Отчёт по 1-му этапу персонального проекта

Размещение на Github pages заготовки для персонального сайта.

Ганина Таисия, НКАбд-01-22

Содержание

# 1 Цель работы

Создать репозиторий для дальнейшей работы с проектом. Создать URL для сайта и оформить шаблон.

# 2 Задание

1. Установить необходимое программное обеспечение.
2. Cкачать шаблон темы сайта.
3. Разместить его на хостинге git.
4. Установить параметр для URLs сайта.
5. Разместить заготовку сайта на Github pages.

# 3 Теоретическое введение

В ходе выполнения первого этапа проекта было произведено много работы с использованием системы контроля версий Git. Так что, как я считаю, будет уместно привести часть команд и подробнее рассказать про эту систему.

Системы контроля версий (Version Control System, VCS) применяются при работе нескольких человек над одним проектом. Обычно основное дерево проекта хранится в локальном или удалённом репозитории, к которому настроен доступ для участников проекта. При внесении изменений в содержание проекта система контроля версий позволяет их фиксировать, совмещать изменения, произведённые разными участниками проекта, производить откат к любой более ранней версии проекта, если это требуется.

В классических системах контроля версий используется централизованная модель, предполагающая наличие единого репозитория для хранения файлов. Выполнение большинства функций по управлению версиями осуществляется специальным сервером. Участник проекта (пользователь) перед началом работы посредством определённых команд получает нужную ему версию файлов. После внесения изменений, пользователь размещает новую версию в хранилище. При этом предыдущие версии не удаляются из центрального хранилища и к ним можно вернуться в любой момент. Сервер может сохранять не полную версию изменённых файлов, а производить так называемую дельта-компрессию — сохранять только изменения между последовательными версиями, что позволяет уменьшить объём хранимых данных.

Системы контроля версий поддерживают возможность отслеживания и разрешения конфликтов, которые могут возникнуть при работе нескольких человек над одним файлом. Можно объединить (слить) изменения, сделанные разными участниками (автоматически или вручную), вручную выбрать нужную версию, отменить изменения вовсе или заблокировать файлы для изменения. В зависимости от настроек блокировка не позволяет другим пользователям получить рабочую копию или препятствует изменению рабочей копии файла средствами файловой системы ОС, обеспечивая таким образом, привилегированный доступ только одному пользователю, работающему с файлом.

Системы контроля версий также могут обеспечивать дополнительные, более гибкие функциональные возможности. Например, они могут поддерживать работу с несколькими версиями одного файла, сохраняя общую историю изменений до точки ветвления версий и собственные истории изменений каждой ветви. Кроме того, обычно доступна информация о том, кто из участников, когда и какие изменения вносил. Обычно такого рода информация хранится в журнале изменений, доступ к которому можно ограничить.

В отличие от классических, в распределённых системах контроля версий центральный репозиторий не является обязательным.

Среди классических VCS наиболее известны CVS, Subversion, а среди распределённых — Git, Bazaar, Mercurial. Принципы их работы схожи, отличаются они в основном синтаксисом используемых в работе команд.

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.

# 4 Выполнение лабораторной работы

1. Установить необходимое программное обеспечение (рис. [1](#fig:001), [2](#fig:002), [3](#fig:003), [4](#fig:004)).

