**Зачет. Задание 2**

|  |  |
| --- | --- |
| Название программы | **«Инженер-тестировщик»** |
| Группа обучения | **«ИТ-6-Р/2025»** |
| Срок обучения | **«23.09.2025-27.10.2025»** |
| **«‎Елисеев Александр Викторович»** | |
| Номер/Название Кейса | **«№2 Инструменты и технологии тестирования.»** |

Москва 2025\_ г.

**ID:** 2.1

**Заголовок:** Создание базового репозитория в Git и организация работы с ветками.

**Предусловие:** Пользователь на ОС Windows находится в браузере Google Chrom (Версия: 139.0.7258.155), открыта стартовая страница сайта https://git-scm.com.

|  |  |  |
| --- | --- | --- |
| **№** | **Шаг** | **Ожидаемый результат** |
| **1.Установка и настройка Git** | | |
| **1.1** | **Установка Git на локальную машину** | Открывается веб-страница <https://git-scm.com>., в которой кликаем по надписи Downloads.  В открывшемся окне кликаем по Windows, в следующем окне выбираем необходимую нам версию программы. Начинается скачивание дистрибутива, который открываем и устанавливаем локальную машину. |
|  | | |

|  |  |  |
| --- | --- | --- |
| **1.2** | **Базовая конфигурация(имя пользователя и email)** | Открываем на ПК командную строку и в открывшемся окне вводим две команды с указанием Имени и Электронной почты:  -git config --global user.name "Aleksandr Eliseev"  -git config --global user.email x103ey@gmail.com |
|  | | |
|  | | Проверяем корректность введенных данных и вводим команду git config –list |
|  | | |
| **1.3** | **Проверка установки командой**  ***git --version*** | Открываем в меню пуск GIT BASH и проверяем версию установленной программы. |
|  | | |

|  |  |  |
| --- | --- | --- |
| **2.Создание локального репозитория.** | | |
| **2.1** | **Иницилизация нового репозитория командой *git init*** | C помощью файл-менеджера выбираем папку, в которой будет создан наш репозиторий. Открываем ее в GIT BASH, кликнув по ней правой кнопкой мыши, и выбрав соответствующий пункт меню. |
|  | | |
|  | | В открывшемся окне GIT BASH вводим команду  ***git init*** и создаем репозиторий. |
|  | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **2.2** | **Создание базовой структуры проекта** | | Создаем структуру проекта(папки), вводим команду:  ***mkdir -p {src,test,docs}*** |
|  | | | |
| **2.2** | **Добавление файла README.md с описанием проекта** | Добавляем файл README.md с описанием проекта  ***echo "#My Project">README.md*** | |
|  | | | |
| **3.Работа с коммитами** | | | |
| **3.1** | **Добавление файлов в**  **staging area** | Добавляем файл в staging area:  ***git add <имя файла>*** | |
|  | | | |
| **3.2/**  **3.3** | **Создание первого коммита и просмотр истории коммитов.** | Создаем первый коммит и смотрим историю коммитов:  ***git commit -m "Initial commit"***  ***git log*** | |
|  | | | |

|  |  |  |
| --- | --- | --- |
| **4.Работа с ветками** | | |
| **4.1** | **Создание новой ветки**  **feature/login** | Для работы над кодом командой создадим новую ветку «feature/login», вводим команду  ***git branch feature/login*** |
|  | | |
| **4.2** | **Переключение между ветками** | Переходим в ветку feature/login  ***git checkout feature/login*** |
|  | | |
| **4.3** | **Внесение изменений в новой**  **ветке** | Ввод изменений в новой ветке. Открываем наш файл:  ***start <имя файла>***  Вносим в него изменения, сохраняем и закрываем. |
|  | | |

|  |  |  |  |
| --- | --- | --- | --- |
|  | | Проверяем отслеживаемость изменений файла  ***git status***,  Далее снова добавляем файл в staging area:  ***git add <имя файла>*** | |
|  | | | |
| **4.4** | **Создание коммитов в**  **feature-ветке** | | Добавляем коммит:  ***git commit –m “Добавили 1212”*** |
|  | | | |
| **5.Слияние веток** | | | |
| **5.1** | **Возврат в основную ветку** | | Возвращаемся в основную ветку  ***git checkout <имя ветки>*** |
|  | | | |
| **5.2** | **Слияние изменений** | | Производим слияние изменений  ***git merge feature/login*** |
|  | | | |
| **5.3** | **Разрешение возможных конфликтов** | | Во время слияния конфликтов не возникло |

|  |  |  |
| --- | --- | --- |
| **6.Работа с удаленным репозиторием** | | |
| **6.1** | **Создание репозитория на GitHub** | Отрываем в браузере страницу <https://github.com/> и проходим регистрацию на ней. |
|  | | |
|  | | В открывшемся окне кликаем по New. |
|  | | |

|  |  |  |
| --- | --- | --- |
|  | | В следующем окне вводим название нашего репозитория, вибираем его статус и кликаем по Create repository.  Репозиторий создан. |
|  | | |
| 6.2 | Добавление remote-источника | После выполнения действий указанных в п. 6.1 открывается окно, где необходимо скопировать SSH адрес нашего репозитория. |
|  | | |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | | | Далее открываем на локальной машине Git Bash и добавляем remote-источник  ***git remote add origin < скопированный адрес>*** | |
|  | | | | |
| 6.3 | Отправка изменений | | | Перед отправкой на удаленный сервер рекомендуется добавить в staging area все изменения, сделать коммит и ввести команду ***git push*** |
|  | | | | |
|  | Далее переходим в браузер на страницу с созданным репозиорием и обновляем ее,после чего на ней в нашем аккаунте GitHub появляется структуру отправленного проекта с коммитами, созданная на локальной машине. | | | |
|  | | | | |
| **7.ДОПОЛНИТЕЛЬНЫЕ ЗАДАНИЯ** | | | | |
| В последних версия PyCharm уже интегрирован Git, что урощает контроль версий созданного кода. Поэтому созданные проекты можно обрабатывать в Git и отправлять на удаленный репозиторий GitHub непосредственно из интерфейса PyCharm, при помощи встроенной командной строки. | | | | |
| 7.1 | **Открытие проекта в PyCharm** | С помощью файл-менеджера находим папку с нашим проектом и открываем ее в PyCharm | | |
|  | | | | |
| 7.2 | **Использование .gitignore.** | В корневой папке создаем файл с именем “.gitignore” и добавляем в него файлы и папки, которые GIT будет игнорировать, при контроле изменений. | | |
|  | | | | |
| 7.3 | **Отмена изменений** | Использование команды git reset позволяет убрать из staging area все внесенные файлы с изменениями. | | |
|  | | | | |

|  |  |  |
| --- | --- | --- |
| 7.4 | **Создание и применение тегов** | Теги устанавливаются после создания коммитов и обозначаются в виде версий: - ***git tag -a v1.2.0 -m "Release 1.2.0 - best"*** (создание)  ***- git push origin v1.2.0*** (отправка на удаленный сервер)  - ***git tag – l*** (проверка создания) |
|  | | |