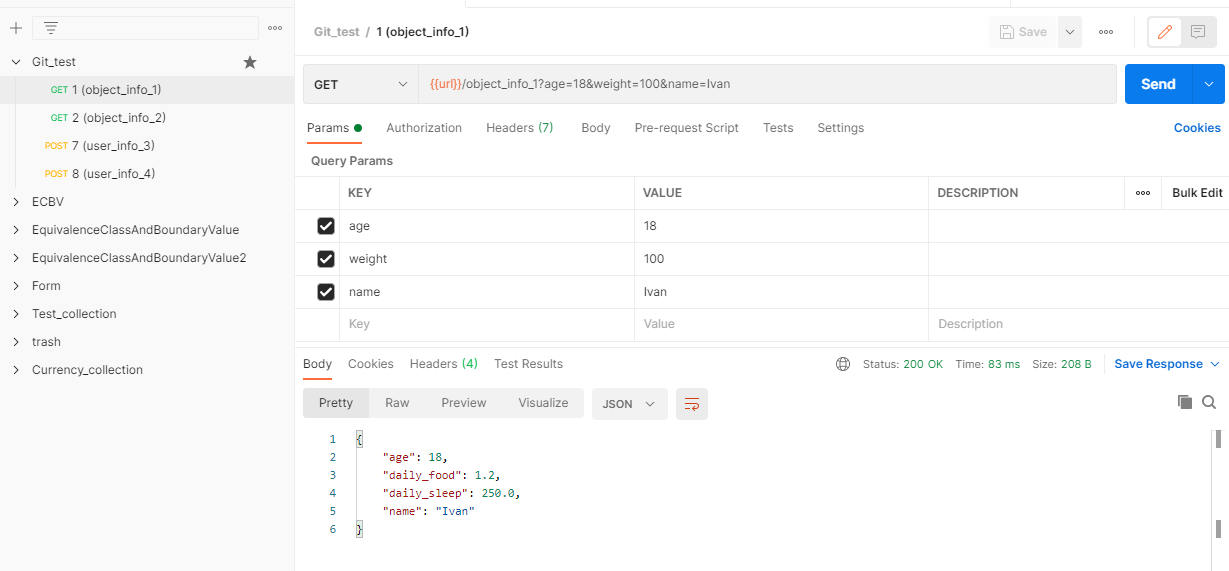
**1) Выполнить в Postman endpoints 1,2,7,8:**

