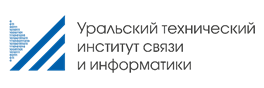
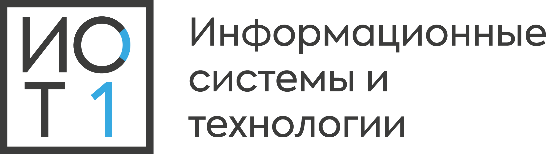
Министерство цифрового развития, связи и массовых коммуникаций РФ Уральский технический институт связи и информатики (филиал) ФГБОУ ВО "Сибирский государственный университет телекоммуникаций и информатики" в г. Екатеринбурге (УрТИСИ СибГУТИ)

КАФЕДРА

Информационных систем и технологий

**ОТЧЕТ**

По дисциплине «Сетевое программирование»

Практическая работа №1

 «Реализация механики выстрела с обратной связью»

|  |  |
| --- | --- |
| Выполнил: | студент гр. ПЕ-21б  Попов В. А. |
| Проверил: | преподаватель   Тупицын К.М. |

Екатеринбург, 2024

**Практическая работа 1**

Работа с системой управления версиями

1. **Цель работы:**
   1. Научиться работать с системой управления версиями;
   2. Закрепить знания по теме «Системы управления версиями».
2. **Перечень оборудования:**
   1. Персональный компьютер;
   2. Система управления версиями Git;
   3. Visual Studio Code.
3. **Ход работы:**
   1. Перед началом работы с Git необходимо создать папку с фамилией и номером группы, в которой будут хранится наши элементы. После этого, необходимо создать следующую структуру, как показано на рисунке 1.

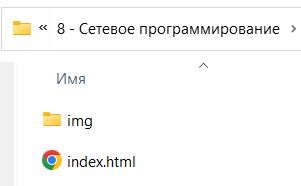


Рисунок 1 – Структура папки

Как видно из структуры, в корневой директории лежат:

1. Основная страница блога index.html
2. Папка img, в которой хранятся изображения ( файлы \*.jpg) сайта.

Тематика блога выбирается самостоятельно, картинки – тоже, но в рамках общественной нормы.

В качестве примера предполагается, что блог посвящен диким животным. Тогда структура файла index.html будет выглядеть как показано в листинге 1.

Листинг 1 – Структура index.html

<!DOCTYPE html>

<html>

<head>

<title>Wild Animals Blog</title>

<meta charset="utf-8">

</head>

<body align=center>

<img width=100px src="pictures/paw\_print.jpg">

|  |  |  |
| --- | --- | --- |
| <h1>Wild Animals</h1>  <small>- Blog about nature -</small> | | |
| <br>  <br>  <br>  <br> |  | |
| <p>Let's talk about wild animals around the world:</p>  <h2>Giraffe</h2>  <img src="pictures/giraffe.jpg">  <p><b>Area:</b> Africa</p>  <p><b>Weight:</b> 900-1200kg</p>  <p><b>Height:</b> 6m</p>  <br> | | |
| <h2>Elehant</h2>  <img width=350px  <p><b>Area:</b> | | src="pictures/elephant.jpg"> Africa, Asia</p> |
| <p><b>Weight:</b> 4000-7000kg</p>  <p><b>Height:</b> 3m</p>  <br>  </body>  </html> | | |

Если отрыть данный файл в браузере, то выглядит основная (и пока единственная) страница блога показана на рисунке 2:

