

# **Отчёт по индивидуальному проекту. Этап 1.**

**Архитектура компьютера и операционные системы**

Рогожина Надежда Александровна, НКАбд-02-22

# Содержание

<b>1</b>	<b>Цель работы</b>	<b>6</b>
<b>2</b>	<b>Задание</b>	<b>7</b>
<b>3</b>	<b>Теоретическое введение</b>	<b>8</b>
3.1	Примеры использования git: . . . . .	8
3.2	Основные команды git: . . . . .	8
<b>4</b>	<b>Выполнение лабораторной работы</b>	<b>10</b>
<b>5</b>	<b>Выводы</b>	<b>26</b>
	<b>Список литературы</b>	<b>27</b>

## Список иллюстраций

4.1	Загрузка нужной нам версии . . . . .	10
4.2	Скачивание . . . . .	10
4.3	Извлечение . . . . .	11
4.4	Извлечение - выбор папки . . . . .	11
4.5	Проверка извлечения . . . . .	12
4.6	Вырезание файла . . . . .	12
4.7	Перенос файла . . . . .	13
4.8	Создание репозитория . . . . .	13
4.9	Создание репозитория . . . . .	14
4.10	Проверка . . . . .	14
4.11	Клонирование репозитория . . . . .	15
4.12	Проверка . . . . .	15
4.13	Hugo . . . . .	15
4.14	Доустановка модуля “GO” . . . . .	16
4.15	HUGO . . . . .	16
4.16	Проверка . . . . .	16
4.17	Удаляем public . . . . .	17
4.18	Запуск hugo server . . . . .	17
4.19	Запуск hugo server . . . . .	17
4.20	Шаблон сайта . . . . .	18
4.21	Закрытие hugo server . . . . .	18
4.22	Github . . . . .	18
4.23	Создаем новый репозиторий . . . . .	19
4.24	Клонирование репозитория . . . . .	19
4.25	Проверка . . . . .	19
4.26	Переключаемся на ветку “main” . . . . .	20
4.27	Создание пустого файла и отправка изменений . . . . .	20
4.28	Проверка . . . . .	20
4.29	Создаем новую ветку . . . . .	21
4.30	Исправление ошибки .gitignore . . . . .	21
4.31	Проверка . . . . .	22
4.32	Повтор введенной команды . . . . .	22
4.33	HUGO . . . . .	22
4.34	Проверка . . . . .	23
4.35	Загружаем обновления . . . . .	23
4.36	Загружаем обновления . . . . .	23
4.37	Проверка github . . . . .	24

4.38 Шаблон сайта готовый . . . . .	25
-------------------------------------	----

# Список таблиц

3.1	Описание некоторых команд системы контроля версий Git . . . .	8
-----	---	---

# 1 Цель работы

- Создать свой сайт (разместить на Github pages заготовки для персонального сайта)

## 2 Задание

- Установить необходимое программное обеспечение.
- Скачать шаблон темы сайта.
- Разместить его на хостинге git.
- Установить параметр для URLs сайта.
- Разместить заготовку сайта на Github pages.

## 3 Теоретическое введение

### 3.1 Примеры использования git:

- Система контроля версий Git представляет собой набор программ командной строки. Доступ к ним можно получить из терминала посредством ввода команды git с различными опциями.
- Благодаря тому, что Git является распределённой системой контроля версий, резервную копию локального хранилища можно сделать простым копированием или архивацией.

### 3.2 Основные команды git:

Например, в табл. 3.1 приведено краткое описание основных команд Git.

Таблица 3.1: Описание некоторых команд системы контроля версий Git

Команда	Описание команды
git init	Создание основного дерева репозитория
git pull	Получение обновлений(изменений текущего дерева из центрального репозитория
git push	Отправка всех произведённых изменений локального дерева в центральный репозиторий
git status	Просмотр списка изменённых файлов в текущей директории



Команда	Описание команды
git diff	Просмотр текущих изменений
git add .	Добавление все изменённые и/или созданные файлы и/или каталоги
git rm имя-на_файлов	Удаление файлов и/или каталогов из индекса репозитория
git commit -am 'Описание коммита'	Сохранение всех добавленных изменений и всех изменённых файлов
git commit	Сохранение добавленный изменений с внесением комментария через встроенный редактор
git checkout -b имя_ветки	Создание новой ветки, базирующейся на текущей
git branch -d имя_ветки	Удаление локальной уже слитой с основным деревом ветки
git branch -D имя_ветки	Принудительное удаление локальной ветки

Полный список команд можно посмотреть на официальном сайте: [Github.com](https://github.com)

## 4 Выполнение лабораторной работы

Загружаем последнюю версию hugo (рис. 4.1):



Рис. 4.1: Загрузка нужной нам версии

Файл скачивается в папку “Загрузки” (рис. 4.2):