endpoint\_1: object\_info\_1

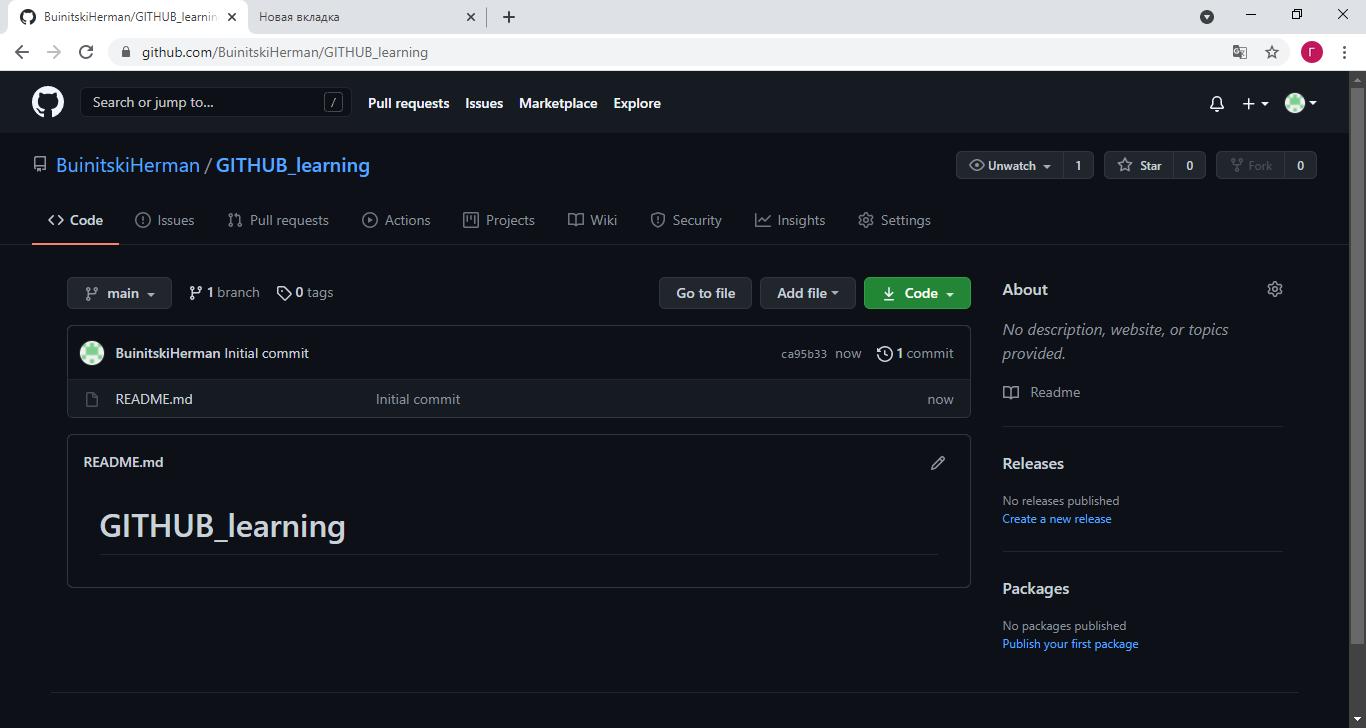
endpoint\_2: object\_info\_2

endpoint\_7: user\_info\_3

endpoint\_8: user\_info\_4



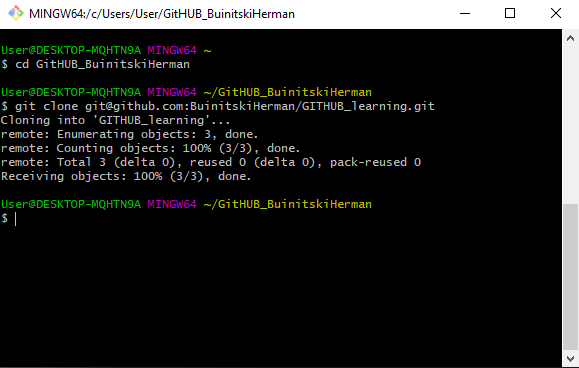
**2) Сделать репозиторий в Гит:**



**3) Склонировать репозиторий на локальный компьютер:**

Перейдем в рабочую папку локального Гит и воспользуемся командой

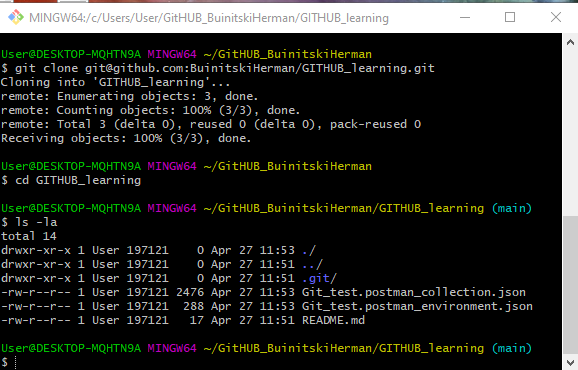
git clone git@github.com/GitHub\_username/Repository\_name.git



**4) Экспортировать тесты endpoints 1,2,7,8 и окружение в папку git склонированного репозитория:**

Экспортируем из Postman два файла:

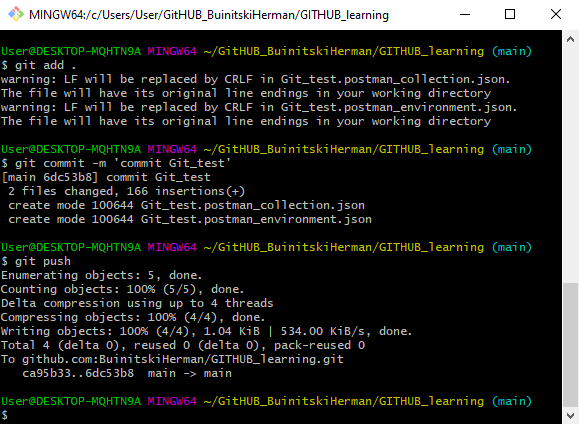
* 1й файл json с коллекцией из 4х запросов (Нажать три точки возле названия коллекции – Export - Collection v2.1 (recommended) - Export).
* 2й файл json с окружением (Значок глаза [просмотр окружения] – Edit – Значок “три точки” - Export).



