

Университет ИТМО

Факультет «Инфокоммуникационных технологий»

Лабораторная работа №2  
«Использование Git и Gulp для решения задач web-разработки»

Выполнил:  
Кузнецов Никита Сергеевич  
Группа №К33212  
Проверил:  
Марченко Елена Вадимовна

Санкт-Петербург  
2023

## Цель работы:

С помощью утилиты Git научиться базовому использованию локальных репозиториев, созданием в них коммитов и синхронизации изменений с удаленным репозиторием. С помощью утилиты Gulp научиться создавать базовые задания на JavaScript и запускать их из командной строки. Разработать программу-клиент, которая показывает заданные веб-страницы из списка с определенным интервалом.

## Ход работы:

**1 задание:** установить Git на компьютер (локальный репозиторий), настроить на работу с проектом (можно использовать файлы предыдущей работы), выполнить изменения в файлах проекта. Для выполняемых изменений сделать коммиты (не менее трех). Проверить, что коммиты создаются. Локальный репозиторий синхронизировать с удаленным (можно использовать GitHub, GitLab, др.). Привести в отчете ссылку на проект.

Для решения данного задания была установлена программа Git на компьютер, затем для тестов была создана папка, в которой был создан пустой файл hello.txt (рис. 1). Далее, для инициализации локального репозитория была прописана команда `git init` и в консоли появился успешный результат её выполнения (рис. 2).

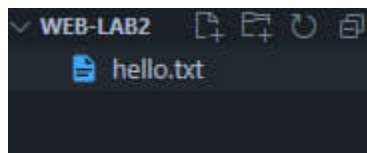


Рисунок 1 – Папка с пустым текстовым файлом hello.txt.



Рисунок 2 – Результат команды `git init`, показывающий создание пустого локального репозитория.

Затем, в файле была добавлена строка «hello!» (рис. 3). Для добавления этих изменений в Git и последующим их сохранении были прописаны следующие команды (рис. 4):

- `git add .` – для добавления всех изменений в Git;
- `git commit -m “first commit!”` – для коммита (сохранения) изменений с сообщением «first commit!».

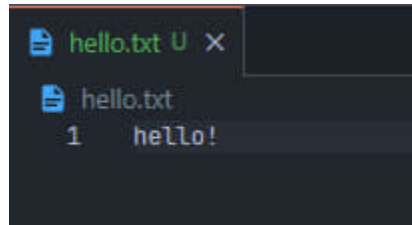


Рисунок 3 – Добавление первой строки в файл hello.txt.

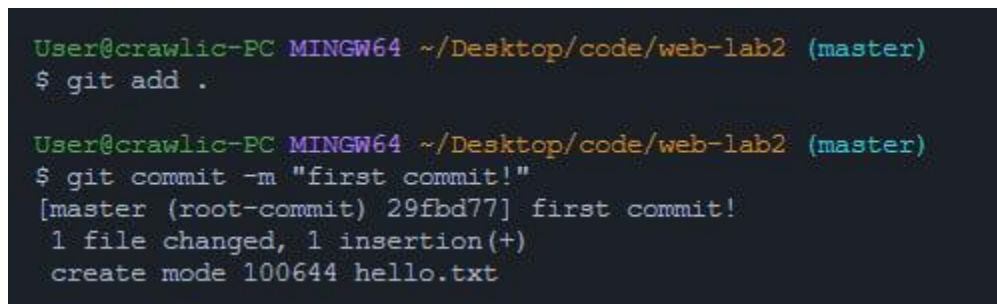


Рисунок 4 – Добавление и сохранение первого коммита в Git.

Далее, были проделаны аналогичные действия со вторым коммитом (рис. 5-6).