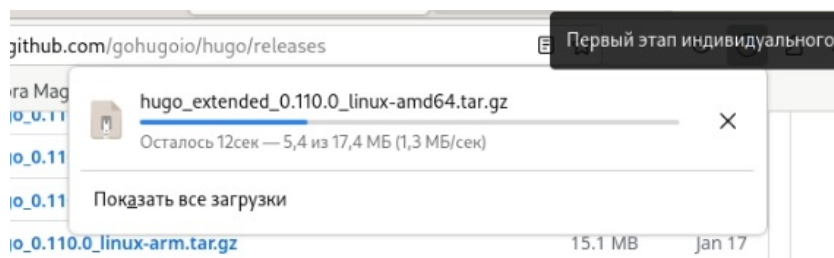


Рис. 4.2: Скачивание

По завершении скачивания извлекаем архив в ту же папку, в которой мы находимся:

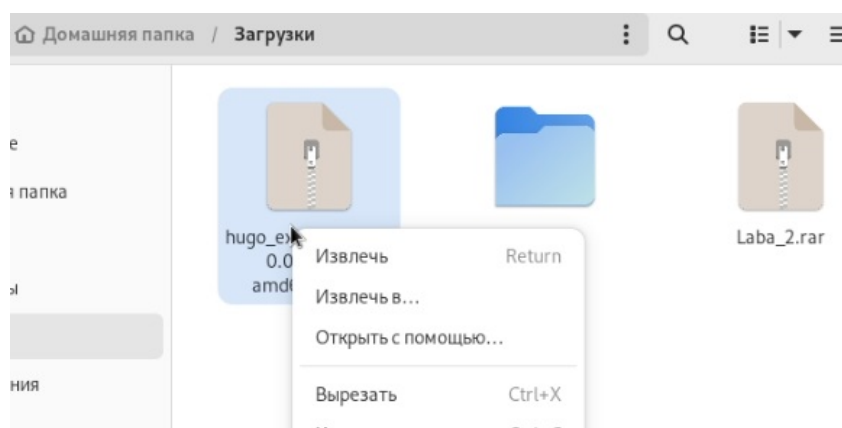


Рис. 4.3: Извлечение

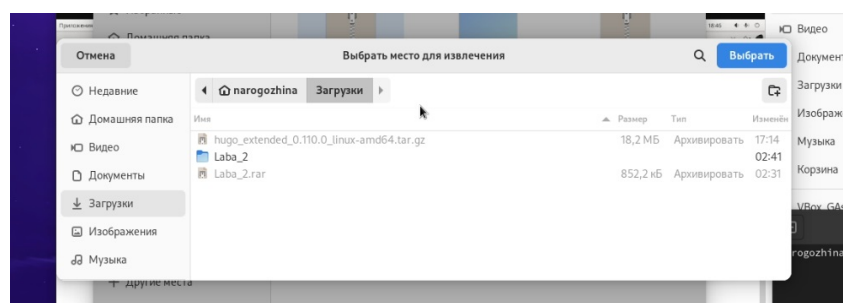


Рис. 4.4: Извлечение - выбор папки

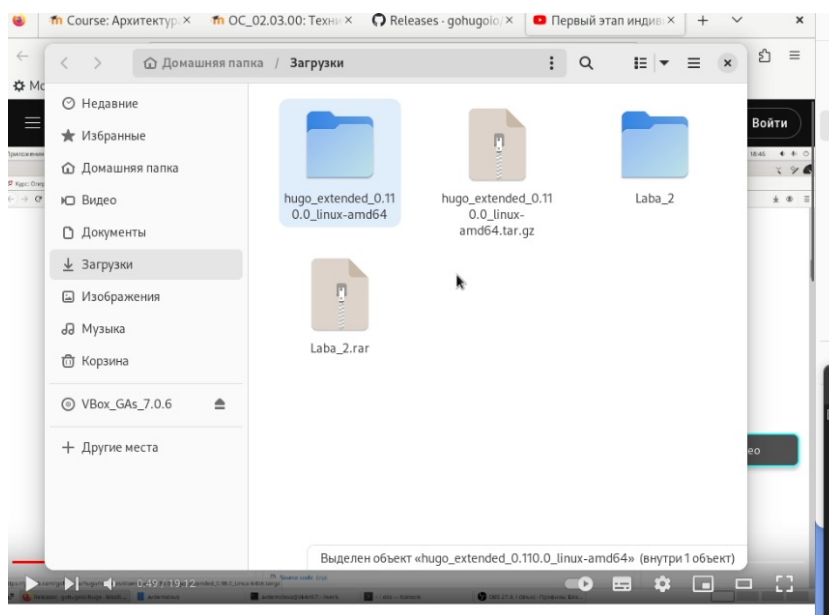


Рис. 4.5: Проверка извлечения

После извлечения файла, нам его необходимо вырезать и вставить в папку `/usr/local/bin`:

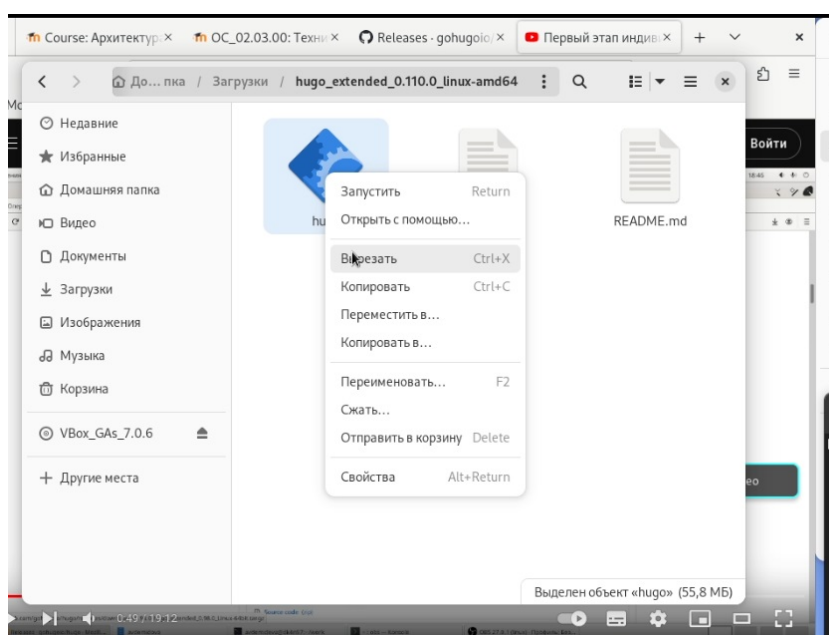


Рис. 4.6: Вырезание файла

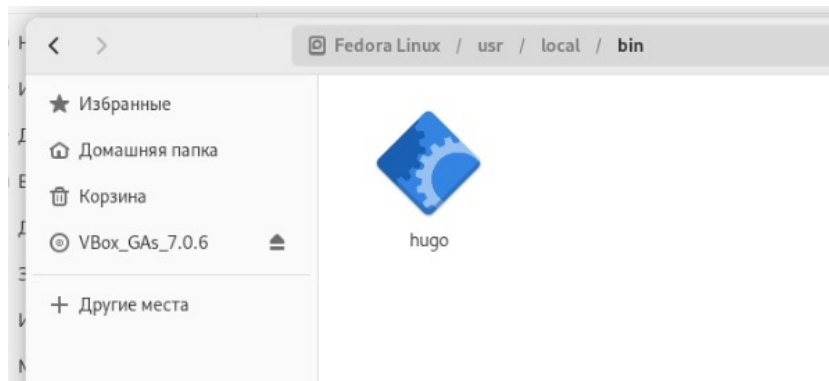


Рис. 4.7: Перенос файла

Далее открываем наш **github** и создаем репозиторий на основе данного нам: Репозиторий

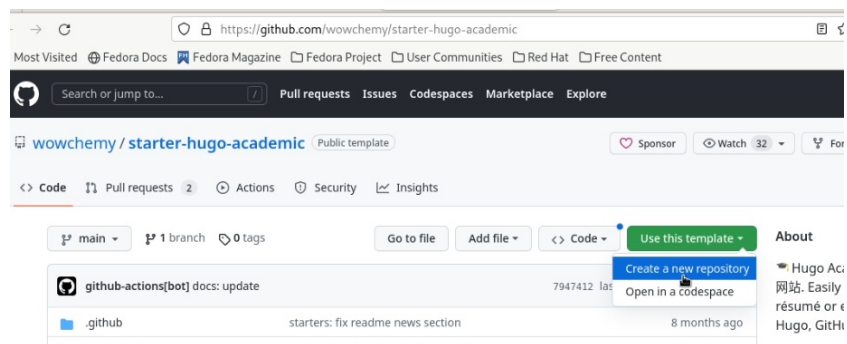


Рис. 4.8: Создание репозитория

При создании даём ему имя **blog**:

## Create a new repository from starter-hugo-academic

The new repository will start with the same files and folders as [wowchemy/starter-hugo-academic](#).

Owner \*

MikoGreen

Repository name \*

blog

Great repository names are [blog is available](#), memorable. Need inspiration? How about [fuzzy-adventure](#)?

Description (optional)

☒ Public



Anyone on the internet can see this repository. You choose who can commit.

☐ Private



You choose who can see and commit to this repository.

☐ Include all branches

Copy all branches from wowchemy/starter-hugo-academic and not just main.

You are creating a public repository in your personal account.