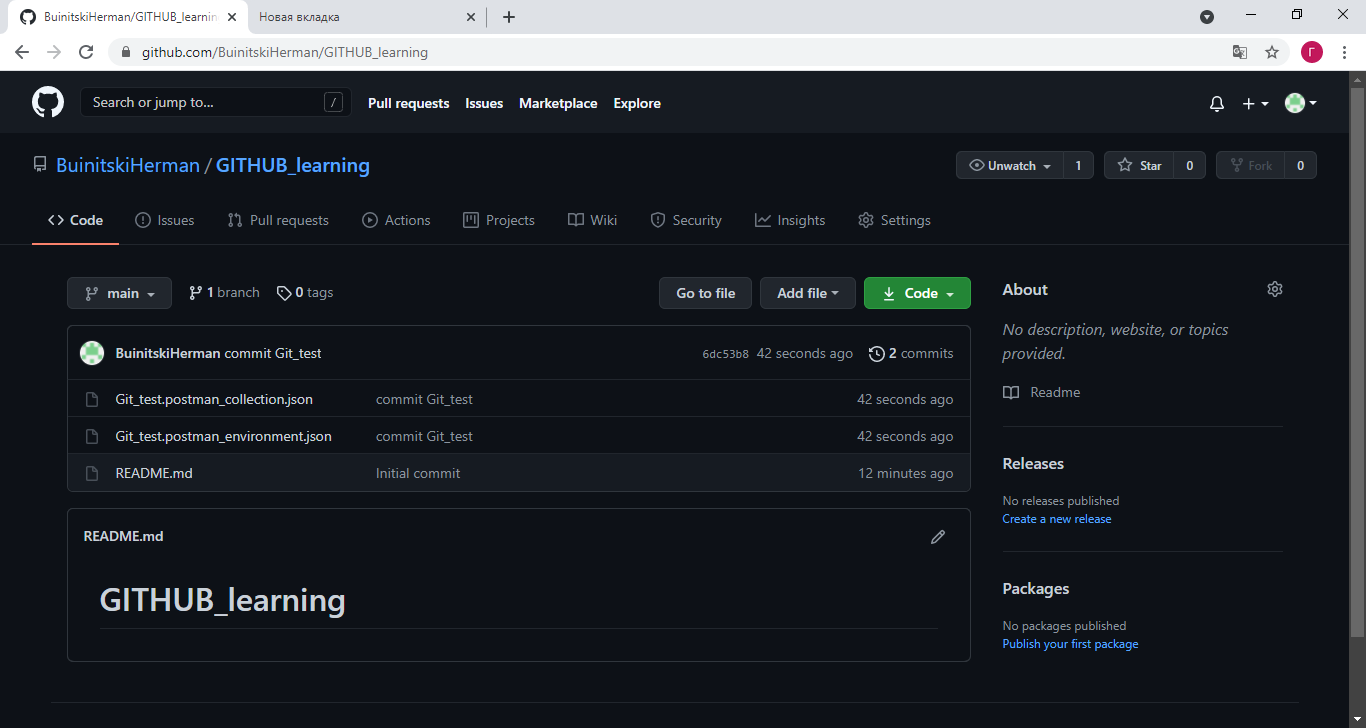
**5) Запушить тесты на внешний репозиторий**:

Используем команду

git push



На внешнем репозитории отображаются новые файлы:

