Практическая работа №4
Изучение работы в системе контроля версий

1 Цель работы

1.1 Изучить основы работы с Git

1.2 Получить навыки работы с Git.

2 Литература

2.1 Гагарина, Л. Г. Технология разработки программного обеспечения : учебное пособие / Л.Г. Гагарина, Е.В. Кокорева, Б.Д. Сидорова-Виснадул ; под ред. Л.Г. Гагариной. — Москва : ФОРУМ : ИНФРА-М, 2022. — 400 с. — (Среднее профессиональное образование). – URL: https://znanium.com/catalog/product/1794453 . – Режим доступа: по подписке. – Текст : электронный. – гл.8.

3 Подготовка к работе

3.1 Повторить теоретический материал (см. п.2).

3.2 Изучить описание Практической работы.

4 Основное оборудование

4.1 Персональный компьютер.

5 Задание

5.1 Подготовка стандартных файлов репозитория

5.1.1 Создание readme.md

Создать начальный файл README.md.

Добавить базовое описание проектав README.md:

- название проекта,
- описание,
- инструкции по установке,
- использование,
- системные требования,
- авторы,
- лицензия.

5.1.2 Создание .gitignore

Написать файл .gitignore, в котором включить запрет на загрузку определенных файлов, расширений, директорий. Добавить исключения из запрета.

Дополнительно включить игнорирование:

- все файлы расширения log, за исключением находящихся в корневой папке apache.
- корневую папку bin.

- файлы, начинающиеся на tmp, находящиеся корневой папке dist или ее подпапках.

5.1.3 Загрузить в удаленный репозиторий файлы readme.md и .gitignore.

5.2 Создание форка

5.2.1 Создать форк открытого репозитория (например, https://github.com/gabrielecirulli/2048), нажав на кнопку Fork в правом верхнем углу страницы репозитория, чтобы создать свою копию.

5.2.2 Склонировать на свой компьютер репозиторий из форка.

5.3 Работа с коммитами

5.3.1 Создание новой ветки

Создать новую ветку для работы с изменениями:

git checkout -b my-feature-branch

Внести изменения в файлы репозитория из форка (например, добавить новый файл, изменить существующий файл).

Добавить изменения в индекс:

git add .

Зафиксировать изменения:

git commit -m "Добавил новую функциональность ..."

5.3.2 Откат коммитов

Сделать еще один коммит с изменениями:

git commit -m "Исправил ошибку 1"

Откатить последний коммит с помощью команды:

git reset --soft HEAD~1

Это вернет изменения в индекс, но оставит их в рабочем каталоге.

5.3.3 Откат коммитов

Сделать еще один коммит с изменениями:

git commit -m "Исправил ошибку 2"

Удалить последний коммит и изменения с помощью команды: git reset --hard HEAD~1

5.4 Слияние веток

5.4.1 Слияние с основной веткой Переключиться на основную ветку:

git checkout main

Обновитьосновнуюветку,чтобыполучитьпоследниеизмененияиз оригинального репозитория:

git remote add upstream <URL оригинального репозитория> git fetch upstream

git merge upstream/main

Затем вернуться к своей ветке:

git checkout my-feature-branch

Выполнить слияние с основной веткой: git merge main.

5.4.2 Разрешение конфликтов:

Внести конфликтующие изменения. Открыть файлы с конфликтами и вручную разрешить их, выбрав нужные изменения.

После разрешения конфликтов добавить измененные файлы в индекс: git add <имя файла>

Завершить слияние:

git commit -m "Разрешены конфликты"

5.4.3 Отправка изменений в форк

Отправить изменения в форк на GitHub:

git push origin my-feature-branch

5.5 Применение команд работы с измененими

5.5.1 Временное сохранение изменений

Применить git stash несколько раз, чтобы сохранить и восстановить разные изменения при переключении между разными ветками, не выполняя коммит.

5.5.2 Откат изменений

Создать несколько коммитов и применить git revert для отмены нескольких из них, чтобы увидеть, как это влияет на историю коммитов.

6 Порядок выполнения работы

6.1 Ознакомиться с заданием.

6.2 Выполнить задания средствами Git, где это возможно.

6.3 При выполнении работы соблюдать ясные наименованияк оммитов и дописывать описание к ним, если требуется.

6.4 Ответить на контрольные вопросы.

7 Содержание отчета

7.1 Титульный лист

7.2 Цель работы

7.3 Ответы на контрольные вопросы

7.4 Вывод

8 Контрольные вопросы

8.1 Как следует инициализировать репозиторий, чтобы избежать «лишних» файлов?

8.2 Зачем требуется давать ясные наименования коммитам?

8.3 Что такое HEAD указатель?

8.4 В чем отличие между git reset --soft и git reset --hard?

8.5 Зачем нужен .gitignore?

8.6 Зачем нужен readme.md?