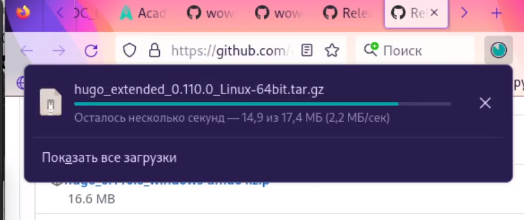


Figure 1: Скачивание нужного архива

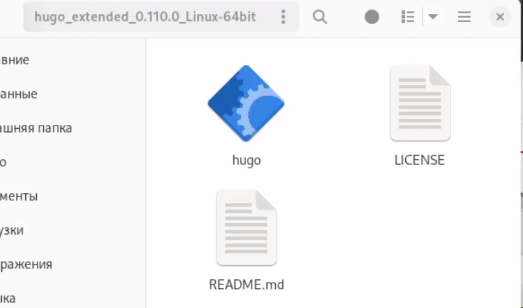


Figure 2: Распаковка архива

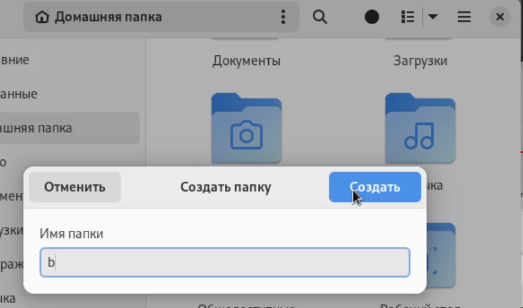


Figure 3: Создание папки bin для использования hugo

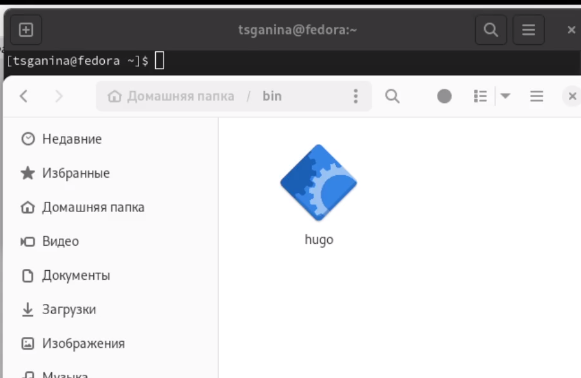


Figure 4: Перенос исполняемого файла в папку bin

1. Cкачать шаблон темы сайта (рис. [5](#fig:005))

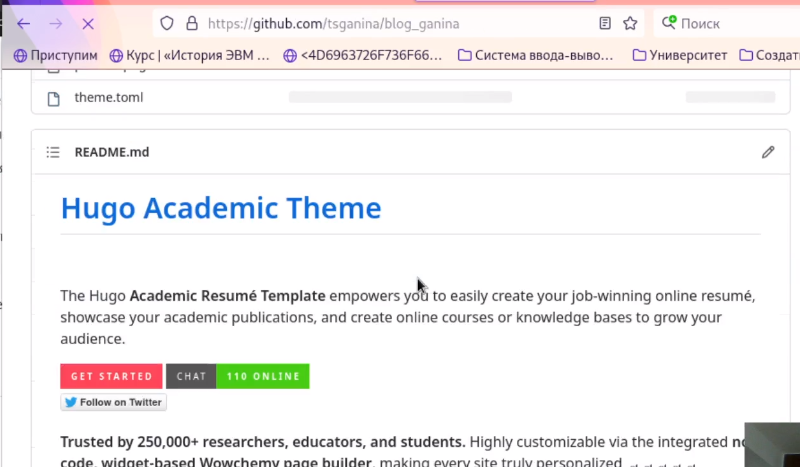


Figure 5: Нужный репозиторий

1. Разместить его на хостинге git (рис. [6](#fig:006), [7](#fig:007), [8](#fig:008), [9](#fig:009), [10](#fig:010), [11](#fig:011)).

