УРОК 18. MERGE КОНФЛИКТЫ, MONOREPO

MERGE КОНФЛИКТЫ

Merge конфликты возникают в Git, когда Git не может автоматически слить изменения из разных веток из-за конфликта в изменениях в одном и том же файле или строке кода.

Виды merge конфликтов:

- Конфликты файлов: Когда две ветки вносят изменения в один и тот же файл, а изменения несовместимы. Например, изменения в одной ветке могут затрагивать строки кода, которые были удалены или изменены в другой ветке.
- Конфликты слияния директорий: Возникают, когда Git не может определить, как объединить изменения в директории из-за конфликтующих файлов.

Создание merge conflict:
Создадим новую директорию и инициализируем репозиторий в ней.
Создадим новый текстовый файл merge.txt с некоторым содержимым.
Добавим merge.txt в репозиторий и зафиксируем его.

```
Unset
$ mkdir git-merge-test
$ cd git-merge-test
$ git init .
$ echo "this is some content to mess with" > merge.txt
$ git add merge.txt
$ git commit -am"we are commiting the inital content"
[main (root-commit) d48e74c] we are commiting the inital content
1 file changed, 1 insertion(+)
create mode 100644 merge.txt
```

Теперь у нас есть новый репозиторий с одной веткой main и файлом merge.txt с содержимым. Далее мы создадим новую ветку, которая будет использоваться в качестве конфликтующего слияния.

```
Unset
$ git checkout -b new_branch_to_merge_later
```


```
$ echo "totally different content to merge later" > merge.txt
$ git commit -am"edited the content of merge.txt to cause a conflict"
[new_branch_to_merge_later 6282319] edited the content of merge.txt to cause a
conflict
1 file changed, 1 insertion(+), 1 deletion(-)
```

Теперь создадим ветку new_branch_to_merge_later и переключимся на нее.
Изменим содержимое файла merge.txt
Выполним commit.

```
Unset
git checkout main
Switched to branch 'main'
echo "content to append" >> merge.txt
git commit -am"appended content to merge.txt"
[main 24fbe3c] appended content to merge.tx
1 file changed, 1 insertion(+)
```

Теперь попробуем слить эти ветки:

```
Unset
$ git merge new_branch_to_merge_later
Auto-merging merge.txt
CONFLICT (content): Merge conflict in merge.txt
Automatic merge failed; fix conflicts and then commit the result.
```

Появляется конфликт. Git сообщили нам об этом!

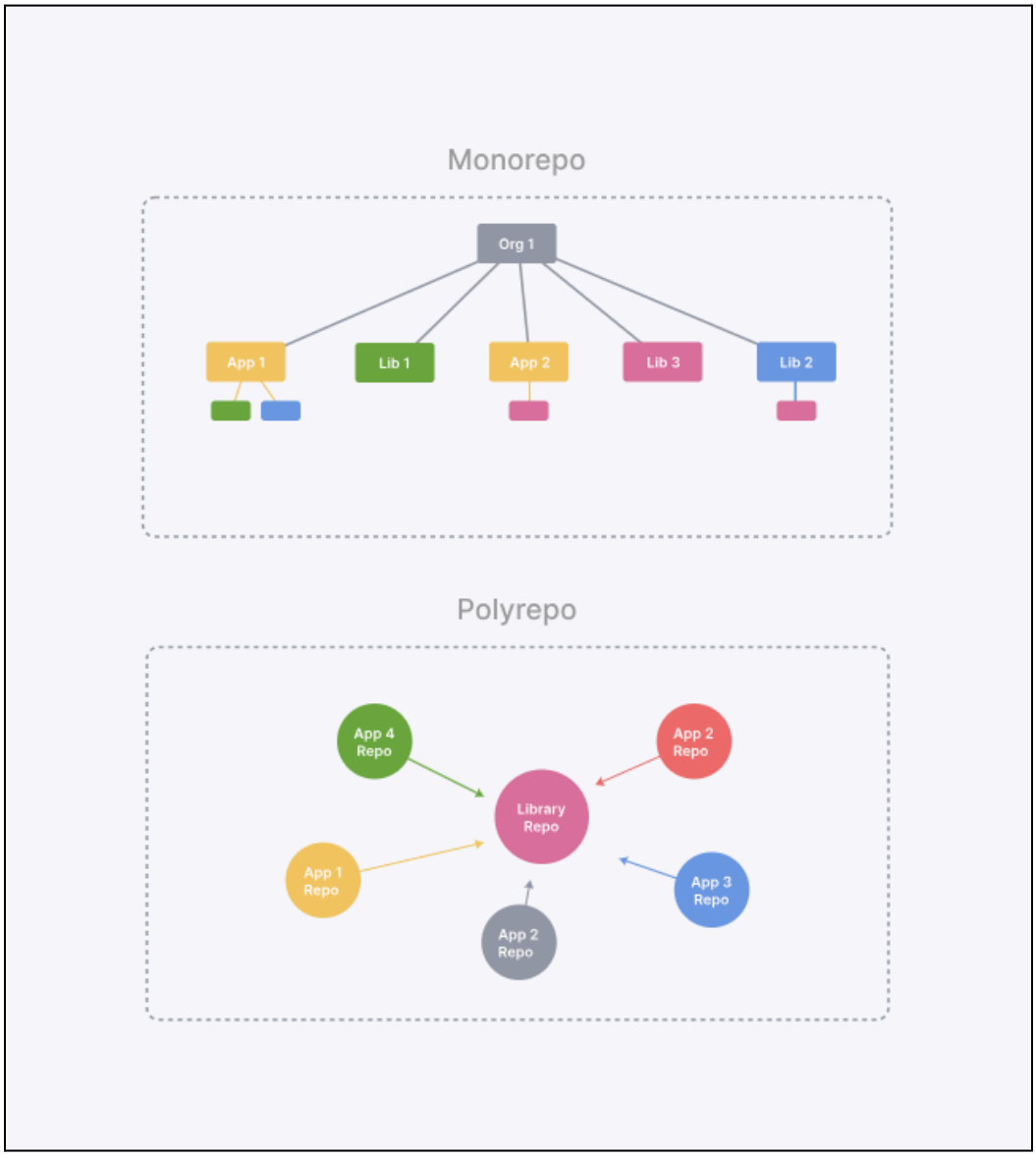
Решение merge конфликтов:

