

Отчет по первому этапу индивидуального проекта

Размещение на Github pages заготовки для персонального сайта

Старовойтов Егор Сергеевич

Содержание

Цель работы	1
Задание	1
Теоретическое введение	2
Hugo	2
Основные преимущества Hugo	2
Документация	2
Структура.....	2
Github pages.....	4
Выполнение лабораторной работы	4
Шаг 1 - установка исполняемого файла Hugo.....	4
Шаг 2 - создание репозитория на основе шаблона.....	4
Шаг 3 - запуск Hugo.....	5
Шаг 4 - Удаляем файл demo.md	6
Шаг 5 - Создаем репозиторий PrototypeRailGun.github.io	7
Шаг 5 - Клонирование репозитория PrototypeRailGun.github.io	7
Шаг 6 - Создание ветки main	8
Шаг 7 - создаем README.md файл	8
Шаг 8 - Настраиваем рабочий процесс	8
Шаг 9 - Генерируем файлы сайта	9
Шаг 10 - Синхронизируем public с репозиторием	10
Вывод.....	11

Цель работы

Разместить на Github pages заготовки для персонального сайта.

Задание

1. Установить необходимое программное обеспечение.

2. Скачать шаблон темы сайта.
3. Разместить его на хостинге git.
4. Установить параметр для URLs сайта.
5. Разместить заготовку сайта на Github pages.

Теоретическое введение

Hugo

Для реализации сайта используется генератор статических сайтов Hugo.

Hugo — один из самых популярных генераторов статических сайтов с открытым исходным кодом, написан на языке Go. Благодаря своей удивительной скорости и гибкости, Hugo делает создание веб-сайтов увлекательным.

Основные преимущества Hugo

- Очень быстрый и гибкий
- Для него легко настроить хостинг
- Безопасный
- Хорошая структура исходников
- Возможность хранить содержимое в удобном формате (YAML, JSON или TOML)
- Поддержка тем. Есть готовый набор тем, более 200
- Легко SEO-оптимизировать
- i18n с коробки
- Хорошая поддержка таксономии
- Быстрый в освоении. Исчерпывающая документация










Документация

Фреймворк имеет очень хорошую документацию. Она доступна только на английском языке. Информация очень хорошо структурирована, что позволяет освоить данную технологию, за несколько дней. Для лучшего восприятия, практически в каждой главе есть обучающее видео от разработчиков. Все это позволяет очень быстро приступить к созданию собственных сайтов.

Структура

После установки фреймворка, сайт можно легко создать с помощью команды: `hugo new site website-name`

Далее hugo сгенерирует следующую структуру проекта:

```
>  archetypes
>  content
>  data
>  i18n
>  public
>  resources
>  static
>  themes
   config.toml
```

Content

Основной контент или содержимое сайта храниться в формате .md в папке content. В роле контента могут выступать ваши статьи, новости, продукты интернет магазина и прочее.

Data

Каталог «data» используется для хранения файлов конфигурации, которые Hugo может использовать при создании вашего веб-сайта. Вы можете записать эти файлы в формате YAML, JSON или TOML.

Archetypes

Архетипы используют для создания содержимого сайта на основе заготовок. Это экономит время и обеспечивает единообразие для сайтов, использующих несколько типов контента. Вы также можете создавать свои собственные архетипы с предварительно настроенными полями основного материала.

I18n

Этот каталог предназначен для хранения конфигурации сайта на различных языках.

Resources

Hugo использует этот каталог для хранения кеша. Это ускоряет сборку сайта.

Static

Здесь храниться весь статический контент (CSS, JavaScript, и т.п).

Layouts

Хранит шаблоны в виде файлов .html, которые определяют, как просмотры вашего контента будут отображаться на статическом веб-сайте.

Themes

Для хранения различных тем.

Public

Сгенерированные исходники веб-сайта. Именно эту директорию следует заливать на хостинг.