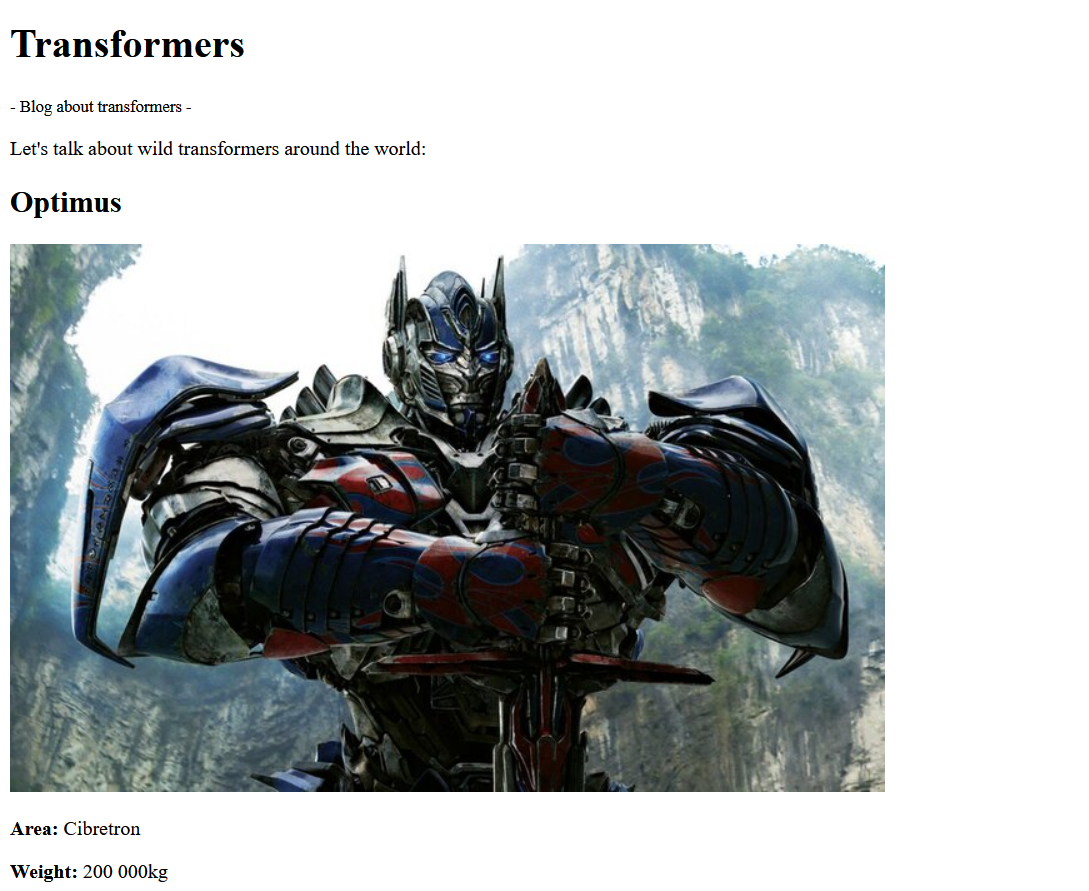


Рисунок 2 – Страница блога в браузере

Далее требуется создать репозиторий для нашего блога. Для этого необходимо перейти в директорию будущего репозитория (или открыть при помощи терминала вашу папку)

$ cd **название вашей папки**

Далее необходимо прописать команду git. Если у Вас открывается сообщение, как показано на рисунке 3, то значит git установлен. Если подобное сообщение не открывается необходимо установить его в зависимости от Вашей операционной системы с сайта [https://git-](https://git-scm.com/book/ru/v2/Введение-Установка-Git) scm.com/book/ru/v2/Введение-Установка-Git

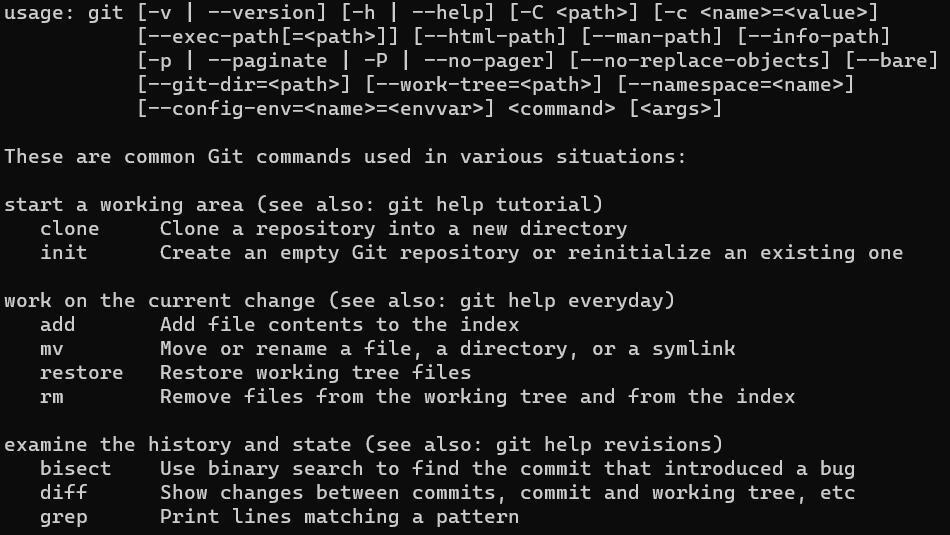


Рисунок 3 – Сообщение, если git установлен

Для создания репозитория необходимо прописать команду git init, после чего появится сообщение:

git init

Initialized empty Git repository in **/Путь к вашей папке**/.git/

После такого сообщения, можно проверить структуру git файла, как показано в листинге 2:

Листинг 2 – Проверка структуры файла .git

tree .git

.git

├── HEAD

├── config

├── description

|  |  |
| --- | --- |
| ├── hooks  ├── info  │ └── exclude  ├── objects  │ ├── info  │ └── pack | |
| └── refs  ├──  └── | heads tags |

Далее необходимо проверить текущее состояние рабочей директории, как показано в листинге 3:

Листинг 3 – Проверка текущего состояния директории

|  |  |
| --- | --- |
| git status  On branch master  No commits yet | |
| Untracked files:  (use "git add | <file>..." to include in what will be committed) |
| img/  index.html  nothing added to commit but untracked files present (use "git add" to track) | |

Как видно из листинга, есть не отслеживаемые файлы. Как сделать отслеживаемыми – будет чуть-чуть попозже.

Далее необходимо выполнить следующее задание – настроить пользователя Git на уровне локального репозитория:

1. Изучите содержимое файла конфигурации Git для текущего репозитория (.git/config)
2. Настройте имя и email пользователя для текущего репозитория
3. Убедитесь, что файл .git/config изменился соответствующим образом Для вывода содержимого файла конфигурации Git для текущего

репозитория необходимо воспользоваться командой, как показано в листинге 4.

Листинг 4 – Вывод содержимого файла конфигурации

cat .git/config // команда

[core] // отсюда начинается результат

repositoryformatversion = 0

filemode = false

bare = false

logallrefupdates = true

symlinks = false

ignorecase = true

Далее необходимо задать имя и email пользователя для текущего репозитория при помощи команды, показанной в листинге 5. Значение name и [your@email.ru](mailto:your@email.ru) указываются студентами самостоятельно

Листинг 5 – Добавление имени и email пользователя

$ git config user.name **name**

[$ git config user.email **your@email.ru**](mailto:%24gitconfiguser.emailyour@email.ru)

После этого, необходимо проверить содержимое файла конфигурации при помощи команды, которая была использована в листинге 4. Результат выполнения для тестового задания представлен в листинге 6.

Листинг 6 – Вывод содержимого файла конфигурации

|  |  |
| --- | --- |
| [core] |  |
|  | repositoryformatversion = 0  filemode = false  bare = false  logallrefupdates = true  symlinks = false  ignorecase = true |
| [user] | name = bdi  [email = bdi@urtisi.ru](mailto:email%3Dbdi@urtisi.ru) |

Следующим заданием будет необходимо создать первый коммит. Для этого необходимо посмотреть состояние рабочей директории, как объяснялось ранее.

После того, как было проверено текущее состояние директории, необходимо добавить все файлы в индекс при помощи команды **git add \*.**

После этого, снова необходимо проверить статус репозитория. Если все сделано верно, то файлы попали в индекс и теперь готовы к коммиту, как показано в листинге 7.

Листинг 7 – Статус репозитория после добавления в индекс

|  |  |  |
| --- | --- | --- |
| On branch master  No commits yet | | |
| Changes to be committed:  (use "git rm --cached | <file>..." to unstage) | |
| new file: img/elephant.jpg  new file: img/giraphe.jpeg | |  |
| new file: img/paw\_print.jpg  new file: index.html | | |

Далее необходимо выполнить коммит при помощи команды **git commit - m "Project initial"**. После этого должен быть ответ, как показан на листинге 8.