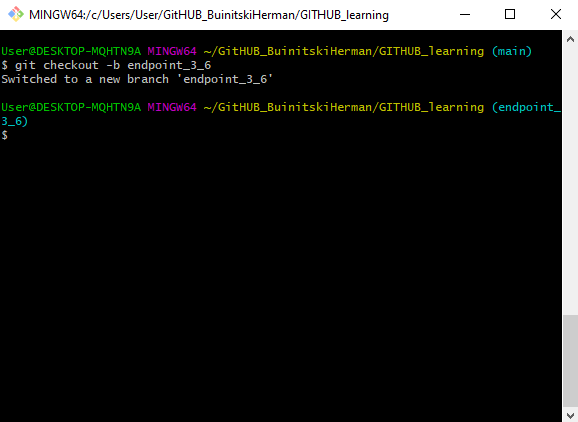


**6) Сделать в GIT отдельную ветку:**

Используем команду

git checkout –b new\_branch\_name

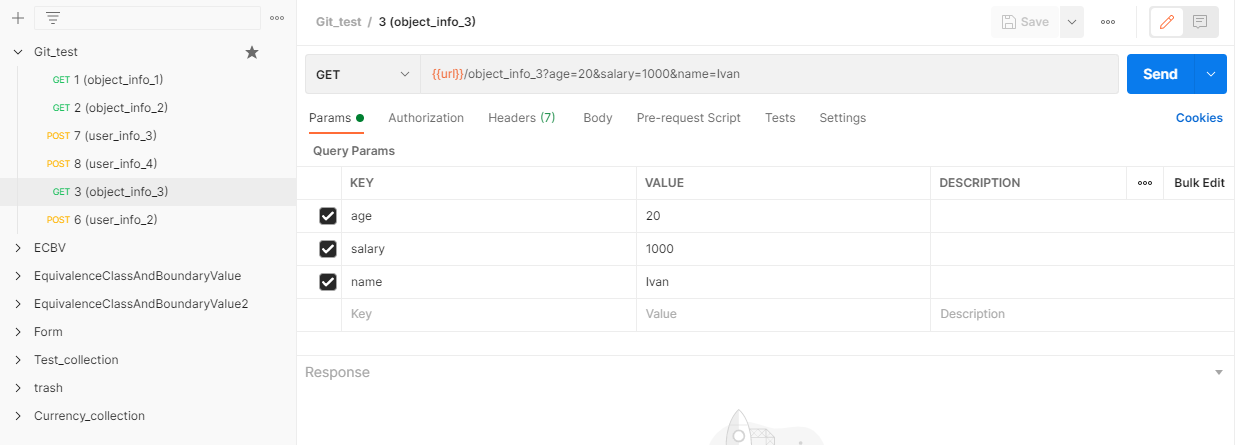
Создается ветка с указанным именем и мы заходим в нее (на скрине видно, как main поменялось на endpoint\_3\_6)



**7) Сделать тесты ендроинтов 3, 6 в Postman:**

endpoint\_3: object\_info\_3

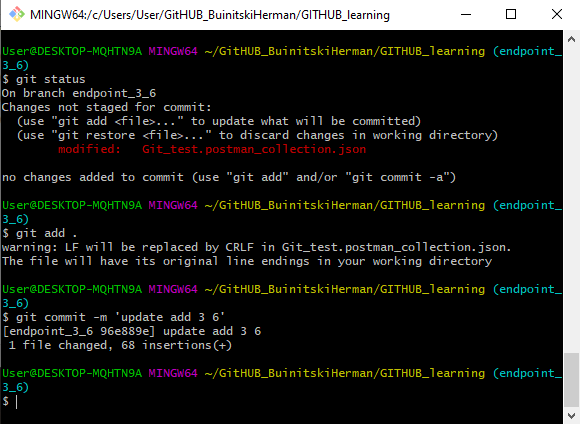
endpoint\_6: user\_info\_2



**8) Экспортировать тесты в новую, созданную ветку GIT:**

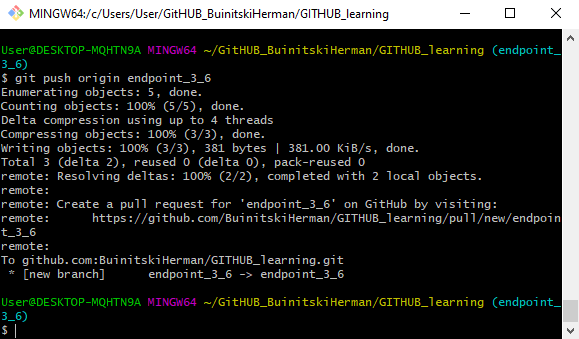
git add .

git commit –m ‘commit\_name’