Github pages

Для размещения статического веб-сайта можно воспользоваться сервисом GitHub Pages. Статический сайт состоит из HTML-страниц с неизменным содержимым. Такие сайты могут использоваться как визитки, портфолио, презентационные страницы.

Сервис Github Pages предоставляет следующие возможности для статических сайтов:

- Использование HTML, CSS, языка разметки Markdown;
- Встраивание изображений и другого медиа;
- Использование JavaScript.

Ограничения:

- Нельзя использовать на сайте PHP, Python и другие серверные языки;
- Серверный код и серверные скрипты выполняться не будут;
- Cookies не используются.

Выполнение лабораторной работы

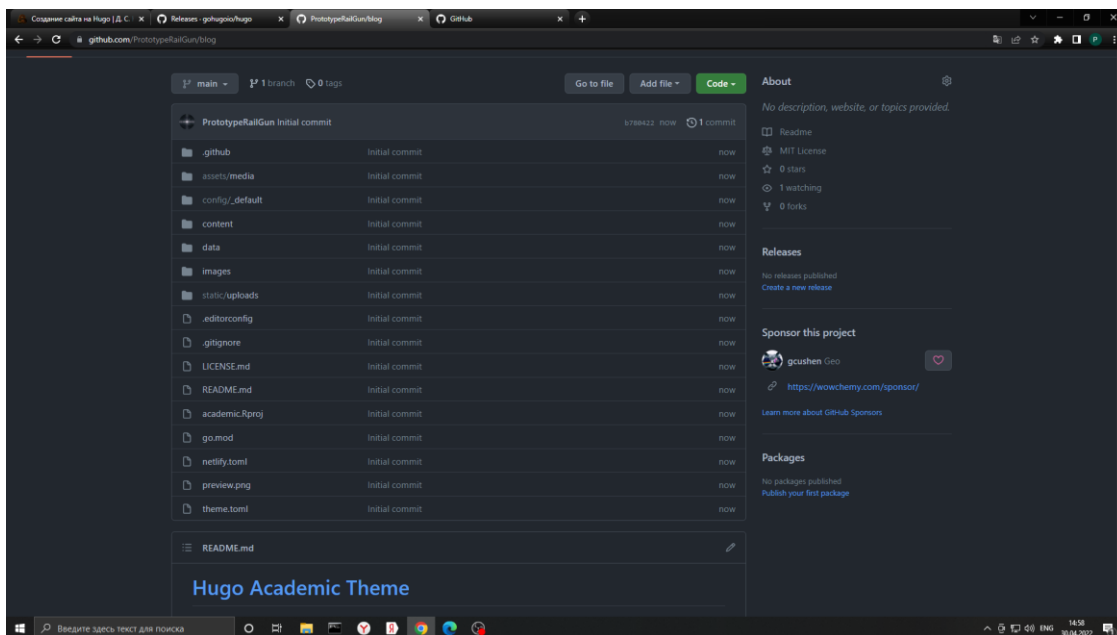
Шаг 1 - установка исполняемого файла Hugo

Шаг 2 - создание репозитория на основе шаблона

Шаблон Hugo Academic Theme: - Демо-сайт: <https://academic-demo.netlify.app/> -

Репозиторий: <https://github.com/wowchemy/starter-hugo-academic>

На фотографии ниже представлен созданный на основе этого шаблона репозиторий blog:



Созданный на основе шаблона репозиторий

Шаг 3 - запуск Hugo.

На этом этапе нужно перейти в папку с клонированным репозиторием blog, и находясь в ней запустить исполняемый файл hugo с аргументом server. При попытке это сделать возникла ошибка: "Error: failed to download modules: binary with name 'go' not found". Это связано с тем, что у меня на компьютере не было необходимой установки языка программирования Go.

```
D:\Операционные системы\project\work\blog>..\bin\hugo server
Error: failed to download modules: binary with name "go" not found
D:\Операционные системы\project\work\blog>
```

