Отчёт по лабораторной работе 3

(лаб 2 переделанная в маркдаун)

Свояк Валерия Дмитриевна

Содержание

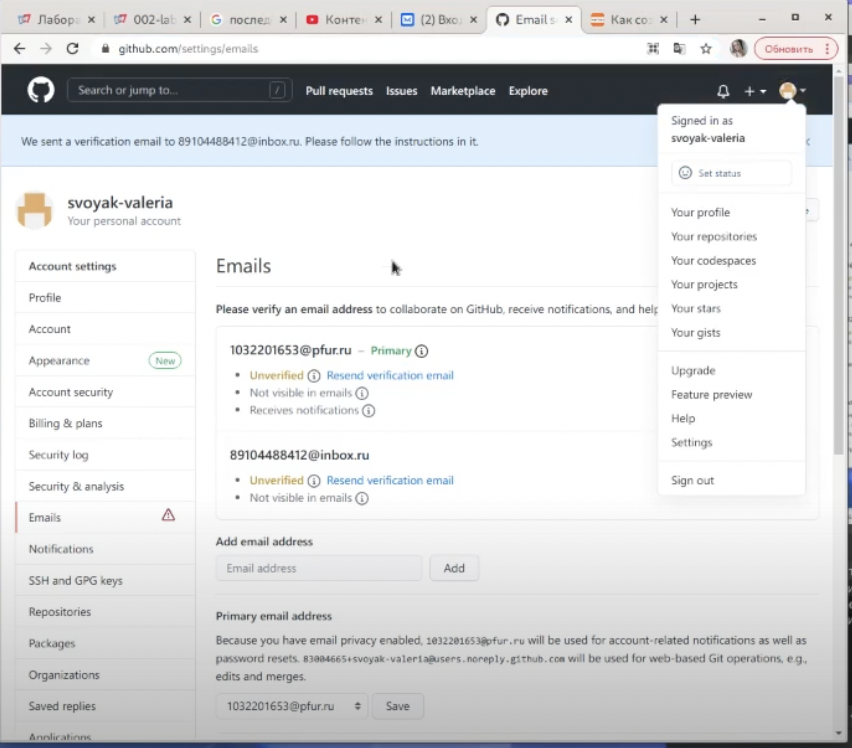
# Цель работы

Изучить идеологию и применение средств контроля версий.

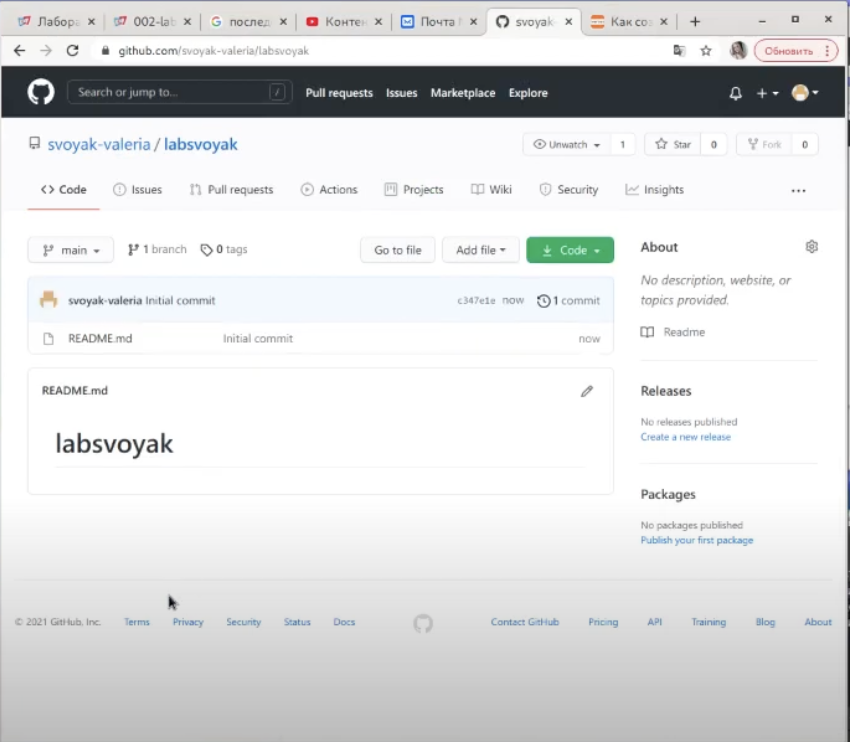
# Задание

Лабораторная работая вляется небольшой научно-исследовательской работой,которую и оформлять следует по всем утверждённым требованиям.При подготовке отчета по лабораторной работе вы освоите ряд важных элементов,которые вдальнейшем пригодятся вам при написании курсовой и дипломной работы.