Create repository from template

Рис. 4.9: Создание репозитория

Проверка:

The screenshot shows the GitHub interface for a newly created repository named 'blog' by user 'MikoGreen'. The repository is public and was generated from the 'wowchemy/starter-hugo-academic' template. The main content area displays a table of files and folders, all marked as 'Initial commit'. The files include configuration files like '.github', '.editorconfig', '.gitignore', 'LICENSE.md', 'README.md', 'academic.Rproj', 'go.mod', and 'netlify.toml', as well as content folders like 'assets/media', 'config/\_default', 'content', 'data', 'images', and 'static/uploads'. The right sidebar shows repository statistics (0 stars, 0 forks, 1 watch) and sections for 'Releases' and 'Sponsors'.

File/Folder	Status	Commit
.github	Initial commit	now
assets/media	Initial commit	now
config/_default	Initial commit	now
content	Initial commit	now
data	Initial commit	now
images	Initial commit	now
static/uploads	Initial commit	now
.editorconfig	Initial commit	now
.gitignore	Initial commit	now
LICENSE.md	Initial commit	now
README.md	Initial commit	now
academic.Rproj	Initial commit	now
go.mod	Initial commit	now
netlify.toml	Initial commit	now

Рис. 4.10: Проверка

После клонируем данный репозиторий в путь `/home/narogozhina/work`:

```
[narogozhina@narogozhina work]$ git clone --recursive git@github.com:MikoGreen/blog.git blog
Клонирование в «blog»...
remote: Enumerating objects: 103, done.
remote: Counting objects: 100% (103/103), done.
remote: Compressing objects: 100% (91/91), done.
remote: Total 103 (delta 3), reused 80 (delta 0), pack-reused 0
Получение объектов: 100% (103/103), 5.88 МБ | 1.28 МБ/с, готово.
Определение изменений: 100% (3/3), готово.
[narogozhina@narogozhina work]$
```

Рис. 4.11: Клонирование репозитория

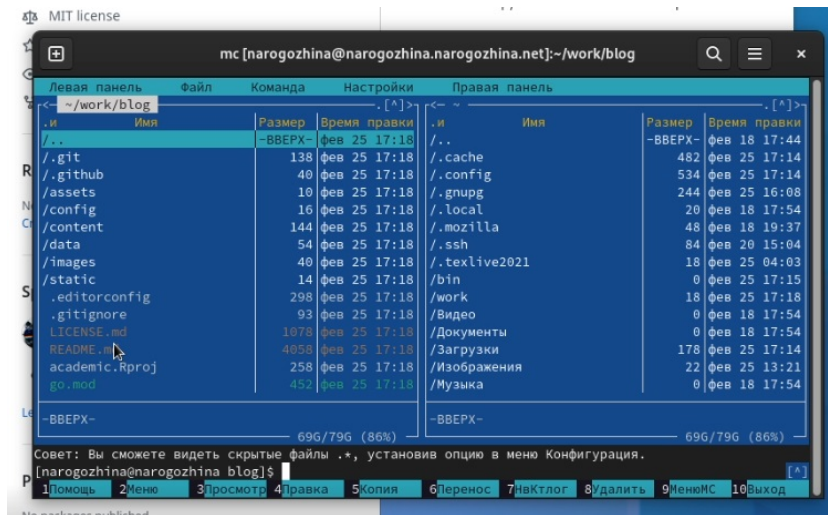


Рис. 4.12: Проверка

Переходим в папку *blog* и запускаем *hugo*:

```
[narogozhina@narogozhina blog]$ hugo
Error: failed to download modules: binary with name "go" not found
Total in 2 ms
[narogozhina@narogozhina blog]$ dnf i
```

Рис. 4.13: Hugo

Так как выдало ошибку, доустановим модуль **go** из-под суперпользователя:

```
[narogozhina@narogozhina blog]$ sudo -i
[sudo] пароль для narogozhina:
[root@narogozhina ~]# dnf install go
```

Рис. 4.14: Доустановка модуля “GO”

Возвращаемся в папку *blog* и запускаем *hugo*:

```
[narogozhina@narogozhina blog]$ hugo
hugo: downloading modules ...
hugo: collected modules in 44630 ms
Start building sites ...
hugo v0.110.0-e32a493b7826d02763c3b79623952e625402b168+extended linux/amd64 BuildDate=2023-01-17T12:
16:09Z VendorInfo=gohugoio

| EN
-----|-----
Pages | 55
Paginator pages | 0
Non-page files | 16
Static files | 9
Processed images | 37
Aliases | 15
Sitemaps | 1
Cleaned | 0

Total in 51952 ms
[narogozhina@narogozhina blog]$
```

Рис. 4.15: HUGO

После установки необходимых модулей проверяем создание папок и файлов:

```
[narogozhina@narogozhina blog]$ mc
[narogozhina@narogozhina blog]$ ls
academic.Rproj  config  data  go.sum  LICENSE.md  preview.png  README.md  static
assets          content  go.mod  images  netlify.toml  public  resources  theme.toml
[narogozhina@narogozhina blog]$
```