Отсутствует go

Устанавливаю язык программирования Go с помощью командной консоли Windows Power Shell и пакетного менеджера chocolatey:

```

PS C:\Users\proto> choco install -y golang
Chocolatey v1.1.0
Installing the following packages:
golang
By installing, you accept licenses for the packages.
Progress: Downloading golang 1.18.1... 100%

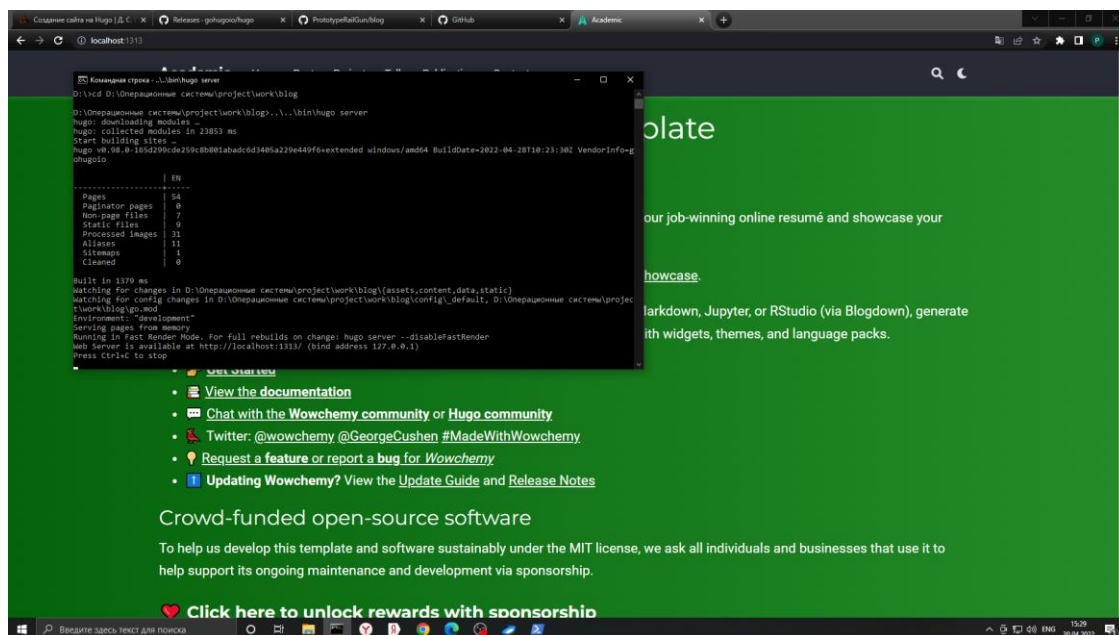
golang v1.18.1 [Approved]
golang package files install completed. Performing other installation steps.
Downloading golang 64 bit
from 'https://golang.org/dl/go1.18.1.windows-amd64.msi'
Progress: 100% - Completed download of C:\Users\proto\AppData\Local\Temp\chocolatey\golang\1.18.1\go1.18.1.windows-amd64.msi (132.46 MB).
Download of go1.18.1.windows-amd64.msi (132.46 MB) completed.
Hashes match.
Installing golang...
golang has been installed.
golang may be able to be automatically uninstalled.
Environment Vars (like PATH) have changed. Close/reopen your shell to
see the changes (or in powershell/cmd.exe just type `refreshenv`).
The install of golang was successful.
Software installed as 'msi', install location is likely default.

Chocolatey installed 1/1 packages.
See the log for details (C:\ProgramData\chocolatey\logs\chocolatey.log).
PS C:\Users\proto>