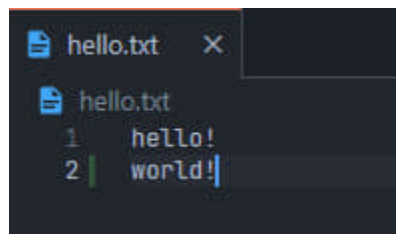


Рисунок 5 – Добавление второй строки в файл hello.txt.

```

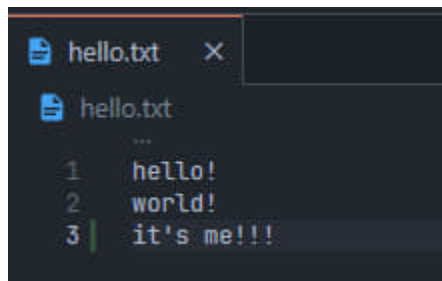
User@crawlic-PC MINGW64 ~/Desktop/code/web-lab2 (master)
$ git add .

User@crawlic-PC MINGW64 ~/Desktop/code/web-lab2 (master)
$ git commit -m "second commit!"
[master 1f4a84e] second commit!
1 file changed, 2 insertions(+), 1 deletion(-)

```

Рисунок 6 - Добавление и сохранение второго коммита в Git.

С третьим коммитом, была сделана ошибка в сообщении, поэтому был использован флаг `--amend`, позволяющий переписать предыдущий коммит (рис. 7-8).



```

hello.txt
...
1 hello!
2 world!
3 | it's me!!!

```

Рисунок 7 – Добавление третьей строки в файл hello.txt.

```

User@crawlic-PC MINGW64 ~/Desktop/code/web-lab2 (master)
$ git add .

User@crawlic-PC MINGW64 ~/Desktop/code/web-lab2 (master)
$ git commit -m "second commit!"
[master 7daee80] second commit!
1 file changed, 2 insertions(+), 1 deletion(-)

User@crawlic-PC MINGW64 ~/Desktop/code/web-lab2 (master)
$ git commit -m "third commit!" --amend
[master ff68ceb] third commit!
Date: Sat Oct 21 22:27:26 2023 +0300
1 file changed, 2 insertions(+), 1 deletion(-)

```

Рисунок 8 – Добавление и сохранение третьего коммита в Git с флагом `--amend`.

Далее, для проверки того, что коммиты были успешно созданы была запущена команда `git log`, показывающая историю всех коммитов (рис. 9). По результатам команды можно заметить, что все три коммита успешно сохранены в Git.

```
User@crawlic-PC MINGW64 ~/Desktop/code/web-lab2 (master)
$ git log
commit ff68ceb7a7a3df2e22e2790944940fef21484e89 (HEAD -> master)
Author: crawlic-stud <nikitosik0726@gmail.com>
Date: Sat Oct 21 22:27:26 2023 +0300

    third commit!

commit 1f4a84eb05020c68c23b082e6395b7f83be21d7c
Author: crawlic-stud <nikitosik0726@gmail.com>
Date: Sat Oct 21 22:27:05 2023 +0300

    second commit!

commit 29fbd77de4f116d159deefba4c50e84131e3b0d7
Author: crawlic-stud <nikitosik0726@gmail.com>
Date: Sat Oct 21 22:26:34 2023 +0300

    first commit!
```

Рисунок 9 – Вывод истории всех коммитов локального репозитория командой git log.

Далее, для того чтобы присоединить локальный репозиторий к удаленному, была изменена немного структура папки – файл `hello.txt` был перемещен в папку `lab2` (рис. 10). Это было сделано с целью сохранения структуры удаленного репозитория, в котором находятся остальные лабораторные работы, которые придерживаются такой же структуры.

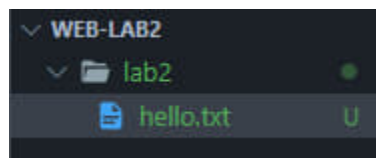


Рисунок 10 – Изменение структуры локального репозитория.

Далее, было произведено добавление удаленной ветки с помощью команды `git remote add origin https://github.com/crawlic-stud/web-programming-itmo.git` (рис. 11), где:

- `origin` – это название удаленного репозитория;
- `https://github.com/crawlic-stud/web-programming-itmo.git` — это ссылка на удаленный репозиторий на сайте Github.

```
User@crawlic-PC MINGW64 ~/Desktop/code/web-lab2 (master)
$ git remote add origin https://github.com/crawlic-stud/web-programming-itmo.git
```

Рисунок 11 – Добавление удаленного репозитория к локальному с помощью команды git remote add.