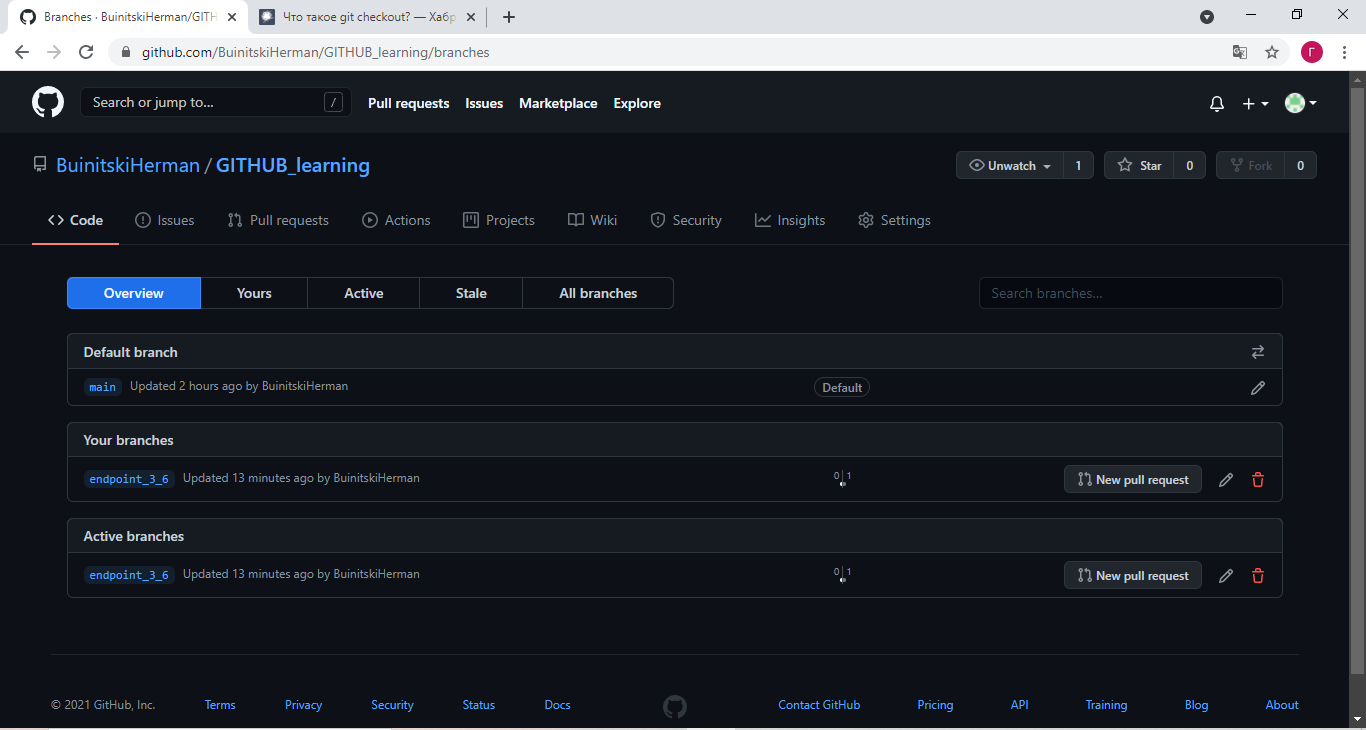


**9) Запушить новую ветку на внешний репозиторий. Тесты должны появиться на внешнем репозитории созданной в ветке:**

git push origin branch\_name



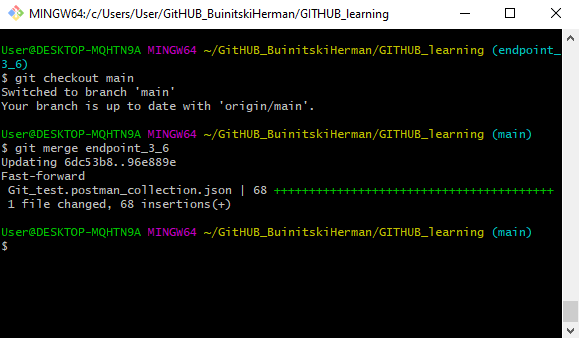
На внешнем репозитории теперь две ветки, в которых есть разные версии тестов. На ветке main тесты запросов 1, 2, 7, 8. На ветке endpoint\_3\_6 к запросам 1, 2, 7, 8 добавлены запросы 3 и 6.



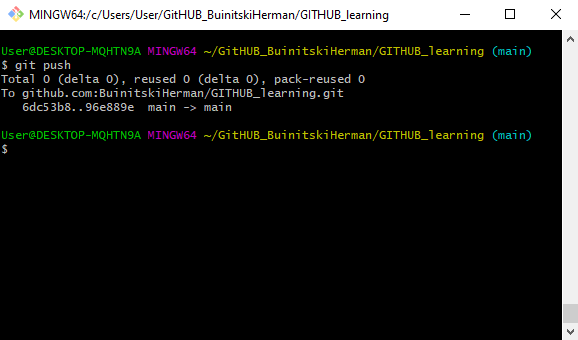
**10) Вмержить изменения новой ветки в основную:**