Рис. 4.16: Проверка

Удаляем каталог *public* с помощью *mc*:



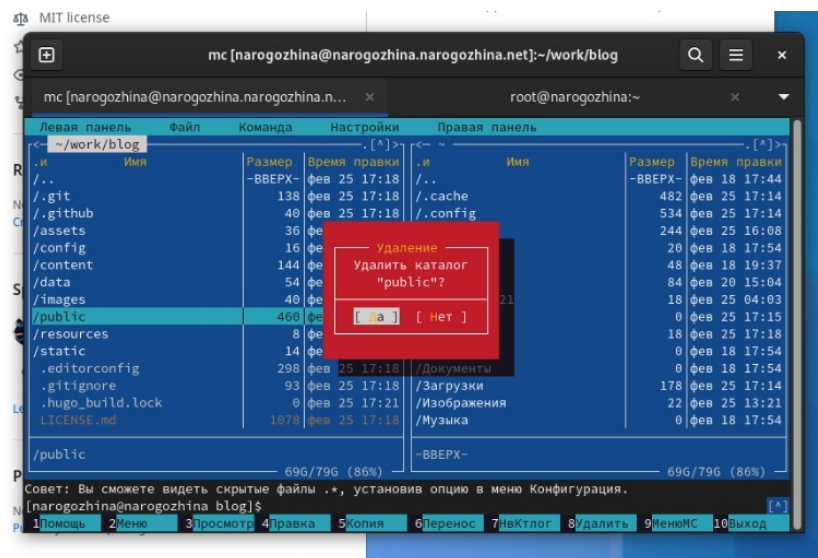


Рис. 4.17: Удаляем public

Запускаем *hugo server*:

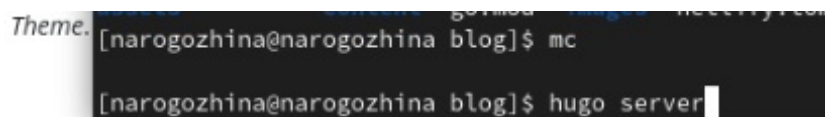


Рис. 4.18: Запуск hugo server

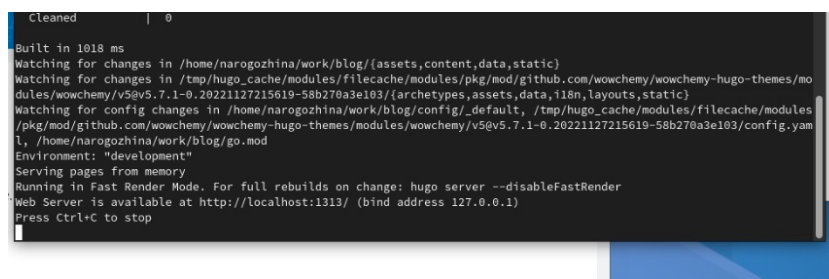


Рис. 4.19: Запуск hugo server

Открываем ссылку в браузере и видим сайт:

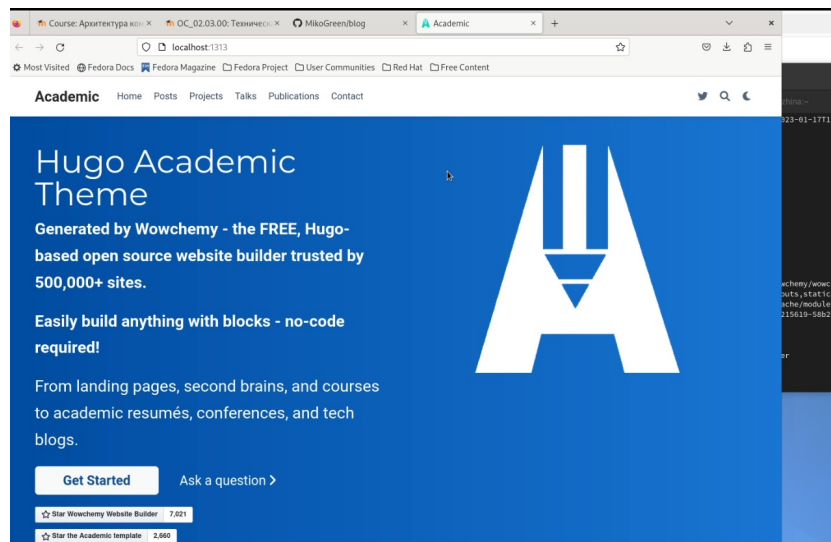


Рис. 4.20: Шаблон сайта

После проверки в браузере закроем сервер:

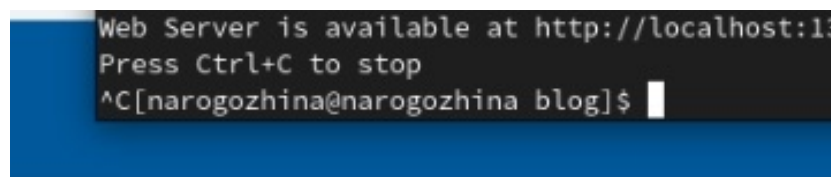


Рис. 4.21: Заккрытие hugo server

Создание нового репозитория:

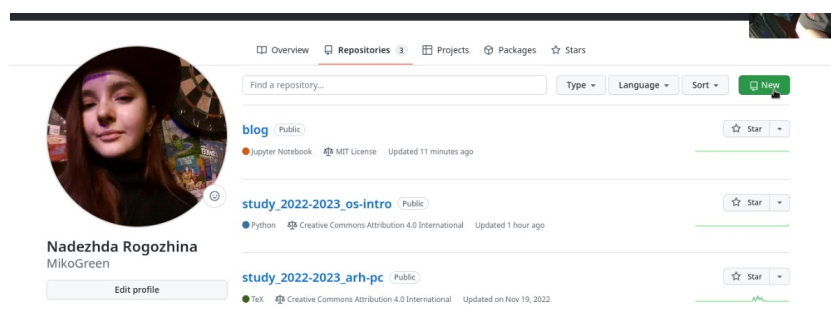


Рис. 4.22: Github

Название репозитория должно полностью совпадать с именем владельца + github.io:

Create a new repository

A repository contains all project files, including the revision history. Already have a project repository elsewhere? [Import a repository.](#)

Repository template

Start your repository with a template repository's contents.

No template ▾

Owner \* Repository name \*

MikoGreen / MikoGreen.github.io ✓

Great repository names are MikoGreen.github.io is available. Inspiration? How about [vigilant-octo-eureka?](#)

Description (optional)

Public  
Anyone on the internet can see this repository. You choose who can commit.

Private  
You choose who can see and commit to this repository.

Рис. 4.23: Создаем новый репозиторий

Возвращаемся в терминал, в папку work и клонируем туда наш репозиторий (свежесозданный):

```
[narogozhina@narogozhina work]$ git clone --recursive git@github.com:MikoGreen/MikoGreen.github.io.git
Клонирование в «MikoGreen.github.io»...
warning: Похоже, что вы клонировали пустой репозиторий.
[narogozhina@narogozhina work]$
```

Рис. 4.24: Клонирование репозитория

```
[narogozhina@narogozhina work]$ ls
blog MikoGreen.github.io study
[narogozhina@narogozhina work]$ m
```

Рис. 4.25: Проверка

Переключаемся на ветку “main”:

```
[narogozhina@narogozhina MikoGreen.github.io]$ git checkout -b main
Переключились на новую ветку «main»
[narogozhina@narogozhina MikoGreen.github.io]$
```

Рис. 4.26: Переключаемся на ветку “main”

Создаем пустой файл README.md, а затем коммитим все изменения и отправляем на github:

```
[narogozhina@narogozhina MikoGreen.github.io]$ touch README.md
[narogozhina@narogozhina MikoGreen.github.io]$ git add .
bash: git: команда не найдена...
Аналогичная команда: 'dot'
[narogozhina@narogozhina MikoGreen.github.io]$ git add .
[narogozhina@narogozhina MikoGreen.github.io]$ git commit -am 'Добавили README.md'
[main (корневой коммит) 31cd8bd] Добавили README.md
1 file changed, 0 insertions(+), 0 deletions(-)
create mode 100644 README.md
[narogozhina@narogozhina MikoGreen.github.io]$ git push origin main
Перечисление объектов: 3, готово.
Подсчет объектов: 100% (3/3), готово.
Запись объектов: 100% (3/3), 904 байта | 904.00 КиБ/с, готово.
Всего 3 (изменений 0), повторно использовано 0 (изменений 0), повторно использовано пакетов 0
To github.com:MikoGreen/MikoGreen.github.io.git
 * [new branch]      main -> main
[narogozhina@narogozhina MikoGreen.github.io]$
```