Так как в удаленном репозитории присутствуют изменения, которых нет в локальном, то была запущена команда git pull origin master, где master – это название ветки в удаленном репозитории. Эта команда выдала ошибку, которая была исправлена добавлением флага --allow-unrelated-histories в команду (рис. 12). Как можно заметить, команда была выполнена успешно и изменения были добавлены (рис. 13).

```
User@crawlic-PC MINGW64 ~/Desktop/code/web-lab2 (master)
$ git pull origin master
From https://github.com/crawlic-stud/web-programming-itmo
 * branch          master      -> FETCH_HEAD
fatal: refusing to merge unrelated histories

User@crawlic-PC MINGW64 ~/Desktop/code/web-lab2 (master)
$ git pull origin master --allow-unrelated-histories
From https://github.com/crawlic-stud/web-programming-itmo
 * branch          master      -> FETCH_HEAD
Merge made by the 'ort' strategy.
 lab1/lab1_Kuznetsov_K33212.pdf | Bin 0 -> 175717 bytes
 lab1/sample01.html             | 10 ++++
 lab1/sample02.html             | 20 +++++++
 lab1/sample03.html             | 23 ++++++++
 lab4/index.html                | 17 ++++++
 lab4/lab2_Kuznetsov_K33212.pdf | Bin 0 -> 297267 bytes
 lab4/scripts/database.db       | Bin 0 -> 12288 bytes
 lab4/scripts/login.php         | 86 ++++++
 lab4/scripts/save_data.php     | 101 ++++++
 lab4/static/order.html         | 42 ++++++
 lab4/style.css                 | 87 ++++++
11 files changed, 386 insertions(+)
create mode 100644 lab1/lab1_Kuznetsov_K33212.pdf
create mode 100644 lab1/sample01.html
create mode 100644 lab1/sample02.html
create mode 100644 lab1/sample03.html
create mode 100644 lab4/index.html
create mode 100644 lab4/lab2_Kuznetsov_K33212.pdf
create mode 100644 lab4/scripts/database.db
create mode 100644 lab4/scripts/login.php
create mode 100644 lab4/scripts/save_data.php
create mode 100644 lab4/static/order.html
create mode 100644 lab4/style.css
```

Рисунок 12 – Добавление изменений с удаленного репозитория с помощью команды git pull.



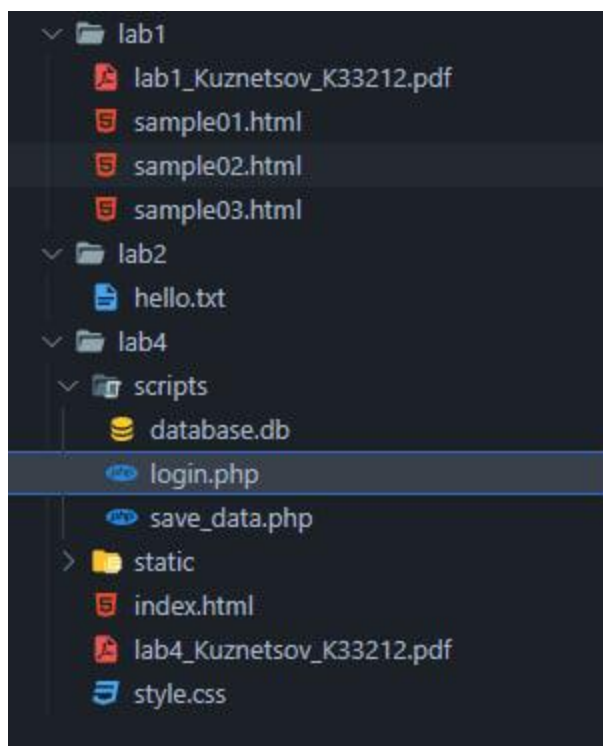


Рисунок 13 – Новая структура проекта, после соединения с удаленным репозиторием с остальными лабораторными работами.

В самом конце, для сохранения локальных изменений в удаленном репозитории, была выполнена команда `git push origin master` (рис. 14). Данная команда была выполнена успешно и локальные изменения оказались на сайте Github в публичном репозитории по ссылке: <https://github.com/crawlic-stud/web-programming-itmo> (рис. 15).

```
User@crawlic-PC MINGW64 ~/Desktop/code/web-lab2 (master)
$ git push origin master
Enumerating objects: 13, done.
Counting objects: 100% (13/13), done.
Delta compression using up to 6 threads
Compressing objects: 100% (5/5), done.
Writing objects: 100% (12/12), 1.03 KiB | 527.00 KiB/s, done.
Total 12 (delta 0), reused 0 (delta 0), pack-reused 0
To https://github.com/crawlic-stud/web-programming-itmo.git
075fc13..c3efb8b master -> master
```

Рисунок 14 – Добавление локальных изменений в удаленный репозиторий на Github.