```

Установка Go

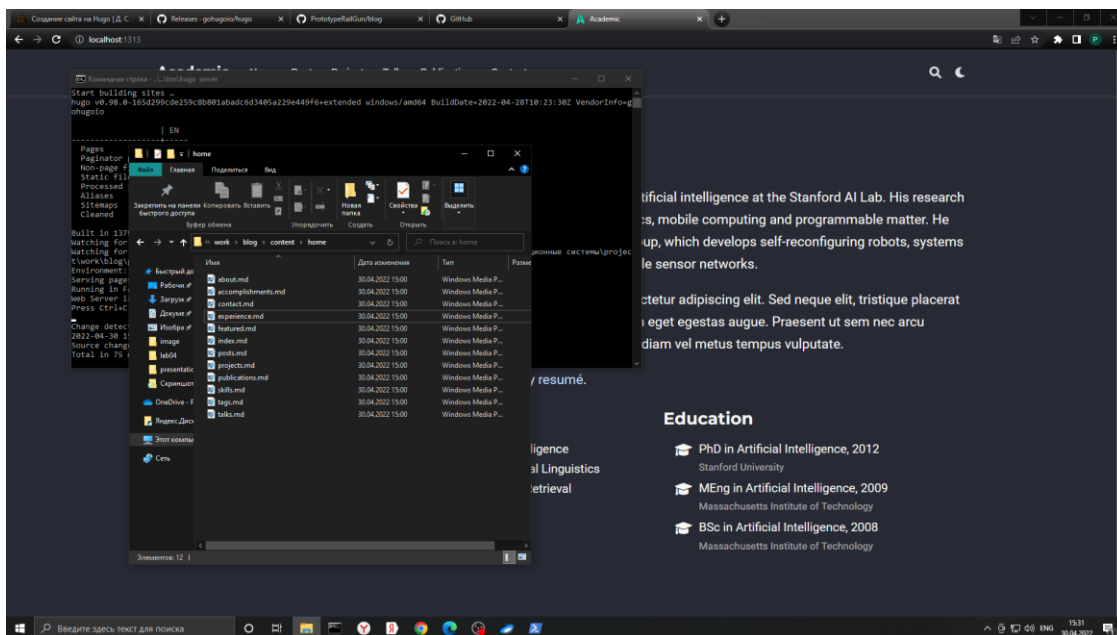
Далее я повторяю попытку выполнить `hugo server`, на этот раз успешно. На этом этапе сайт можно увидеть только на моем компьютере. Копирую ссылку `"http://localhost:1313/"` и перехожу по ней в браузере:



Заготовка сайта на локальном сервере

Шаг 4 - Удаляем файл `demo.md`

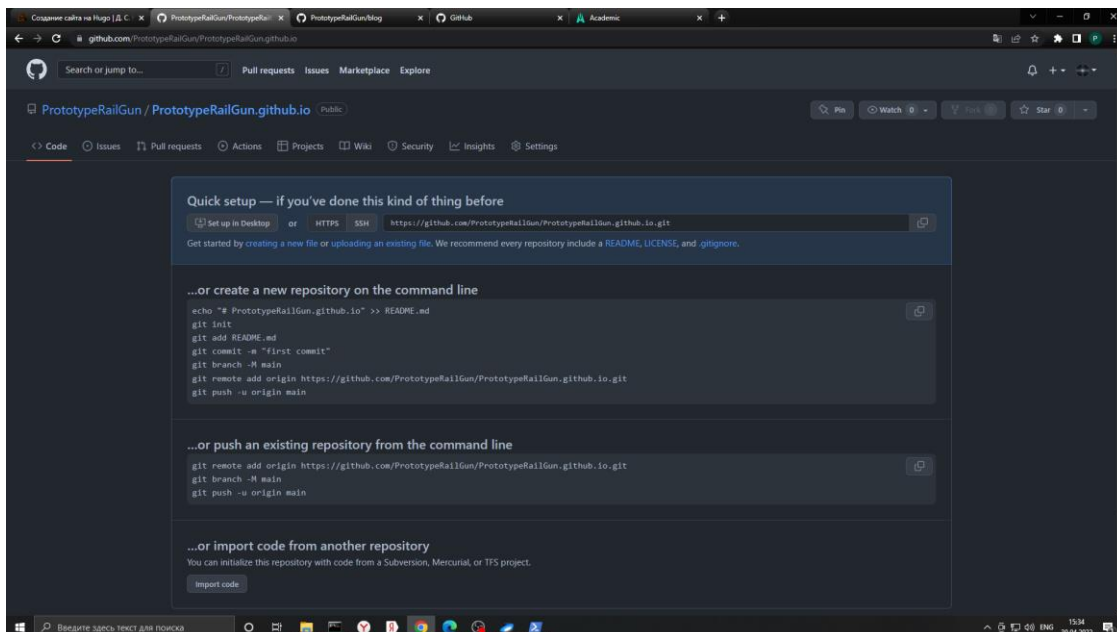
Файл `demo.md` находится в папке `content/home/` и представляет собой зеленую шапку сайта. На фото ниже видно сайт уже без шапки и папку `home` без файла `demo.md`.