Рис. 4.27: Создание пустого файла и отправка изменений

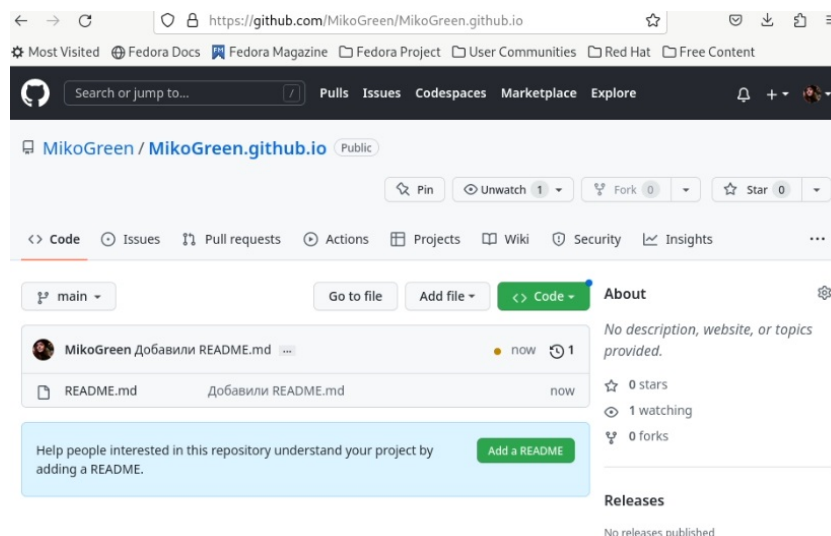


Рис. 4.28: Проверка

Создаем ветку подмодуля, клонируя репозиторий с нашего Github:

```
[narogozhina@narogozhina blog]$ git submodule add -b main git@github.com:MikoGreen/MikoGreen.github.io.git public
Клонирование в «/home/narogozhina/work/blog/public»...
remote: Enumerating objects: 3, done.
remote: Counting objects: 100% (3/3), done.
remote: Total 3 (delta 0), reused 3 (delta 0), pack-reused 0
Получение объектов: 100% (3/3), готово.
Следующие пути игнорируются одним из ваших файлов .gitignore:
public
подсказка: Use -f if you really want to add them.
подсказка: Turn this message off by running
подсказка: "git config advice.addIgnoredFile false"
fatal: Failed to add submodule 'public'
[narogozhina@narogozhina blog]$
```

Рис. 4.29: Создаем новую ветку

Так как вылезает ошибка `.gitignore`, то нужно её исправить прежде чем идти дальше - в файле `.gitignore` закомментируем папку `public`, чтобы этот путь не игнорировался:

```
mc [narogozhina@
.gitignore      [-M--]  1 L: [ 1+ 5  6/ 11]
# IDEs
.idea/
# Hugo
resources/
..#public/
jsconfig.json
node_modules/
go.sum
.hugo_build.lock
```

Рис. 4.30: Исправление ошибки `.gitignore`

```
[narogozhina@narogozhina blog]$ cat .gitignore
# IDEs
.idea/

# Hugo
resources/
#public/
jsconfig.json
node_modules/
go.sum
.hugo_build.lock
[narogozhina@narogozhina blog]$
```

Рис. 4.31: Проверка

Повторяем команду:

```
[narogozhina@narogozhina blog]$ git submodule add -b main git@github.com:MikoGreen/MikoGreen.github.io.git public
Adding existing repo at 'public' to the index
[narogozhina@narogozhina blog]$
```

Рис. 4.32: Повтор введенной команды

После выполнения запускаем hugo:

```
Adding existing repo at 'public' to the index
[narogozhina@narogozhina blog]$ hugo
Start building sites ...
hugo v0.110.0-e32a493b7826d02763c3b79623952e625402b168+extended linux/amd64 BuildDate=2023-01-17T12:16:09Z VendorInfo=gohugoio
```

	EN
Pages	55
Paginator pages	0
Non-page files	16
Static files	9
Processed images	37
Aliases	15
Sitemaps	1
Cleaned	0

```
Total in 1276 ms
[narogozhina@narogozhina blog]$
```

Рис. 4.33: HUGO

Проверим подключение каталога к репозиторию командой **git remote -v**:

```

remote
[narogozhina@narogozhina public]$ git remote -v
origin  git@github.com:MikoGreen/MikoGreen.github.io.git (fetch)
origin  git@github.com:MikoGreen/MikoGreen.github.io.git (push)
[narogozhina@narogozhina public]$

```

Рис. 4.34: Проверка

Добавим изменения на github:

```

origin  git@github.com:MikoGreen/MikoGreen.github.io.git (push)
[narogozhina@narogozhina public]$ git add .
[narogozhina@narogozhina public]$ git commint -am 'Добавили сайт'
git: «commint» не является командой git. Смотрите «git --help».

Самые похожие команды:
  commit
[narogozhina@narogozhina public]$ git commit -am 'Добавили сайт'

```

Рис. 4.35: Загружаем обновления

```

create mode 100644 webfonts/fa-v4compatibility.woff2
[narogozhina@narogozhina public]$ git push origin main
Перечисление объектов: 237, готово.
Подсчет объектов: 100% (237/237), готово.
Сжатие объектов: 100% (197/197), готово.
Запись объектов: 100% (236/236), 6.88 МиБ | 998.00 КиБ/с, готово.
Всего 236 (изменений 56), повторно использовано 0 (изменений 0), повторно использовано пакетов 0
remote: Resolving deltas: 100% (56/56), done.
To github.com:MikoGreen/MikoGreen.github.io.git
  31cd8bd..f37cf85  main -> main
[narogozhina@narogozhina public]$