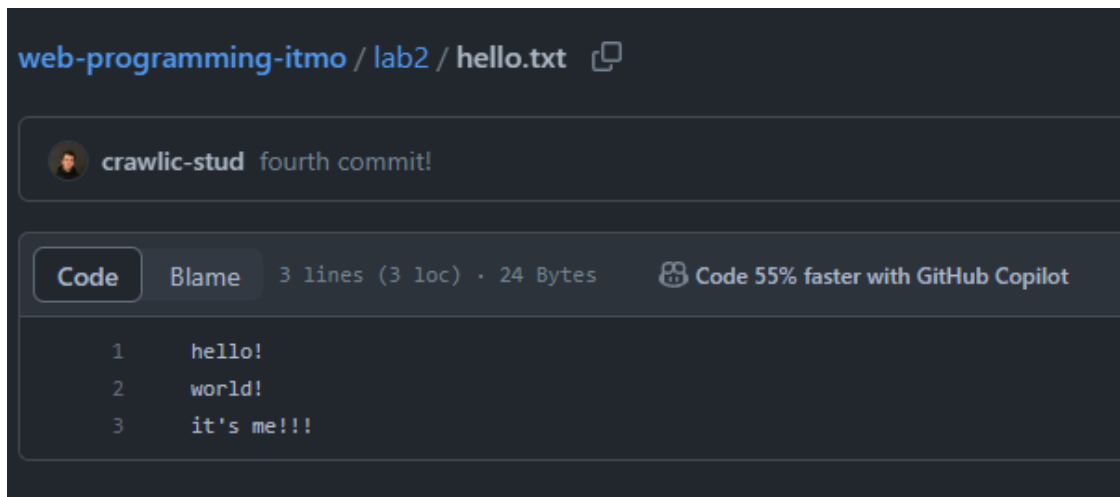


Рисунок 15 – Сохраненные изменения на сайте Github.

**2 задание:** установить gulp. Проверить процесс установки, отметить основные этапы. Создать task (можно всего 1 и можно простой, в более сложном варианте создать Task для работы с BrowserSync <https://browsersync.io/>).

Перед началом работы необходимо установить на компьютер программу node.js вместе с установщиком npm. Они необходимы для запуска библиотеки Gulp, которая использует JavaScript.

После установки, необходимо установить саму библиотеку Gulp, это делается с помощью двух команд:

- `npm install --save-dev gulp` – установка самой библиотеки для JavaScript;
- `npm install --global gulp-cli` – установки утилиты для запуска Gulp из командной строки.

После завершения установки, можно проверить работоспособность Gulp, прописав в командной строке `gulp --version` (рис. 16). Как можно заметить, оба пакета библиотеки успешно установились.

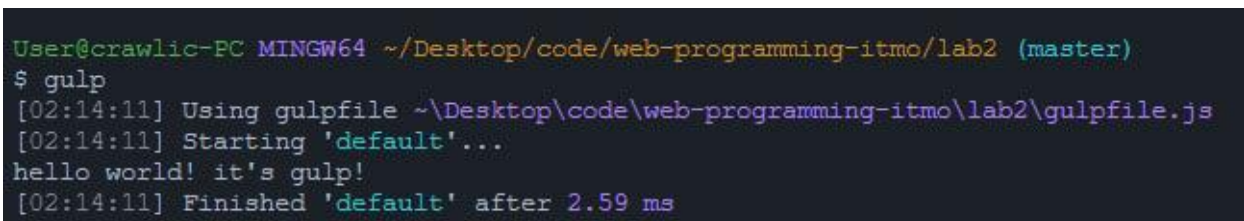


Рисунок 16 – Проверка установки утилиты Gulp.



Для создания базовых задач был создан файл `gulpfile.js` в корневой директории (полный код файла можно посмотреть в приложении 1). Для проверки работоспособности, был скопирован пример из документации Gulp, в котором создается стандартный task, который будет запускаться если в командную строку не передать никаких других параметров. С его кодом можно ознакомиться в приложении 1.