Файл *demo.md* удален

Шаг 5 - Создаем репозиторий PrototypeRailGun.github.io

Завершаю hugo server и приступаю к созданию репозитория со специальным именем. “github.io” в имени репозитория означает, что этот репозиторий будет являться хостингом моего будущего сайта. PrototypeRailGun - мой ник на гитхабе.



Репозиторий *PrototypeRailGun.github.io*

Шаг 5 - Клонирование репозитория PrototypeRailGun.github.io

Клонирую репозиторий PrototypeRailGun.github.io на рабочий компьютер рядом с клоном репозитория blog.

```
D:\Операционные системы\project\work>git clone --recursive https://github.com/PrototypeRailGun/PrototypeRailGun.github.io.git
Cloning into 'PrototypeRailGun.github.io'...
warning: You appear to have cloned an empty repository.
D:\Операционные системы\project\work>
```

Клонирование *PrototypeRailGun.github.io*

Шаг 6 - Создание ветки main

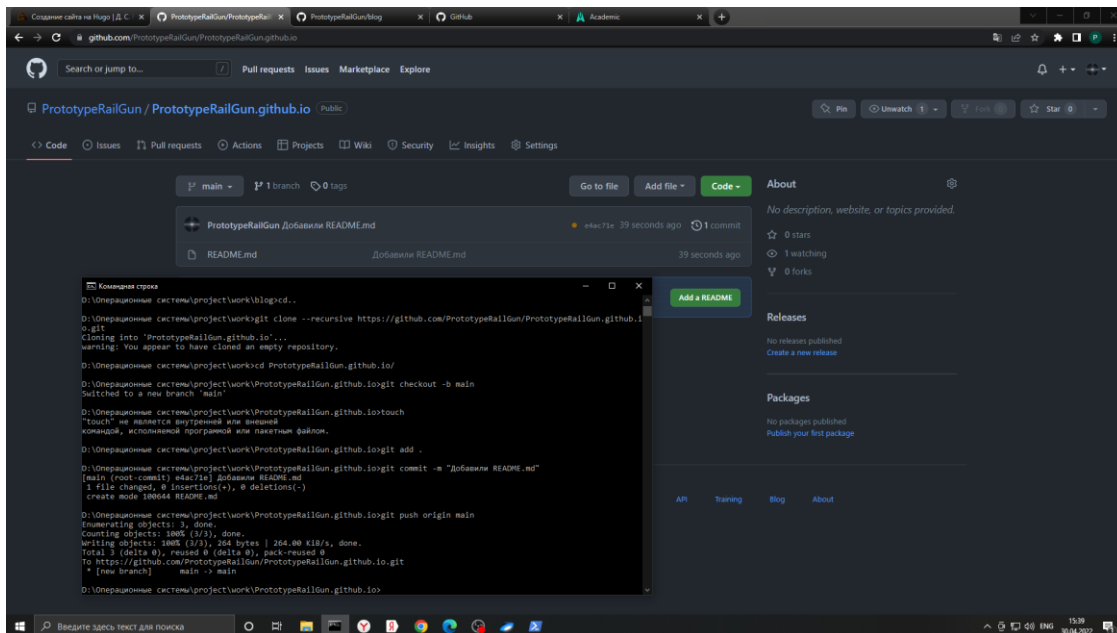
main - основная ветка репозитория *PrototypeRailGun.github.io*.

```
D:\Операционные системы\project\work\PrototypeRailGun.github.io>git checkout -b main
Switched to a new branch 'main'
D:\Операционные системы\project\work\PrototypeRailGun.github.io>
```