# Выполнение лабораторной работы

**Настройка git** 1. Создайте учётную запись на https://github.com. 

2. Настройте систему контроля версий git, как это описано выше c использованием сервера репозиториев https://github.com/.   
![Название рисунка](img/img02.png){ #fig:001 width=70% }

Создаем публичный репозиторий на гетхаб 1  Появился

3. Устанавливаем ключ для связи  
• Заходим в настройки гетхаб и в установку SSH ключей  
  
• Прописываем команду cd.ssh для перехода в каталог и генерируем публичный и приватный ключи ssh-keygen -C “Свояк Валерия <89104488412@inbox.ru>”  
  
• Убедились, что ключ появился:  
  
  
4. Чтобы вставить ключ в гетхаб, нам нужно получить на него ссылку. Для этого воспользуемся текстовым редактором. На фото можно видеть, как мы выбрали публичный ключ и открыли его с помощью текстового редактора  
  
5. Вставляем ссылку на гетхаб   
  
  
  
  
6. Ключ установился  
  
  
7. Создаем каталог лаборатория и клонируем в него наш ключ  
  
  
8. Посмотрим, какие файлы есть в каталоге.   
  
9. Чтобы создать первый коммит, изменим содержание файла README.md. Для этого:  
• В текстовый документ добавляем фразу «Что-то новое» с помощью команды echo  
•   
• Текст добавился   
•   
• Можем просмотреть это через команду cat   
•   
10. Изменения произошли, можно создавать коммит. Для этого:  
• Используем команду git commit -m “first commit”  
• Используя команду git config –global user.name вводим почту и имя  
• Устанавливаем коммит командой git commit -m “first commit”  
•   
• Добавляем файл README.cd и с помощью команды git push передаем данные в гетхаб  
•   
• Коммит появился  
•   
11. Теперь делаем первичную конфигурацию – создаем лицензию:  
• Вставляем файл лицензии - wget https://creativecommons.org/licenses/by/4.0/legalcode.txt   
  
• Добавляем шаблон игнорируемых файлов curl -L -s https://www.gitignore.io/api/list  
  
• Скачиваем шаблон и добавим все файлы (git add .)  
  
• Устанавливаем второй коммит  
  
• И отправляем его на гитхаб  
•   
  
  
  
• Проверяем – они появились  
  
12. Конфигурация git flow

–  Инициализируем git-flow git flow init Префикс для ярлыков установим в v.

–  Проверьте,что Вы на ветке develop: git branch

–  Создадим релиз с версией 1.0.0 - git flow release start 1.0.0

–  Запишем версию: echo “1.0.0” >> VERSION

–  Добавим в индекс: git add . git commit -am ‘chore(main): add version’

–  Зальём релизную ветку в основную ветку git flow release finish 1.0.0

–  Отправим данные на github git push –all git push –tags

–  Создадим релиз на github (появились ветки и в них файлы)

Вывод: Изучила идеологию и применение средств контроля версий. Ответы на контрольные вопросы: 1. Система управления версиями - ПО для облегчения работы с изменяющейся информацией. Система управления версиями позволяет хранить несколько версий одного и того же документа, при необходимости возвращаться к более ранним версиям, определять, кто и когда сделал то или иное изменение, и многое другое. 2. Основным понятием VCS является репозиторий (repository) – специальное хранилище файлов и папок проекта, изменения в которых отслеживаются. В распоряжении разработчика имеется так называемая “рабочая копия” (working copy) проекта, с которой он непосредственно работает. Рабочую копию необходимо периодически синхронизировать с репозиторием, эта операция предполагает отправку в него изменений, которые пользователь внес в свою рабочую копию (такая операция называется commit) и актуализацию рабочей копии, в процессе которой к пользователю загружается последняя версия из репозитория (этот процесс носит название update). 3. • Централизованная, предусматривающая концентрацию функций в рамках единой службы материально-технического обеспечения.  • Децентрализованная, предусматривающая осуществление МТО бизнес единицами самостоятельно.  • Смешанная форма организации материально-технического обеспечения, состоящая в комбинирования указанных выше способов. 6. Система контроля версий ГИТ представляет собой набор программ командной строки. Доступ к ним можно получить из терминала посредством ввода команды гит с различными опциями. 7. git add Команда git add добавляет содержимое рабочей директории в индекс (staging area) для последующего коммита. По умолчанию git commit использует лишь этот индекс, так что вы можете использовать git add для сборки слепка вашего следующего коммита. git status Команда git status показывает состояния файлов в рабочей директории и индексе: какие файлы изменены, но не добавлены в индекс; какие ожидают коммита в индексе. Вдобавок к этому выводятся подсказки о том, как изменить состояние файлов. git commit Команда git commit берёт все данные, добавленные в индекс с помощью git add, и сохраняет их слепок во внутренней базе данных, а затем сдвигает указатель текущей ветки на этот слепок. git branch Команда git branch — это своего рода “менеджер веток”. Она умеет перечислять ваши ветки, создавать новые, удалять и переименовывать их.

1. Ветка в Git’е — это просто «скользящий» указатель на один из коммитов. Когда вы создаёте новые коммиты, указатель ветки автоматически сдвигается вперёд, к вновь созданному коммиту.
2. Игнорируемые файлы обычно представляют собой файлы, специфичные для платформы, или автоматически созданные из сборочных систем.  Файл .gitignore указывает, какие неотслеживаемые файлы должен игнорировать Git .

# Выводы

Здесь кратко описываются итоги проделанной работы.