Далее, этот код был запущен с помощью команды `gulp` и в консоли появилось сообщение «hello world! it's gulp!» (рис. 17), которое было объявлено в функции в коде.



```
User@crawlic-PC MINGW64 ~/Desktop/code/web-programming-itmo/lab2 (master)
$ gulp
[02:14:11] Using gulpfile ~\Desktop\code\web-programming-itmo\lab2\gulpfile.js
[02:14:11] Starting 'default'...
hello world! it's gulp!
[02:14:11] Finished 'default' after 2.59 ms
```

Рисунок 17 – Вывод команды `gulp` для стандартного taska `defaultTask`.

Для создания более сложного taska с аргументами и названием была импортирована библиотека `gulp` для JavaScript. Затем, с ее помощью была создана функция `count`, которая принимает аргумент `--num` из консоли и выводит числа от 1 до `N`, где `N` —это числовой аргумент после `--num`. С кодом для этого taska можно ознакомиться в приложении 1.

Теперь, запустив в консоли команду `gulp count --num 5` можно увидеть вывод чисел от 1 до 5 (рис. 18). Также, если запустить эту команду без аргументов, то можно увидеть вывод “Nothing to count” (рис. 19), означающий что нужный аргумент не был передан.



```
User@crawlic-PC MINGW64 ~/Desktop/code/web-programming-itmo/lab2 (master)
$ gulp count --num 5
[02:14:29] Using gulpfile ~\Desktop\code\web-programming-itmo\lab2\gulpfile.js
[02:14:29] Starting 'count'...
1
2
3
4
5
[02:14:29] Finished 'count' after 4.69 ms
```

Рисунок 18 – Вывод команды `gulp count` с аргументом `--num 5`.

```
User@crawlic-PC MINGW64 ~/Desktop/code/web-programming-itmo/lab2 (master)
$ gulp count
[02:14:39] Using gulpfile ~\Desktop\code\web-programming-itmo\lab2\gulpfile.js
[02:14:39] Starting 'count'...
Nothing to count
[02:14:39] Finished 'count' after 2.4 ms
```

Рисунок 19 – Вывод команды gulp count без аргументов.

**3 задание:** рассмотреть программу клиент для показа web-страниц (<https://moodle.itmo.ru/mod/resource/view.php?id=7020>). Написать программу клиент, которая показывает web-страницы одна за другой из списка (в программе можно задавать адреса страниц и интервал показа).

Для написания данного задания я решил использовать HTML, CSS и JavaScript. Код для каждого из файлов представлен в приложениях 2, 3 и 4 соответственно.

В начале, была создана форма в файле index.html (приложение 2), состоящая из трех полей для ссылок (URL), поля для интервала показа в секундах, кнопки запуска слайд-шоу и тэга <iframe>, позволяющего показывать другие веб-страницы внутри текущей HTML-страницы.

Затем был создан файл style.css (приложение 3). В нем были определены стили для перечисленных элементов HTML-страницы.

После стилей был создан файл script.js (приложение 4), в котором была определена логика появления страниц и смена их с заданным интервалом. Это было достигнуто с помощью встроенной функции setTimeout, которая задерживала исполнение функции на определенное количество времени, тем самым позволяя, выводить ссылки так, как было нужно по заданию.

Для запуска приложения был выбран PHP-сервер. Чтобы запустить его необходимо было в корневой директории выполнить команду php -S 127.0.0.1:8000, где 127.0.0.1 – это адрес локального хоста, 8000 – это порт, на котором будет располагаться сервер. Сервер был успешно запущен и на него начали поступать запросы (рис. 20) с клиента. Для демонстрации готового приложения были добавлены ссылки на my.itmo (рис. 21), на Moodle курса WEB-программирования (рис. 22) и на документацию библиотеки Gulp (рис.