1. Ручное разрешение конфликтов: Откройте конфликтующий файл в текстовом редакторе и разрешите конфликт вручную, удаляя ненужные строки или объединяя изменения.

2. Использование инструментов для разрешения конфликтов: Некоторые интегрированные среды разработки предоставляют инструменты для разрешения конфликтов, которые упрощают процесс.
3. Использование команд Git:
    - git status: Показывает, какие файлы имеют конфликты после попытки слияния.
    - git log --merge : создаст журнал со списком коммитов, конфликтующих между объединяющимися ветвями.
    - git diff: Показывает изменения, которые вызвали конфликты.
    - git mergetool: Запускает внешний инструмент для разрешения конфликтов.
4. Повторное слияние после разрешения конфликтов: После разрешения конфликтов файлов добавьте изменения в индекс с помощью git add, а затем завершите слияние с помощью git merge --continue.

МОНОРЕПОЗИТОРИЙ ИЛИ MONOREPO

Существует концепция укрупнения репозиториев в Git, которая называется монорепозиторием или monorepo. Это означает объединение нескольких проектов или компонентов в один общий репозиторий.



Вместо того чтобы каждый проект имел свой собственный репозиторий, все проекты объединяются в один монорепозиторий.

Особенно это полезно в нашем случае, когда у нас много разрозненных репозиториев, каждый из которых был посвящен одному занятию или одной домашней работе.

Преимущества монорепозитория :

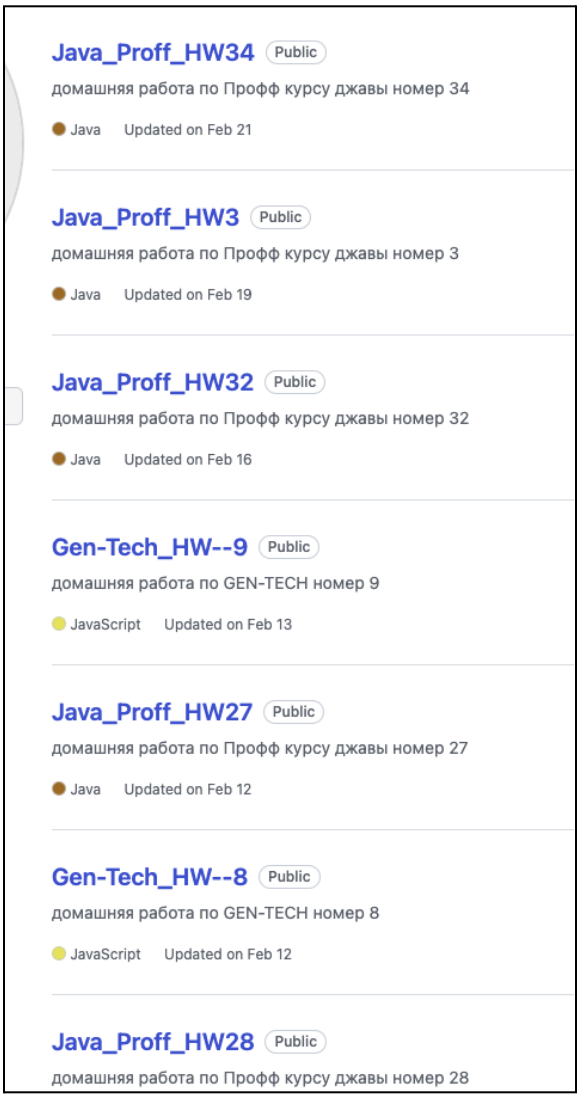
- Улучшение совместной работы
- Упрощение поиска
- Увеличение комфорта работы с проектами
- Удобная группировка репозиториев

Недостатки :

- Увеличение размера репозитория и сложности его управления.
- Увеличение времени загрузки репозитория при клонировании.
- Возможные конфликты при работе над различными частями кода в одном репозитории.

ЗАДАНИЕ ДЛЯ ЗАКРЕПЛЕНИЯ

1. Попробуем мигрировать наши разрозненные репозитории в monorepo, тем самым проделав укрупнение.
   Таким образом мы от:



Укрупним до:

Структуру для укрупнения Вы выбираете сами.

Полезные инструменты для такой миграции:

```
Unset
find . -type d -name ".git" -exec rm -rf {} +
```

Эта команда выполняет поиск (find) в текущей директории (.) и всех её поддиректориях (-type d) директорий с именем ".git" (-name ".git") и удаляет их рекурсивно (rm -rf).

Мы потеряем коммиты в индивидуальных репозиториях, так как удаляется .git вместе с содержимым.

2. Проверка нового укрупненного репозитория: работая с укрупненным репозиторием, проработайте добавление новой директории с уроком или домашним заданием. Обратите внимание на то, как работает git в директориях на разных уровнях. Создайте новый проект при помощи visual studio code и отправьте коммиты в новый укрупненный репозиторий.