Переходим в основную ветку и используем команду

git merge branch\_name



**11) Запушить изменения основной ветки на внешний репозиторий:**

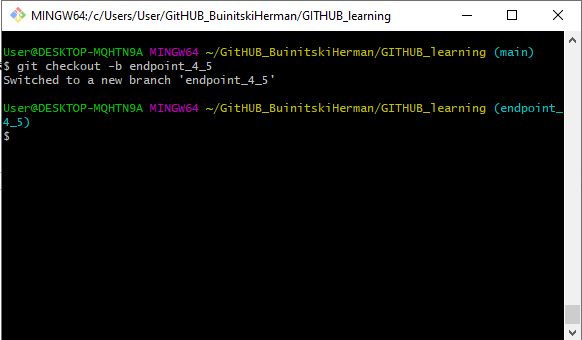


**12) Сделать в GIT отдельную ветку:**

По аналогии с пунктом 6 используем команду

git checkout –b new\_branch\_name

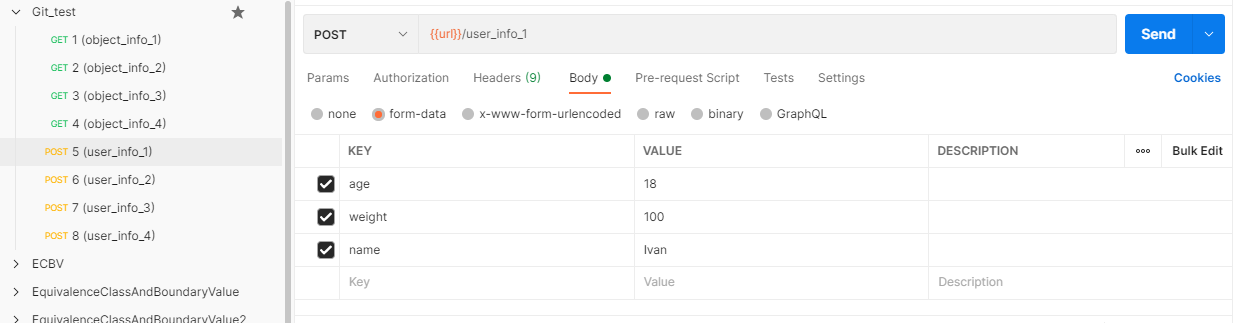
Создается ветка с указанным именем и мы заходим в нее (на скрине видно, как main поменялось на endpoint\_4\_5).



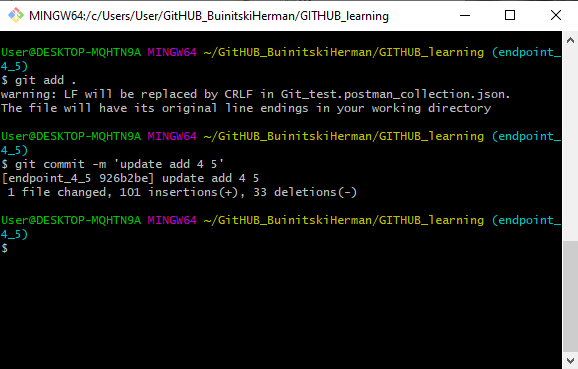
**13) Сделать тесты ендпоинтов 4,5 в Postman:**