23). Как можно заметить, каждая страница была успешно показана на веб-странице.

```
User@crawlic-PC MINGW64 ~/Desktop/code/web-programming-itmo/lab2/task3 (master)
$ php -S 127.0.0.1:8000
[Sun Oct 22 15:50:58 2023] PHP 8.2.11 Development Server (http://127.0.0.1:8000) started
[Sun Oct 22 15:50:59 2023] 127.0.0.1:59670 Accepted
[Sun Oct 22 15:50:59 2023] 127.0.0.1:59670 [200]: GET /
[Sun Oct 22 15:50:59 2023] 127.0.0.1:59670 Closing
[Sun Oct 22 15:50:59 2023] 127.0.0.1:59671 Accepted
[Sun Oct 22 15:50:59 2023] 127.0.0.1:59672 Accepted
[Sun Oct 22 15:50:59 2023] 127.0.0.1:59671 [200]: GET /script.js
[Sun Oct 22 15:50:59 2023] 127.0.0.1:59672 [200]: GET /style.css
```

Рисунок 20 – Успешный запуск сервера на PHP.

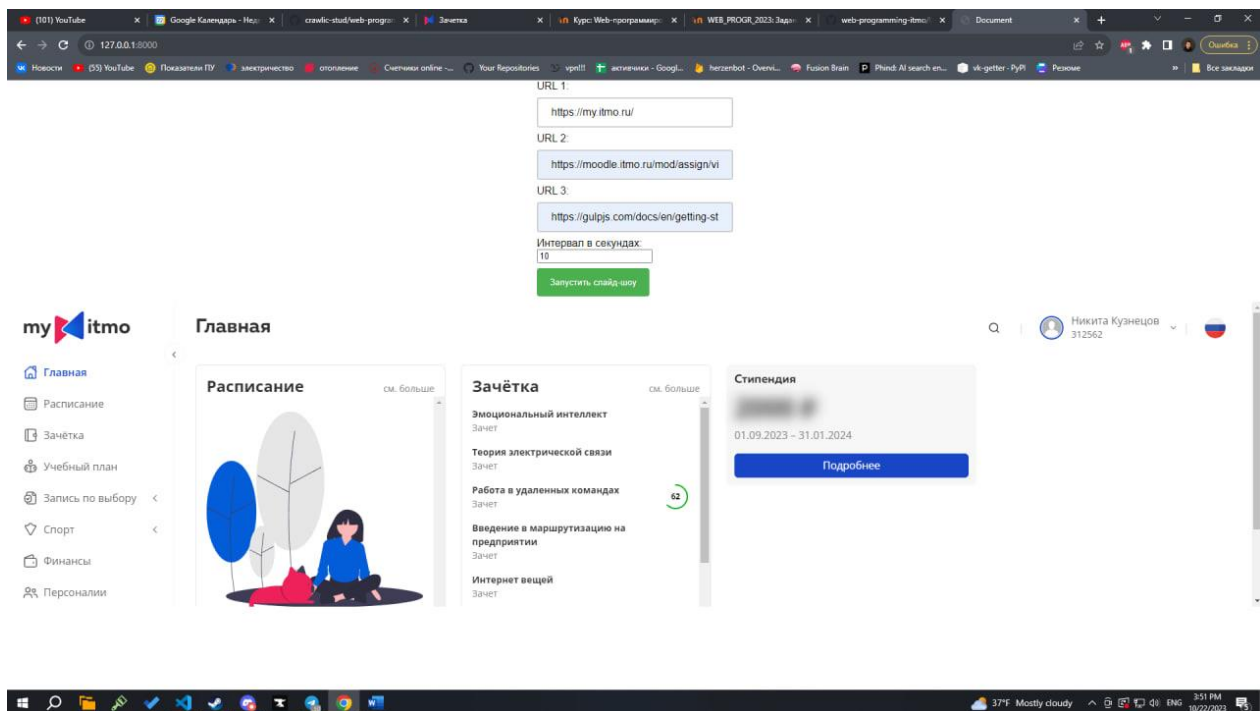


Рисунок 21 – Показ страницы my.itmo внутри веб-приложения.