Создание ветки *main*

Шаг 7 - создаем README.md файл

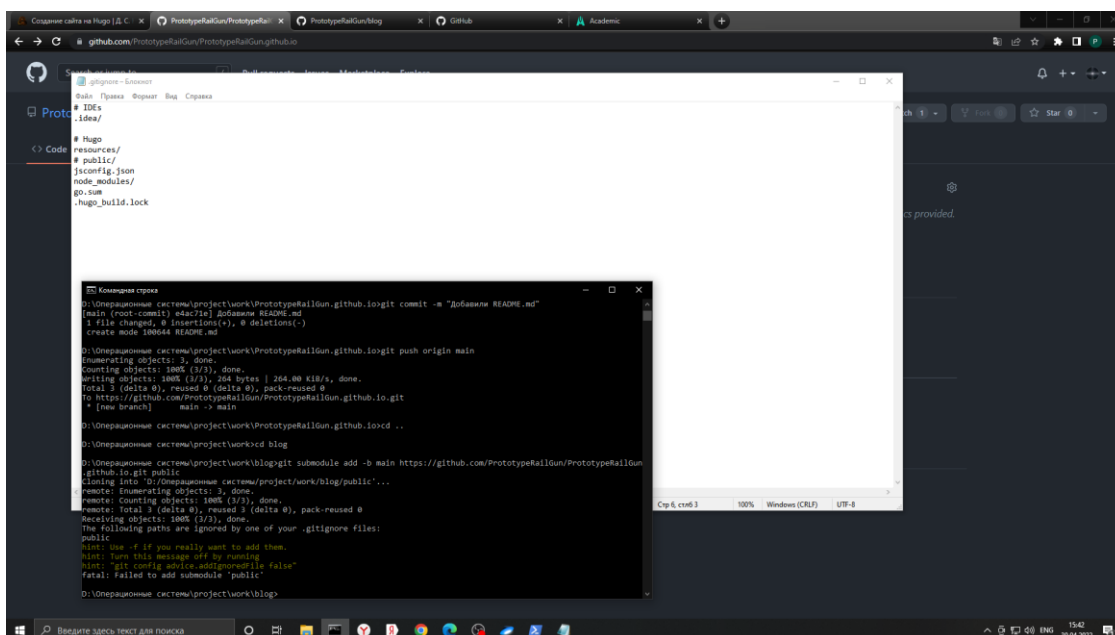
Создаем README.md файл в репозитории *PrototypeRailGun.github.io* и отправляем изменения на сервер.



Создание *README.md*

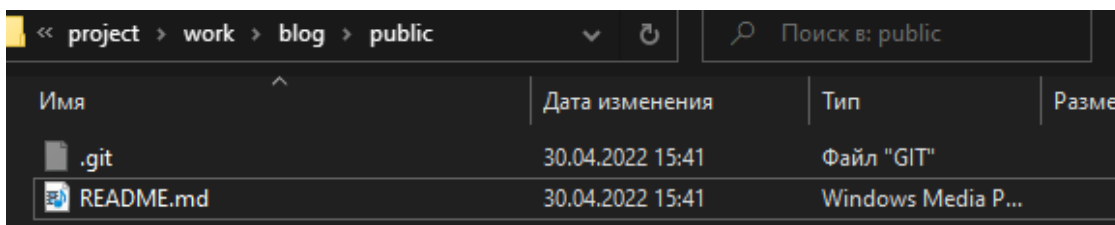
Шаг 8 - Настраиваем рабочий процесс

Напрямую в *PrototypeRailGun.github.io* мы загружать файлы не будем. Выполним команду, которая подключит репозиторий *PrototypeRailGun.github.io* к папке *public* репозитория *blog*. Первая попытка заканчивается ошибкой, так как в файле *.gitignore* сказано игнорировать *public*. Отменим это превратив строчку с *public* в комментарий.



