**Theory:**

Создать новую ветку можно с помощью команды **git branch <название ветки>**. Важно, чтобы название было понятным для вас и других участников проекта.

Создадим новую ветку для проекта: **git branch input.**

Проверим **git status,** убедимся, что рабочее дерево чистое. Проверим историю: **git log.** Видим весь путь работы: последнее изменение и все изменения, которые были еще до ответвления. Чтобы выйти из истории, нужно нажать **q** в английской раскладке.

**git branch**

**git branch branches\_lesson**

**git checkout branches\_lesson**

**git branch apple\_pay**

* **git checkout -b branch\_1** (*Чтобы создать ветку и переключиться на нее за одно действие, можно воспользоваться флагом -b:* ***git checkout -b <название ветки>.****))*
* **git branch input (создать новую ветку)**
* **git branch (посмотреть на какой ты ветке)**
* cat >> cards.json

{"title":"Mastercard"}

* **git add .**
* **git commit -m "add cards"**
* **git checkout main( перейти на главную ветку)**

$ **git checkout branch\_1**

Switched to branch 'branch\_1'

$ **git checkout main**

Switched to branch 'main' (переключение между ветками)

* **git merge apple\_pay** - чтобы сделать git-merge, надо встать на ту ветку, в которую будут заливаться изменения и вызвать команду **Git merge +название ветки, которую будем мерджить**
* **vim cards.json / insert/ input/ esq/ :wq** (делаем изменения на главной ветке чтобы вызвать конфликт)
* **git commit -am "add visa to cards"** (добавить изменения под гит и закоммитить с комментарием)
* **git checkout apple\_pay**
* **vim cards.json**
* ****
* **git commit -am "add Maestro to cards"**
* **git checkout main**
* **git merge apple\_pay**

Auto-merging cards.json

CONFLICT (content): Merge conflict in cards.json

Automatic merge failed; fix conflicts and then commit the result.

* vim cards.json (удаляем конфликт)



* git commit -am "merge Maestro and Visa to cards"
* git push
* git push -u origin apple\_pay (ветка apple\_pay заливается на хитхаб!)

**HW\_3**

1. На локальном репозитории сделать ветки для:

- Postman

- Jmeter

- CheckLists

- Bag Reports

- SQL

- Charles

- Mobile testing

* Создать репозиторий в гитхаб HW\_repo
* Клонировать репозиторий в локальную папку
* Создать любой файл на ветке main ( cat >> my\_repo.txt)

git add .

git commit -m “...”

git push

git branch Postman

git branch BugReports

git branch Charles

git branch CheckLists

git branch Jmeter

git branch MobileTesting

git branch SQL



2. Запушить все ветки на внешний репозиторий

**git checkout \*ветка**

**cat >> \*File (создать любой файл)**

**git push origin --al**

3. В ветке Bag Reports сделать текстовый документ со структурой баг репорта

**cat >> BugReports**

**Если существует: vim BugReports**

4. Запушить структуру багрепорта на внешний репозиторий

**git add BugReports.txt**

**git commit -m “...”**

**git push --set-upstream origin BugReports**

5. Вмержить ветку Bag Reports в Main

**git checkout main**

**git merge BugReports**

6. Запушить main на внешний репозиторий.

**git push -u origin main (ветка main заливается на хитхаб!)**

7. В ветке CheckLists набросать структуру чек листа.

**vim checklists.txt**

8. Запушить структуру на внешний репозиторий

**git add checklist.txt**

**git commit -m “...”**

**git push --set-upstream origin CheckLists**

9. На внешнем репозитории сделать Pull Request ветки CheckLists в main

[Pull requests · egorovaolga888/HW\_repo (github.com)](https://github.com/egorovaolga888/HW_repo/pulls)

10. Синхронизировать Внешнюю и Локальную ветки Main

**git pull / git fetch ???**

**:wq! и Enter**