**Министерство науки и высшего образования Российской Федерации**

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

**НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО**

**ITMO University**

**ОТЧЁТ ПО ЛАБОРАТОРНОЙ РАБОТЕ 2**

**По дисциплине** Web-программирование

**Тема работы** Git, Gulp

**Обучающийся** Бабаев Руслан Сагибович

**Факультет** Факультет инфокоммуникационных технологий

**Группа** К3321

**Направление подготовки** 11.03.02 Инфокоммуникационные технологии и системы связи

**Образовательная программа** Программирование в инфокоммуникационных системах

**Обучающийся** 15.10.2024  Бабаев Р.С.

(дата) (подпись) (Ф.И.О.)

**Руководитель**  Марченко Е.В.

(дата) (подпись) (Ф.И.О.)

Санкт-Петербург  
2024 г.

# Цель

Целью данной работы является рассмотрение на практике работу с системой контроля версий Git и автоматизирующим сборщиком gulp.

# Задачи

1. Повторить работу с репозиториями в GitHub – коммиты, ветки, форки;
2. Изучить основы Gulp;
3. Применить и закрепить знания Gulp на практических примерах.

# Ход работы

## **Часть 1**

Для решения этой части в директории lab2/git\_task был создан файл ex1.html. В нем сделан ряд изменений, каждые из которых фиксировались в git и локальный репозиторий синхронизировался с GitHub. На рисунке 1 можно увидеть, как выглядит веб-страница после всех изменений.



Рисунок 1 – Итоговый вид веб-страницы

Также на рисунке 2 изображен пример того, как изменения сначала сохраняются локально, а затем отправляются на удаленный репозиторий в заранее созданной ветке.

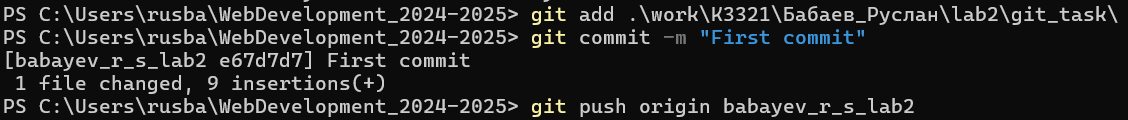


Рисунок 2 – Примеры использования команд git add, commit и push

На рисунке 3 можно увидеть, как все эти коммиты выглядят на Github.



Рисунок 3 – Отображение коммитов на Github

Ссылка на эту часть проекта в GitHub:   
<https://github.com/Jesusya-26/WebDevelopment_2024-2025/tree/babayev_r_s_lab2/work/K3321/Бабаев_Руслан/lab2/git_task>

## **Часть 2**

В этом задании необходимо установить gulp на свой компьютер. Gulp (Global Utility for Liquid Programming) — это инструмент, предназначенный для автоматизации повторяющихся задач в процессе разработки. Он позволяет создавать потоки обработки файлов и выполнять над ними различные операции. Для его установки вам также понадобится загрузить Node.js и пакетный менеджер NPM.

Первым шагом нужно установить nvm (Node Version Manager) — это менеджер версий Node.js (см. рисунок 4).

Изображение выглядит как текст, электроника, снимок экрана, программное обеспечение

Автоматически созданное описание

Рисунок 4 – Установка NVM

Затем устанавливаем сам Node.js, при этом автоматически устанавливается и пакетный менеджер npm (выполняем команду nvm install 20.6.0). После установки выбираем версию для использования (выполняем команду nvm use 20.6.0).

Теперь можно установить gulp с помощью пакетного менеджера npm. Параметр -g необходим для глобальной установки Gulp, что позволит запускать его из любого каталога.

Далее создаём проект командой npm init -y (параметр -y отвечает за автоматическое создание файла метаданных package.json). Результат представлен на рисунке 5.

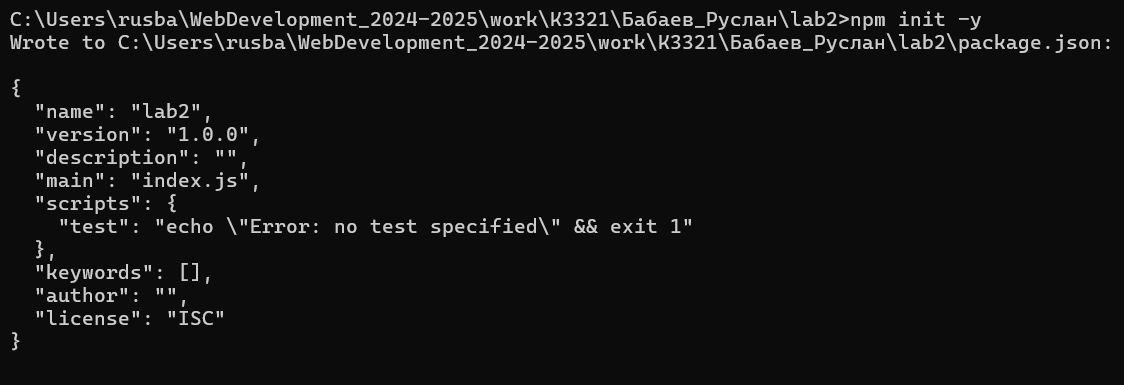


Рисунок 5 – Создание проекта

После этого устанавливаем gulp локально в проект и проверяем установленную версию (см. рисунок 6).