Подключение репозитория

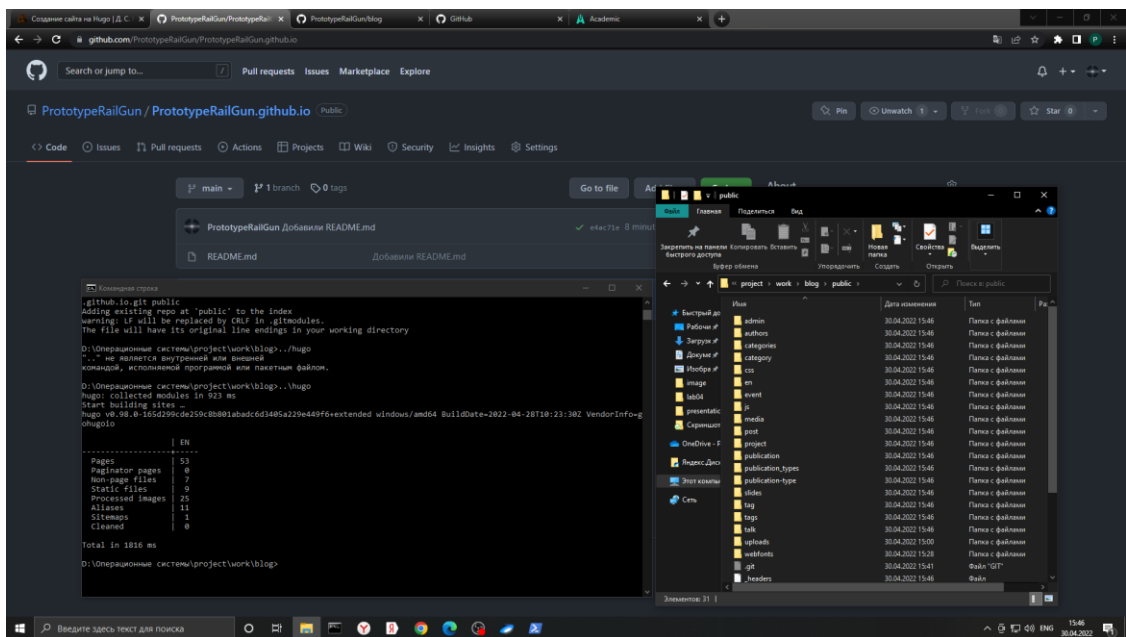
Сейчас в папке public лежит только файл README.md (не считая скрытого .git).



Содержание public

Шаг 9 - Генерируем файлы сайта

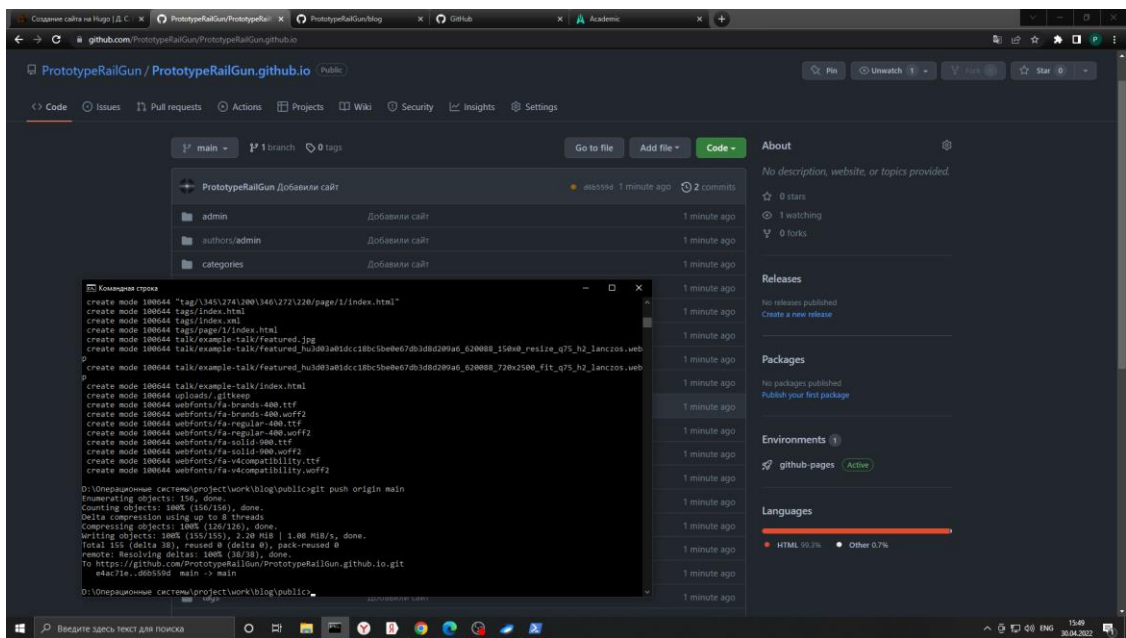
Находясь в каталоге blog запускаем hugo и видим, как в папке public автоматически появились файлы сайта.



