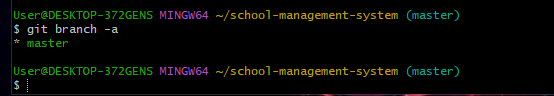
1. Просмотр всех веток (локальных и удалённых)

git branch -a

Пример вывода:

\* master



Это позволяет увидеть список всех веток, чтобы выбрать, какие можно удалить.

2. Удаление локальной ветки

После успешного слияния ветки feature/schedule в master, она больше не нужна.

git branch -d feature/schedule

Если ветка не была слита полностью , Git предупредит об этом.

Чтобы удалить принудительно:

git branch -D feature/schedule

Таким образом, временные ветки не захламляют репозиторий.

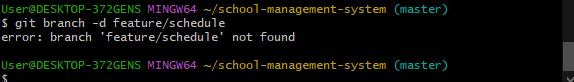
3. Удаление удалённой ветки

Если ветка уже была отправлена на GitHub или другой сервер — её нужно удалить там тоже:

git push origin :feature/schedule

или новая версия команды:

git push origin --delete feature/schedule

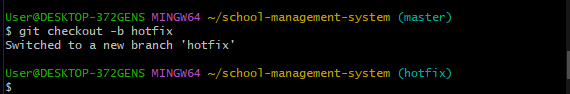


Эта команда удаляет ветку с удалённого репозитория.

4. Пример из твоего проекта: работа с веткой hotfix

Создана временная ветка hotfix для исправления ошибки в schedule.html.

git checkout -b hotfix



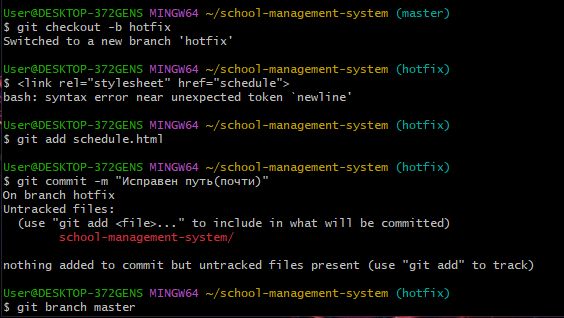
Внесены изменения:

<link rel="stylesheet" href="schedule">

Коммит:

git add schedule.html

git commit -m "Исправлен путь к CSS"



После проверки изменений ветка объединена с master:

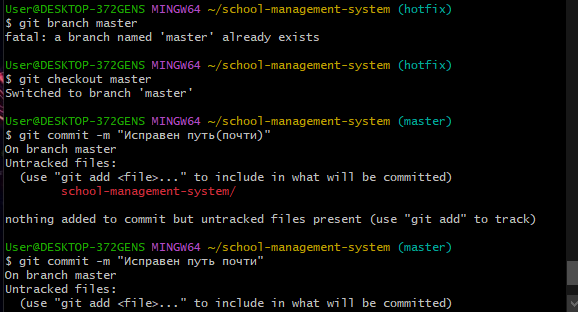
git checkout master

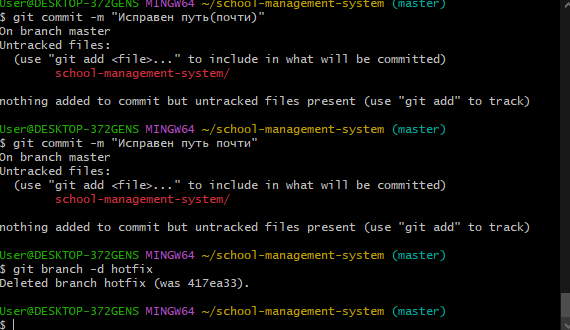
git merge hotfix

Затем ветка удалена:

git branch -d hotfix

git push origin --delete hotfix





Такой подход помогает держать репозиторий аккуратным и организованным.

5. Чистка после нескольких веток

Если было создано несколько веток:

feature/schedule

И они все были успешно слиты в master, их можно удалить:

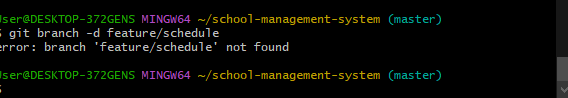
git branch -d feature/schedule

git branch -d feature/grades

git branch -d feature/diary

А также удалить на сервере:

git push origin --delete feature/schedule



После этого история будет выглядеть чисто и понятно.