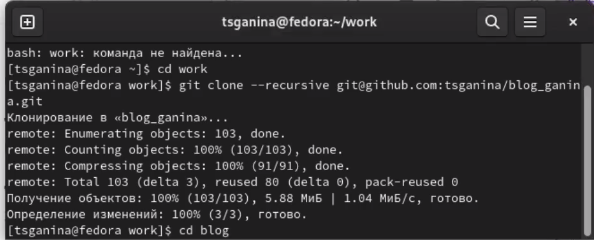


Figure 6: Клонирование репозитория

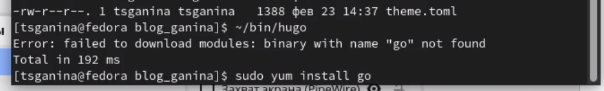


Figure 7: Установка модуля golang

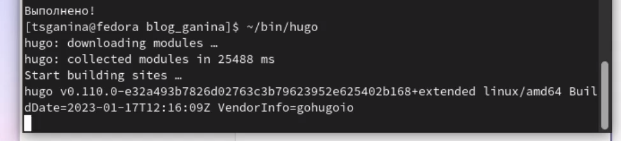


Figure 8: Bulding site

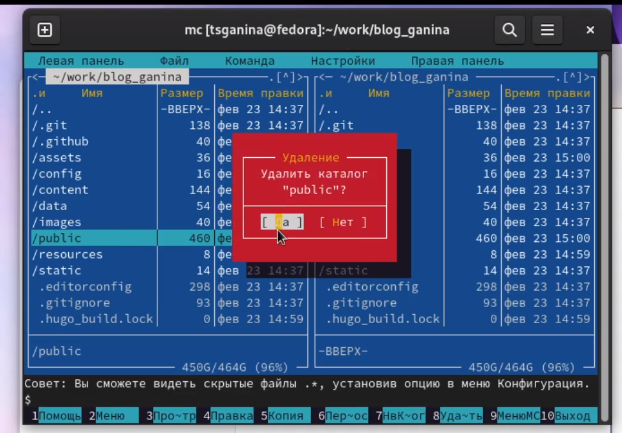


Figure 9: Удаление каталога public

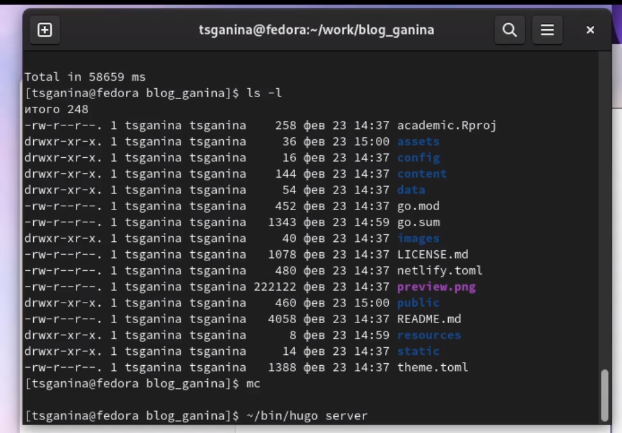


Figure 10: Подключение к серверу

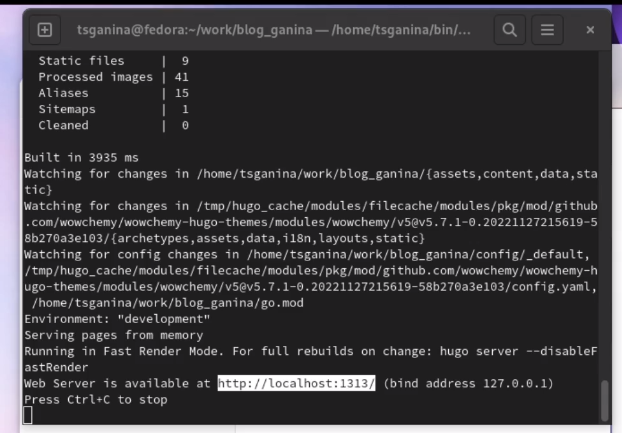


Figure 11: Локальная страничка

1. Установить параметр для URLs сайта (рис. [12](#fig:012), [13](#fig:013), [14](#fig:014), [15](#fig:015), [16](#fig:016)).