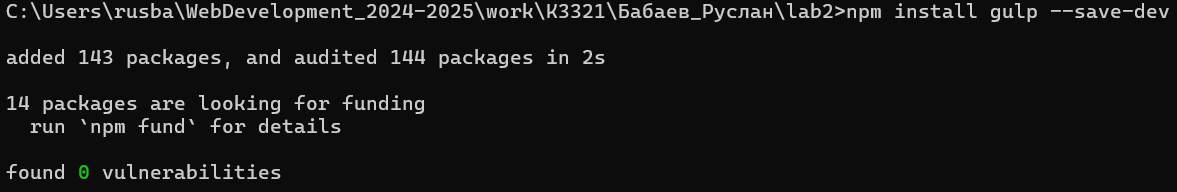


Рисунок 6 – Проверка установленной версии gulp

Затем создаём файл gulpfile.js в директории проекта, в котором создаём задачу (task), выводящую в консоль текст: "Hello, from Gulp!". Результат выполнения представлен на рисунке 7.

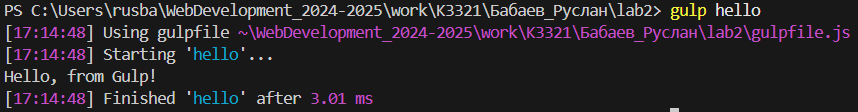


Рисунок 7 – Простая функция вывода приветствия в консоль в gulp

После этого был установлен browser-sync, и создана задача для отслеживания изменений в HTML-файле. Был создан файл track/index.html, и каждое изменение в этом файле отслеживалось и мгновенно синхронизировалось с отображением в браузере. Логи обновления файла страницы показаны на рисунке 8.

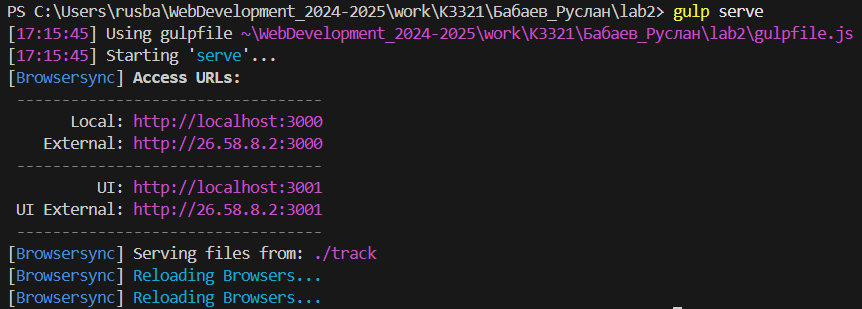


Рисунок 8 – Логи отслеживания изменений в HTML-файле

Полученная HTML-страница представлена на рисунке 9. Как можно заметить, здесь работает именно локальный веб-сервер, запущенный с помощью browser-sync.

