Министерство образования Республики Беларусь

Учреждение образования

«Белорусский государственный университет

информатики и радиоэлектроники»

Кафедра инженерной психологии и эргономики

Дисциплина: Пользовательские интерфейсы информационных систем

Отчет

по практической работе

на тему:

«GIT»

Проверил: Выполнила:

Студентка группы 210902

Давыдович К.И. Кицун Д.Д.

Минск 2024

**Git**

Репозиторий Git — это место, где хранится ваш проект и его история изменений. Коммит можно рассматривать как снимок состояния вашего проекта в определённый момент времени. Однако коммит фиксирует только те изменения, которые были внесены с момента последнего коммита.

Ветка представляет собой независимую версию вашего репозитория.

**Начало работы**

Чтобы начать, необходимо настроить ваше имя и адрес электронной почты:

1. git config --global user.name "ваше имя"
2. git config --global user.email "email@example.com"

**Создание нового репозитория**

Для инициализации нового репозитория используйте команду:

1. git init

Затем добавьте файлы и зафиксируйте изменения:

1. git add my.js
2. git commit -m "новый файл"

Чтобы внести изменения из всех файлов, используйте:

bash

Copy

git add .

Для отмены последнего добавления:

bash

Copy

git reset

Если изменения ещё не добавлены в индекс, отменить их можно с помощью:

bash

Copy

git checkout -- <имя файла>

**Просмотр состояния репозитория**

Чтобы узнать текущее состояние репозитория (например, какие файлы были изменены), используйте:

bash

Copy

git status

Если вы хотите объединить изменения нескольких файлов в один коммит, вы можете использовать:

bash

Copy

git add 1.js

git add 2.js

git commit -m "изменено 2 файла"

Также можно добавить изменения всех файлов сразу:

bash

Copy

git add .

**История изменений**

Для просмотра истории коммитов используйте:

bash

Copy

git log

Чтобы вернуться к определённому коммиту, выполните:

bash

Copy

git checkout <номер коммита>

Для переключения между ветками:

bash

Copy

git switch <название ветки>

**Работа с тегами**

Теги помогают быстро переключаться между версиями. Чтобы создать тег, используйте:

bash

Copy

git tag <название тега>

Для просмотра всех тегов:

bash

Copy

git tag

Чтобы отменить коммит, выполните:

bash

Copy

git revert <номер коммита>

Для отмены последнего коммита:

bash

Copy

git revert HEAD

Если нужно сбросить историю изменений до определённого коммита, выполните:

bash

Copy

git reset --hard <номер коммита>

**Важно:** не рекомендуется использовать эту команду на удалённых репозиториях.

Для удаления тега:

bash

Copy

git tag -d <название тега>

**Внесение изменений в последний коммит**

Чтобы внести изменения в последний коммит:

bash

Copy

git add my.js

git commit --amend -m "исправленный коммит"

**Создание и переключение веток**

Для создания новой ветки:

bash

Copy

git switch -c <название ветки>

Для переключения между ветками алгоритм такой же.

**Просмотр изменений в коммите**

Чтобы увидеть изменения в конкретном коммите, используйте:

bash

Copy

git show <номер коммита>

**Опции для git log**

* **--all**: показывает все ветки, так как по умолчанию отображается только текущая ветка.
* **--graph**: добавляет простую визуализацию дерева коммитов.

**Перемещение файлов**

Чтобы переместить файл в другую директорию:

bash

Copy

git mv my.js jses/my.js

**Объединение и переписывание изменений**

Для объединения изменений из одной ветки в другую:

bash

Copy

git merge main

Команда git rebase позволяет переносить коммиты с одной ветки на другую, упрощая историю проекта.

**Клонирование репозитория**

Чтобы клонировать удалённый репозиторий:

bash

Copy

git clone <название репозитория>

**Удалённые репозитории**

Для просмотра доступных удалённых репозиториев:

bash

Copy

git remote

Удалённые репозитории обозначаются как origin.

**Ветки и их просмотр**

Для просмотра веток в локальном репозитории:

bash

Copy

git branch

Для просмотра удалённых веток:

bash

Copy

git branch -a

**Загрузка изменений**

* **git fetch**: загружает изменения из удалённого репозитория в ваш локальный, не изменяя текущую рабочую ветку.
* **git push**: отправляет ваши локальные изменения в удалённый репозиторий.
* **git pull**: загружает последние изменения из удалённого репозитория и автоматически объединяет их с локальной веткой.