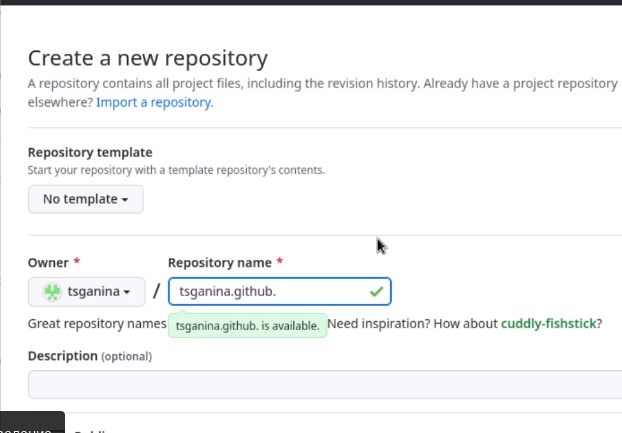


Figure 12: Создание специального репозитория

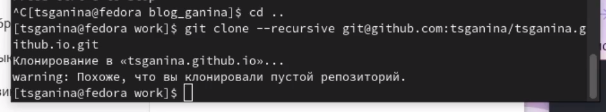


Figure 13: Клонирование пока что пустого репозитория

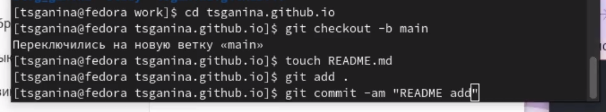


Figure 14: Переключение на новую ветку и создание файла README

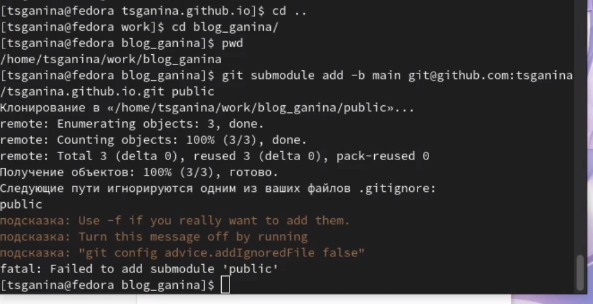


Figure 15: Создание модуля в папке public

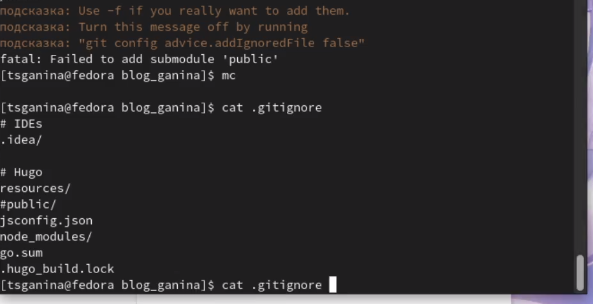


Figure 16: Убрать public из списка игнорируемых

1. Разместить заготовку сайта на Github pages.

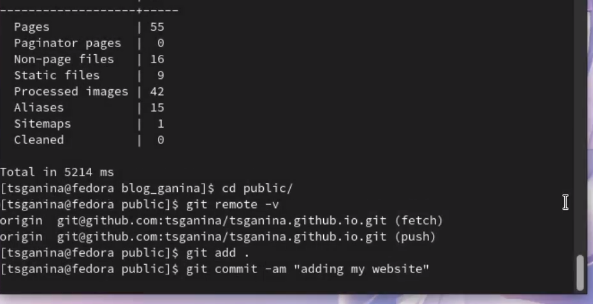


Figure 17: Добавление новых данных на Github

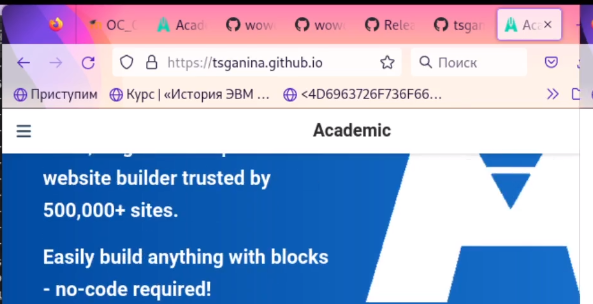


Figure 18: Готовый URL

# 5 Выводы

В ходе этой работы был создан репозиторий для дальнейшей работы с проектом, а также URL для сайта.

# Список литературы

1. [Инструкция по выполнению первого этапа проекта](https://www.youtube.com/watch?v=OpsSv0RE3C4)