```

Рис. 4.36: Загружаем обновления

Проверка обновлений:

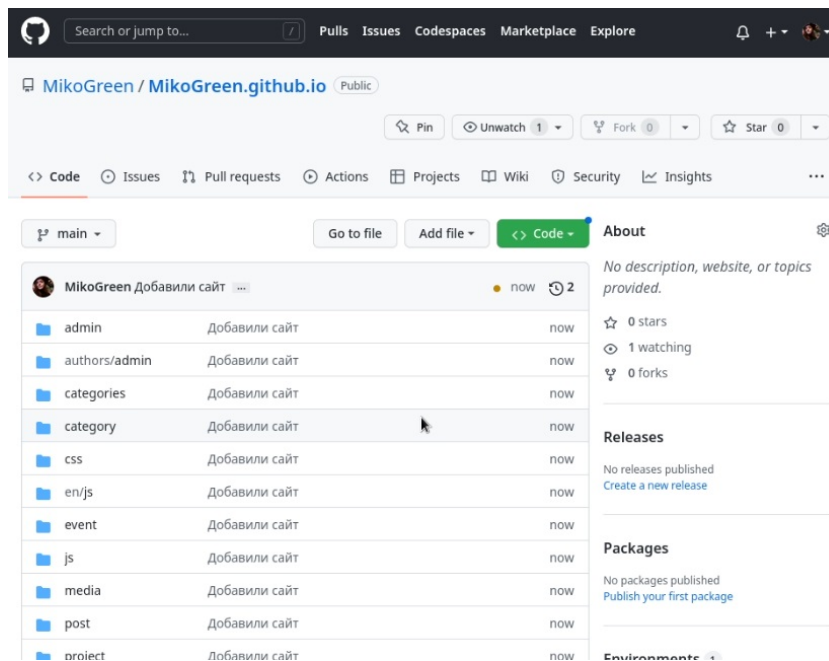


Рис. 4.37: Проверка github

Открываем наш сайт:



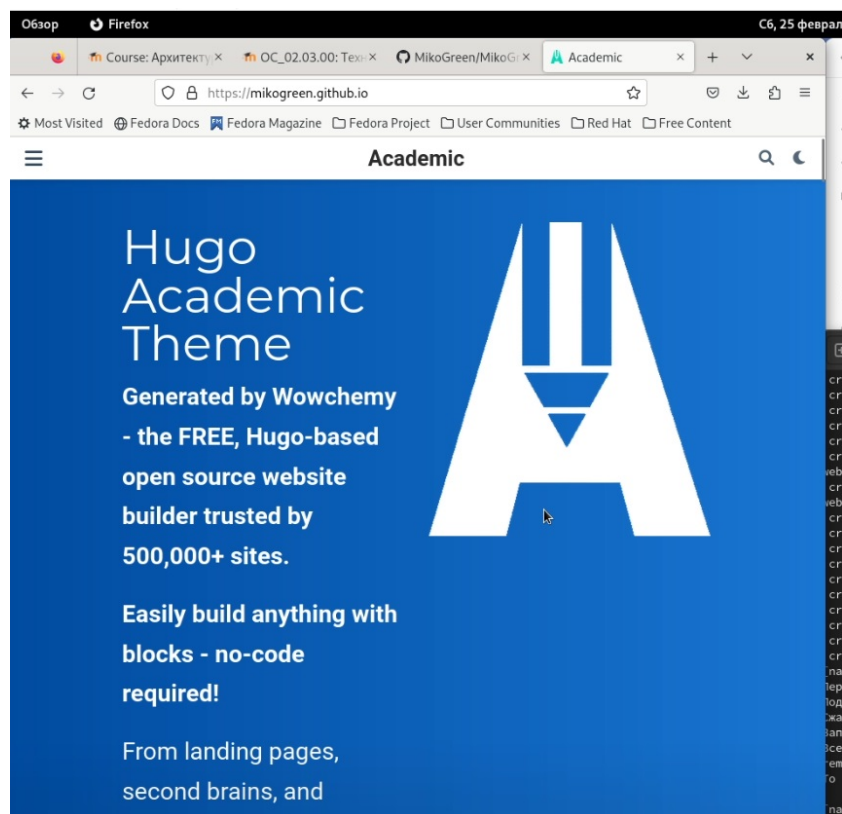


Рис. 4.38: Шаблон сайта готовый

## 5 Выводы

В ходе данной лабораторной работы я создала шаблон своего сайта, который в будущем буду дорабатывать, а также закрепила навыки работы с системой контроля версий *Git*.

# Список литературы

1. Этапы реализации проекта
2. Техническая реализация проекта
3. Руководство по выполнению первого этапа индивидуального проекта
4. Инструменты Git - Подмодули