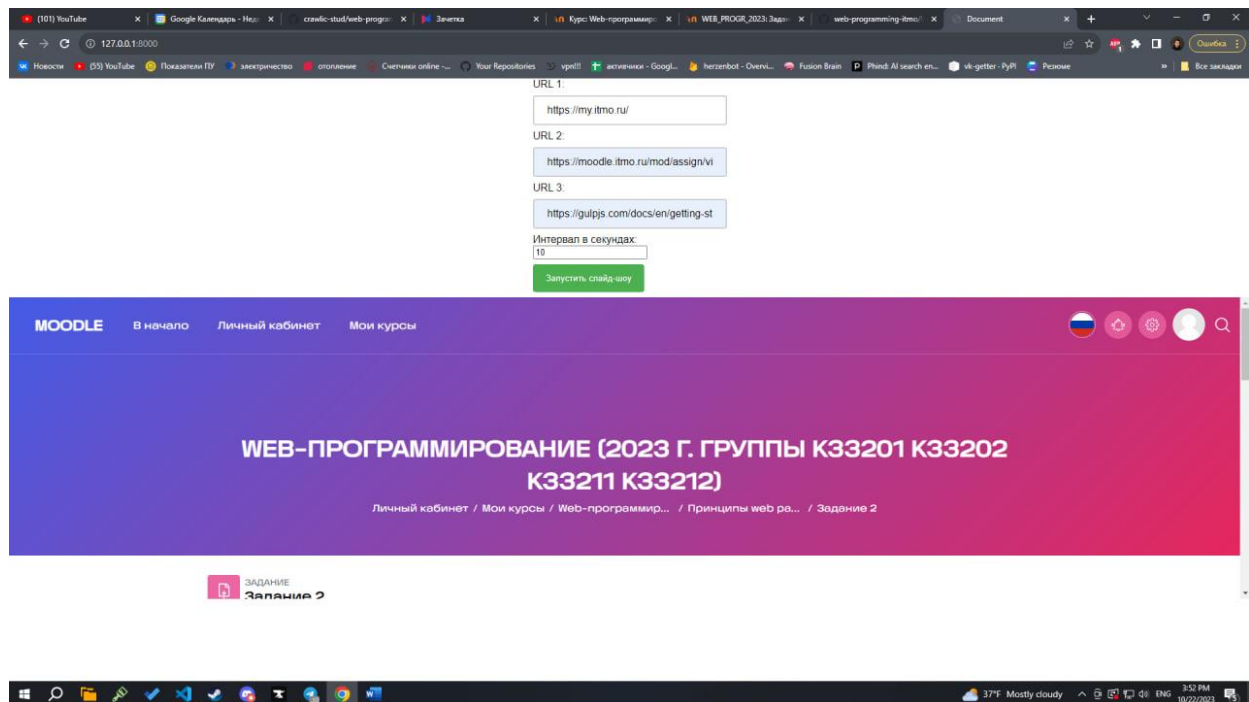


Рисунок 22 – Показ страницы Moodle внутри веб-приложения.