Листинг 8 – Ответ после выполнения коммита

git commit -m "G-02: Initial"

[master (root-commit) 6bf1f85] G-02: Initial

4 files changed, 36 insertions(+)

create mode 100644 img/elephant.jpg

create mode 100644 img/giraphe.jpeg

create mode 100644 img/paw\_print.jpg

create mode 100644 index.html

Далее можно проверить последнее изменение при помощи команды **git log,** как показано в листинге 9.

Листинг 9 – Результат выполнения команды git log

|  |  |
| --- | --- |
| git log commit Author: Date: | 6bf1f8525f7d730a00d0c93635867e2d40d74042 (HEAD -> master)  bdi <[bdi@urtisi.ru](mailto:bdi@urtisi.ru)>  Sun Oct 15 22:31:54 2023 +0500 |
| G-02: Initial | |

В конце необходимо проверить статус репозитория, результат выполнения представлен в листинге 10.

Листинг 10 – Результат выполнения команды git status

git status

On branch master

nothing to commit, working tree clean

Индивидуальное задание по пункту 3.1. В существующий файл index.html необходимо добавить еще одну картинку и дополнительный текст и сделать второй коммит самостоятельно. Результат выполнения может быть представлен в форме листинга/скриншота.

* 1. Индивидуальное задание студента.

1. Зарегистрируйтесь на GitHub и подтвердите свою почту.
2. Настройте SSH или HTTPS подключение (на ваш выбор). Для того, чтобы настроить SSH, можно прочитать здесь: [https://timeweb.cloud/tutorials/windows/kak-sgenerirovat-ssh-klyuch-dlya-](https://timeweb.cloud/tutorials/windows/kak-sgenerirovat-ssh-klyuch-dlya-windows) windows
3. Отправьте ваш локальный репозиторий (созданный в задании 3.1) на аккаунт GitHub (при помощи командной строки или вспомогательного ПО, не имеет значения).
4. Внесите изменения в ваш репозиторий на локальном компьютере, сделайте коммит и отправьте его на GitHub.
5. Создайте ветку с указанием Вашей фамилии и инициалов, например, BurumbaevDI
6. Далее необходимо переключится на ветку с вашей фамилией, изменить ваш файл index.html и отправить его на GitHub (именно с второй ветки).
7. Необходимо выполнить Pull Request, указав комментарий.
8. После этого необходимо произвести слияние версий от главной ветки (master/main) и второстепенной (в случае примера, это BurumbaevDI). Данный раздел необходимо изучить самостоятельно.

Для того, чтобы выполнить индивидуальное задание №2 необходимо ознакомиться с приложением А к первой практической работе.

Приложение А

Создание репозитория на Github

До текущего момента мы работали с локальным репозиторием, который сохранялся в папке на компьютере. Если мы хотим иметь возможность сохранения проекта в интернете, создадим репозиторий на Github. Для начала нужно зарегистрироваться на сайте github.com под именем myuser (в вашем случае это может быть любое другое имя).

После регистрации нажимаем кнопочку "+" и вводим название репозитория. Выбираем тип Public (репозиторий всегда Public для бесплатной версии) и нажимаем Create.

В результате мы создали репозиторий на сайте Github. На экране мы увидим инструкцию, как соединить наш локальный репозиторий со вновь созданным. Часть команд нам уже знакома.

Добавляем удаленный репозиторий (по протоколу SSH или HTTPS) под именем origin (вместо origin можно использовать любое другое имя).

git remote add origin git@github.com:myuser/project.git

Можем просмотреть результат добавления с помощью команды:

git remote -v

Если все было правильно сделано, то увидим:

origin git@github.com:myuser/project.git (fetch) origin git@github.com:myuser/project.git (push)

Для того, чтобы отменить регистрацию удаленного репозитария введите:

git remote rm origin

Это может понадобиться, если вы захотите поменять SSH доступ на HTTPS. После этого можно добавить его опять, например, под именем github и протоколом HTTPS.

git remote add github https://github.com/myuser/project.git

Следующей командой вы занесете все изменения, которые были сделаны в локальном репозитории на Github.

git push --set-upstream origin master

Ключ -u используется для того, чтобы установить связь между удаленным репозиторием github и вашей веткой master. Все дальнейшие

изменения вы можете переносить на удаленный репозиторий упрощенной командой.