Сгенерированные файлы

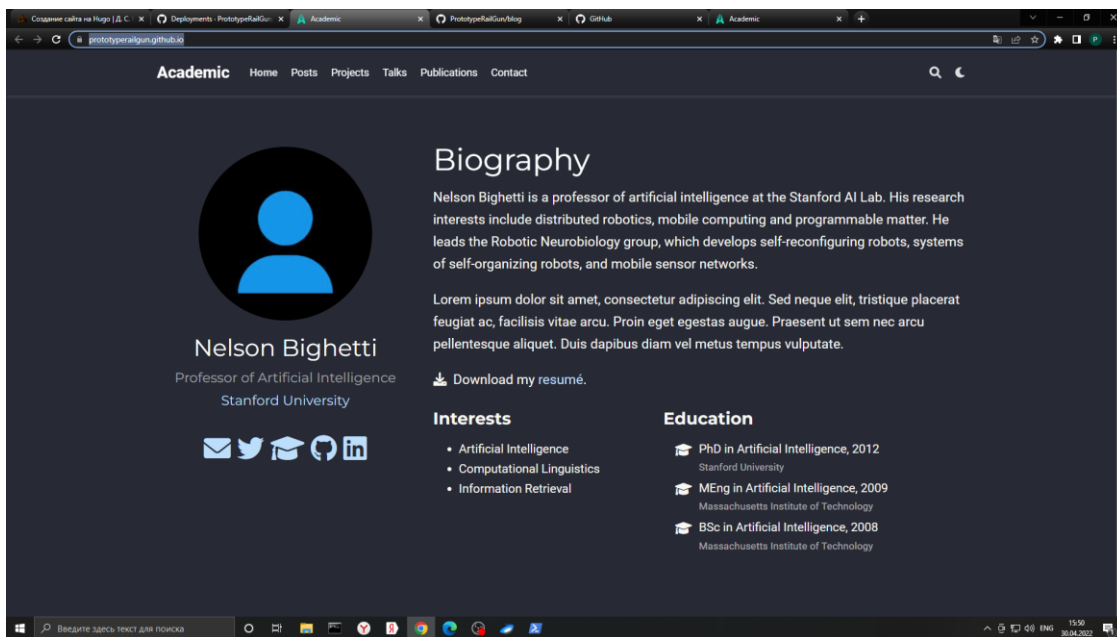
Шаг 10 - Синхронизируем public с репозиторием

Выполняем команды `git add`, `git commit`, `git push` - и созданные на предыдущем шаге файлы теперь на github в репозитории `PrototypeRailGun.github.io`.



Файлы на сервере

Находим наш сайт в интернете и видим все изменения.



Деплой сайта

Вывод

Установлено нужное программное обеспечения, заготовка сайта размещена на github pages. Задание первого этапа индивидуального проекта полностью выполнено.