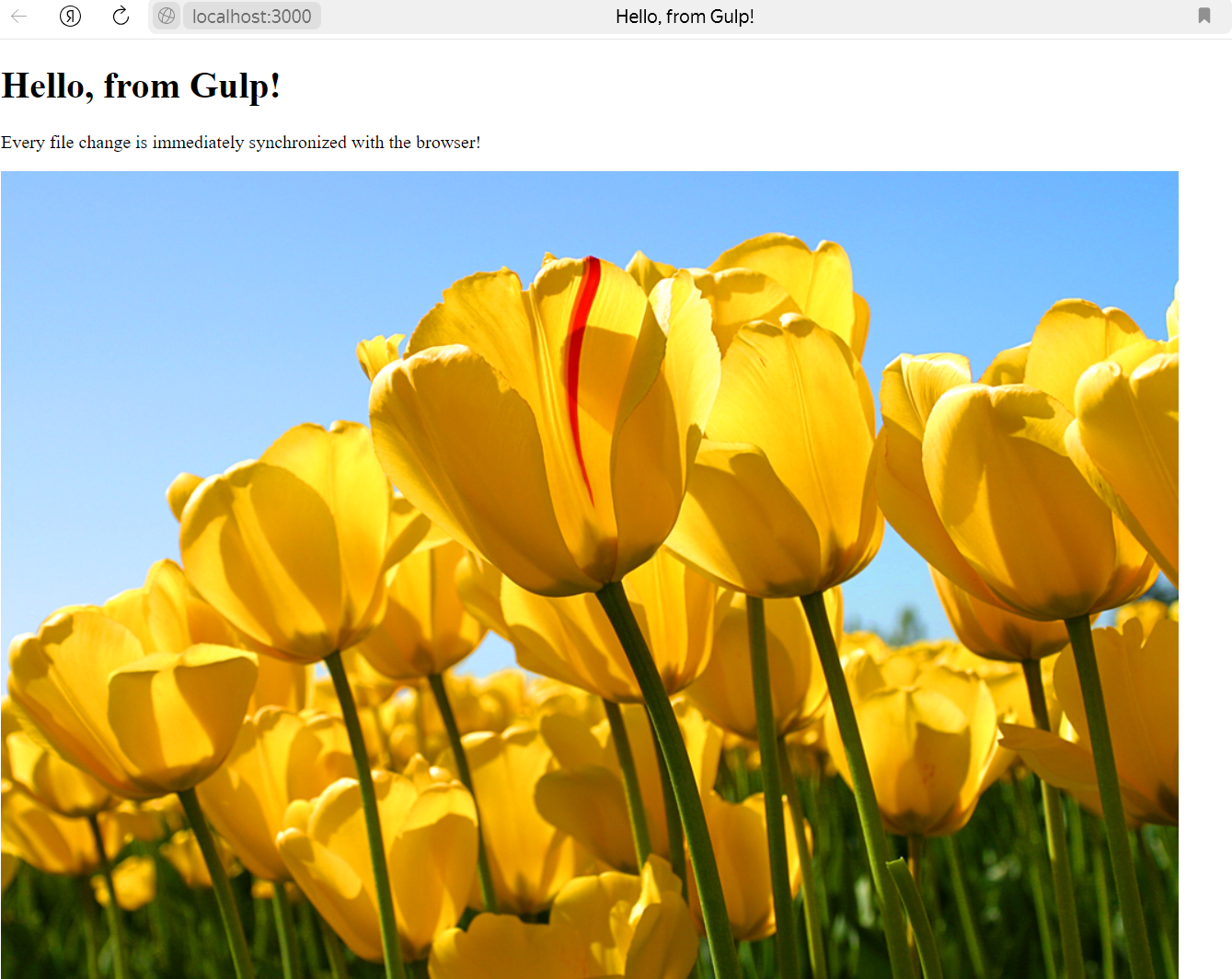


Рисунок 9 – Синхронизированное отображение HTML-страницы, полученное с помощью BrowserSync

## **Часть 3**

В данном коде реализованы функции для автоматизированного открытия веб-страниц с использованием пакета gulp-open. Основная функция openPagesWithDelay позволяет пользователю ввести список URL-адресов и указать задержку между их последовательным открытием. Функция запрашивает данные через консоль, после чего последовательно открывает страницы, используя openPage, и делает паузу между открытиями с помощью setTimeout.

Этот подход автоматизирует процесс открытия нескольких страниц с заданной задержкой, что может быть полезно для тестирования или демонстрации сайтов. Взаимодействие с пользователем происходит через консольный интерфейс, что делает программу гибкой и удобной для использования.

На рисунке 14 представлено взаимодействие с пользователем через терминал при запуске приложения, где вводятся URL-адреса и время ожидания перед открытием каждой страницы.

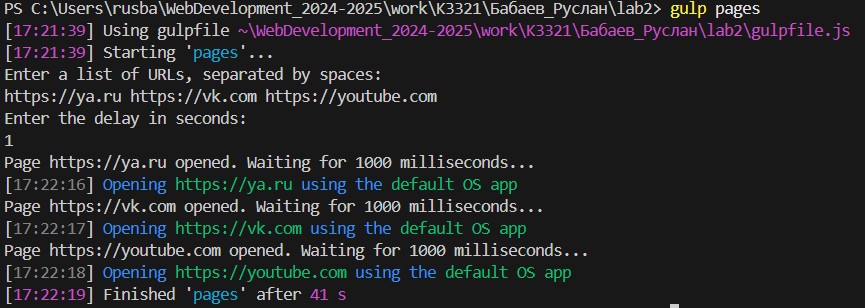


Рисунок 10 – Логи открытия страниц

# Заключение

В ходе данной лабораторной работы были изучены базовые принципы использования Gulp в JavaScript, а также закреплены навыки работы с проектным репозиторием на платформе GitHub. Сначала в резпозиторий было добавлено несколько тестовых коммитов с изменением HTML-файла (подробнее в lab2/git\_task), и разработано приложение с использованием Gulp. Оно включает функции для вывода приветственного сообщения в консоль, синхронизации изменений HTML-файлов с отображением в браузере через BrowserSync, а также функцию последовательного открытия веб-страниц с задержкой. Все поставленные цели и задачи работы были успешно выполнены.