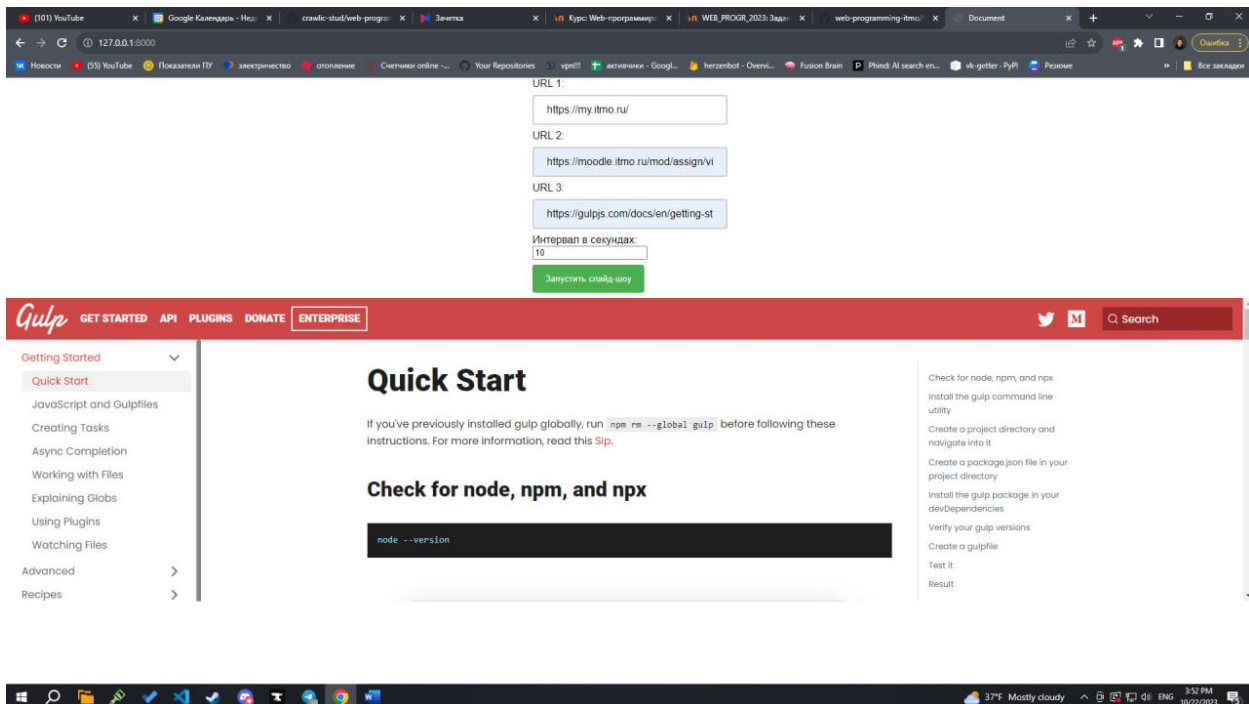


Рисунок 23 – Показ страницы документации Gulp внутри веб-приложения.

## Приложение 1. JavaScript файл gulpfile.js с кодом для библиотеки Gulp.

```
var gulp = require('gulp');

// функция базового таска для gulp
function defaultTask(callback) {
  console.log("hello world! it's gulp!");
  callback();
}

gulp.task("count", function(callback) {
  // пятый аргумент это будет аргумент после --num
  var num = process.argv[4] ? process.argv[4] : 0;
  if (num <= 0) {
    console.log("Nothing to count");
  } else {
    for (var i = 1; i <= num; i++) console.log(i);
  }
  callback();
})

// экспортируем таск как стандартный для gulp
exports.default = defaultTask
```

## Приложение 2. HTML файл index.html с кодом для веб-страницы с формой для показывания слайд-шоу из URL.

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta http-equiv="X-UA-Compatible" content="IE=edge">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <link href="style.css" rel="stylesheet">
  <title>Document</title>
</head>
<body>
  <div class="form">
    URL 1: <input type="text" name="url1"><br>
    URL 2: <input type="text" name="url2"><br>
    URL 3: <input type="text" name="url3"><br>
    Интервал в секундах: <input type="number" id="interval"><br>
    <button onclick="slideshow()">Запустить слайд-шоу</button>
  </div>
  <iframe id="frame"></iframe>
  <script src="script.js"></script>
</body>
</html>
```



### Приложение 3. CSS файл style.css с кодом стилей для веб-страницы index.html.

```
body, html {
    margin: 0;
    padding: 0;
    height: 100%;
    overflow: hidden;
    font-family: Arial, Helvetica, sans-serif;
}

iframe {
    width: 100%;
    height: 50vh;
    border: black;
    border-width: 2px;
    text-align: center;
    position: relative;
    align-items: center;
}

.form {
    width: 300px;
    margin: 0 auto;
}

input[type=text] {
    width: 100%;
    padding: 12px 20px;
    margin: 8px 0;
    box-sizing: border-box;
    border: 2px solid #ccc;
    border-radius: 4px;
    font-size: 16px;
}

button {
    background-color: #4CAF50;
    color: white;
    padding: 14px 20px;
    margin: 8px 0;
    border: none;
    border-radius: 4px;
    cursor: pointer;
}

button:hover {
    background-color: #45a049;
}
```

#### Приложение 4. JavaScript файл script.js с кодом для логики изменения страниц в iframe для веб-страницы.

```
function getUrls() {
    var inputs = document.querySelectorAll('input[name^="url"]');
    var urls = [];
    for (let i of inputs) {
        urls.push(i.value);
    }
    return urls;
}

function showUrl(url) {
    var frame = document.getElementById("frame");
    frame.hidden = false;
    frame.src = url;
}

function slideshow() {
    var urls = getUrls();
    var interval = parseInt(document.getElementById("interval").value);
    function showNextUrl() {
        if (urls.length > 0) {
            showUrl(urls.shift());
            setTimeout(showNextUrl, interval * 1000);
        }
    }
    showNextUrl();
}
```