Другим заданием является ветвление. Ветвление стало неотъемлемой частью командной разработки, потому что оно дает возможность работать над разными версиями исходного кода. Основной идеей ветвления является отклонение от основного кода и продолжение работы независимо от него. Также это удобно в тестировании отдельного функционала, потому что позволяет работать над новой частью кода, не беспокоясь о поломке чего-то в рабочей версии.

Под веткой принято понимать независимую последовательность коммитов в хронологическом порядке. Однако конкретно в Git реализация ветки выполнена как указатель на последний коммит в рассматриваемой ветке. После создания ветки уже новый указатель ссылается на текущий коммит.

Имя основной ветки Git-проекта по умолчанию — master (однако зачастую бывает main, например, в GitHub), она появляется сразу при инициализации репозитория. Эта ветка ничем не отличается от остальных и также ее можно переименовать, но по договоренности master принято считать главной веткой в проекте.

Команда git branch — главный инструмент для работы с ветвлением. С ее помощью можно добавлять новые ветки, перечислять и переименовывать существующие и удалять их.

Чтобы в Git добавить ветку мы используем:

$ git branch <name of new branch>

После данной операции ветка уже была создана, но вы по-прежнему находитесь в прежней ветке. Если вы планируете переместиться на другую ветку, в том числе только что созданную, необходимо написать checkout:

$ git checkout <name of branch>

$ git branch

При выполнении этой строки мы получим список существующих веток, где символом \* будет отмечена ветка, где вы сейчас находитесь. Это может выглядеть так:

first\_branch

\* master second\_branch

С помощью параметра -v можно получить последний сохраненный коммит в каждой ветке.

$ git branch -v

first\_branch 8fa301b Fix math

\* master 225cc2d Merge branch 'first\_branch' second\_branch c56ee12 Refactor code style

После того как вы выполнили создание ветки и изменили документ, Вам необходимо отправить Pull Request.

Pull Request — запрос на включение. На включение написанного вами кода в чужой репозиторий.

Если нужно отправить в свой удалённый репозиторий вновь созданную ветку (не сливать её с master), делаем следующее:

$ git push master new\_branch

Добрались таки до ответа на поставленный вопрос: что такое pull request, зачем оно нужно и как его достичь. Как предложить владельцу репозитория свои изменения?

Для этого зайдите в свой аккаунт, выбирайте репозиторий владельца и ищите небольшую зелёную кнопку.

Перед тем как сделать запрос вы имеете возможность добавить комментарий, просмотреть то, какие файлы будут изменены, какие коммиты добавлены. В верхнем углу окна добавления запроса обратите внимание откуда куда и что вы сливаете. Если необходимо слить основные ветки выбор падёт на репозиторий username:master, если отдельную ветку (вспоминаем branch) — так и указывайте её.

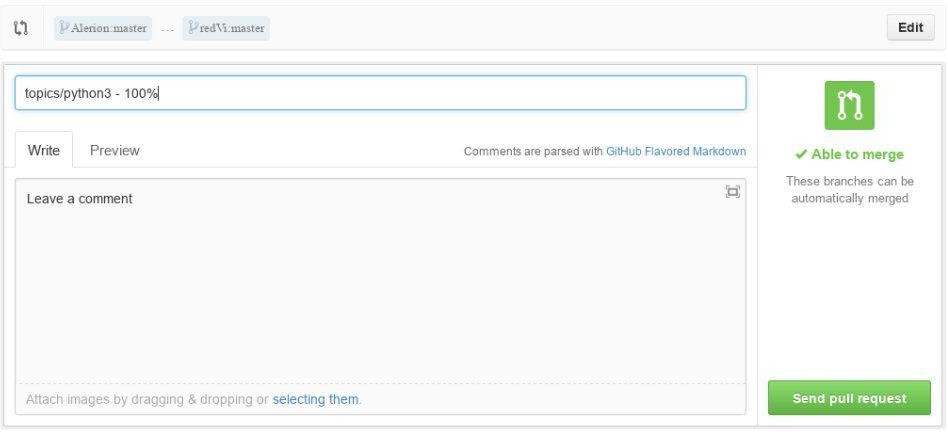


Рисунок 4 – Отправка Pull Request