endpoint\_4: object\_info\_4

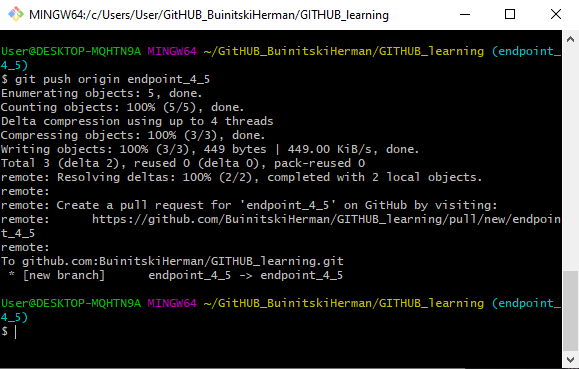
endpoint\_5: user\_info\_1



**14) Экспортировать тесты в созданную ветку GIT:**



**15) Запушить новую ветку на внешний репозиторий. Тесты должны появиться на внешнем репозитории созданной в ветке:**

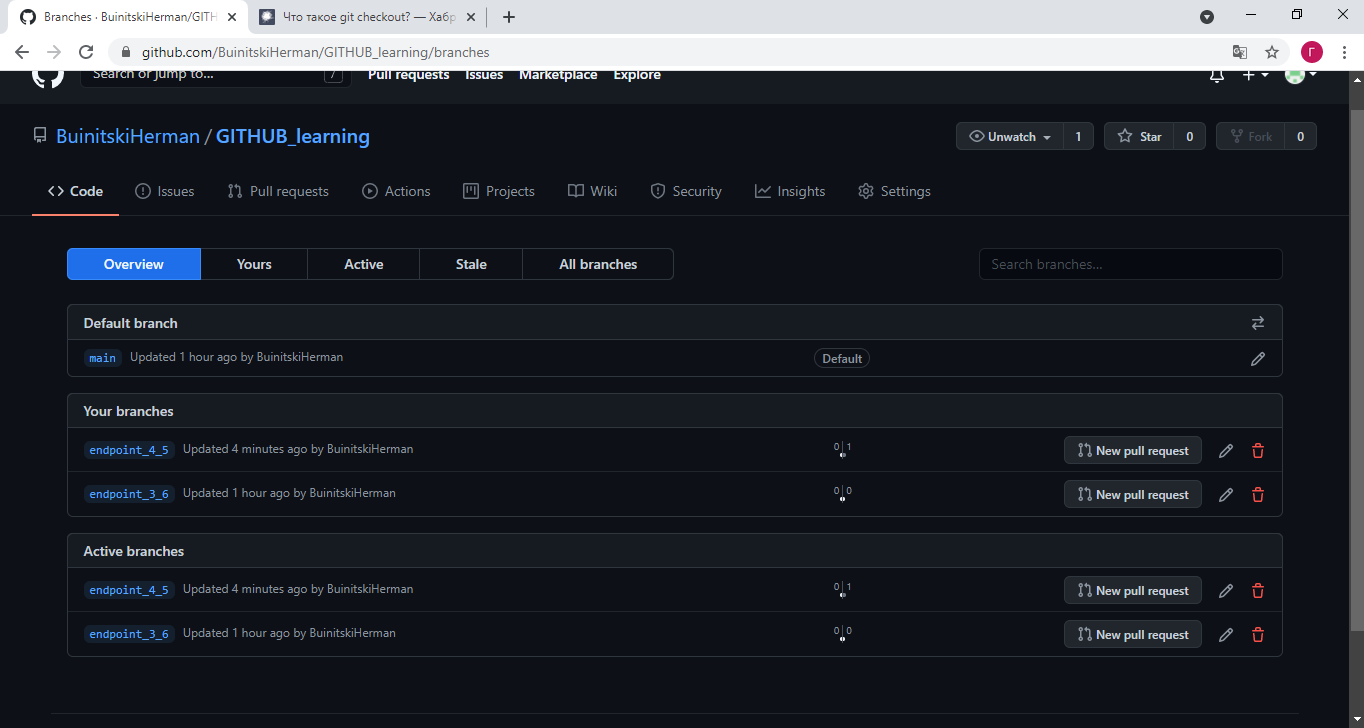


На внешнем репозитории теперь три ветки.

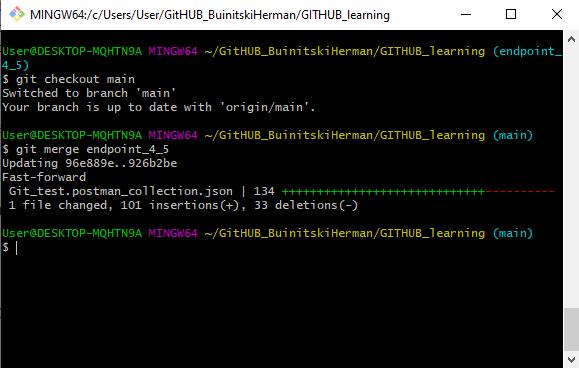
На ветке main тесты запросов 1, 2, 7, 8, 3, 6 (результат merge с endpoint\_3\_6).

На ветке endpoint\_3\_6 к запросам 1, 2, 7, 8, 3, 6.

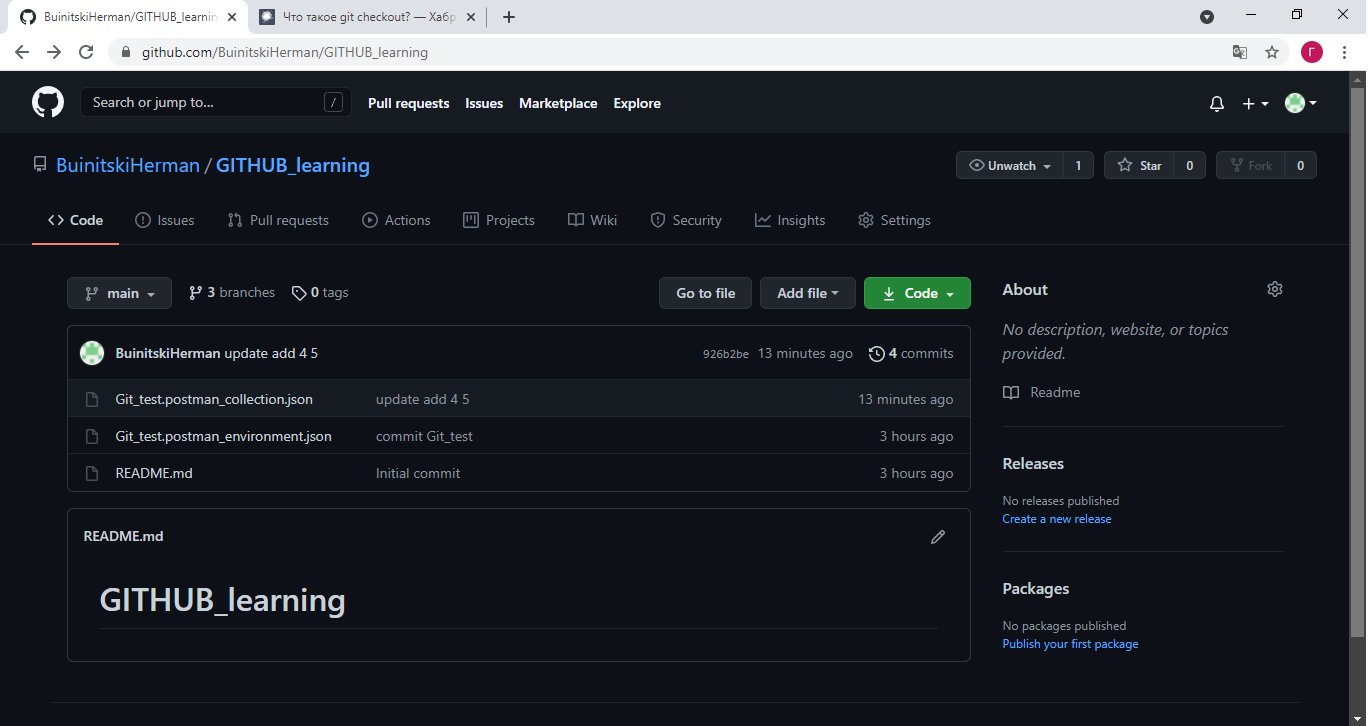
На ветке endpoint\_4\_5 запросы 1, 2, 3, 4, 5, 6, 7, 8.



**16) Вмержить изменения ветки в основную ветку:**



**17) Запушить изменения основной ветки на внешний репозиторий:**



*Обновленная ветка main, в репозитории всего 3 ветки*

# **Приложение 1. Описание всех эндпоинтов Postman из задания**

endpoint\_1: object\_info\_1

method: GET

request\_params:

1) age (int)

2) weight (int)

3) name (str)

response:

{'name': name,

'age': age,

'daily\_food': weight \* 0.012,

'daily\_sleep': weight \* 2.5}

========

endpoint\_2: object\_info\_2

method: GET

request\_params:

1) age (int)

2) salary (int)

3) name (str)

response:

{'start\_qa\_salary': salary,

'qa\_salary\_after\_6\_months': salary \* 2,

'qa\_salary\_after\_12\_months': salary \* 2.7,

'qa\_salary\_after\_1.5\_year': salary \* 3.3,

'qa\_salary\_after\_3.5\_years': salary \* 3.8,

'person': {'u\_name': [user\_name, salary, age],

'u\_age': age,

'u\_salary\_5\_years': salary \* 4.2}

}

=========

endpoint\_3: object\_info\_3

method: GET

request\_params:

1) age (int)

2) salary (int)

3) name (str)

response:

result = {'name': name,

'age': age,

'salary': salary,

'family': {'children': [['Alex', 24], ['Kate', 12]],

'pets': {'cat':{'name':'Sunny',

'age': 3},

'dog':{'name':'Luky',

'age': 4}},

'u\_salary\_1\_5\_year': salary \* 4}

}

=========

endpoint\_4: object\_info\_4

method: GET

request\_params:

1) age (int)

2) name (str)

3) salary (int)

response:

result = {'name': name,

'age': int(age),

'salary': [salary, str(salary \* 2), str(salary \* 3)]}

============================

endpoint\_5: user\_info\_1

method: POST

form\_params:

1) age (int)

2) weight (int)

3) name (str)

response:

{'name': name,

'age': age,

'daily\_food': weight \* 0.012,

'daily\_sleep': weight \* 2.5}

========

endpoint\_6: user\_info\_2

method: POST

form\_params:

1) age (int)

2) salary (int)

3) name (str)

response:

{'start\_qa\_salary': salary,

'qa\_salary\_after\_6\_months': salary \* 2,

'qa\_salary\_after\_12\_months': salary \* 2.7,

'qa\_salary\_after\_1.5\_year': salary \* 3.3,

'qa\_salary\_after\_3.5\_years': salary \* 3.8,

'person': {'u\_name': [user\_name, salary, age],

'u\_age': age,

'u\_salary\_5\_years': salary \* 4.2}

}

=========

endpoint\_7: user\_info\_3

method: POST

form\_params:

1) age (int)

2) salary (int)

3) name (str)

response:

result = {'name': name,

'age': age,

'salary': salary,

'family': {'children': [['Alex', 24], ['Kate', 12]],

'u\_salary\_1\_5\_year': salary \* 4}

}

=========

endpoint\_8: user\_info\_4

method: POST

form\_params:

1) age (int)

2) name (str)

3) salary (int)

response:

result = {'name': name,

'age': int(age),

'salary': [salary, str(salary \* 2.5), str(salary \* 3.5)]}