КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ІМЕНІ ТАРАСА ШЕВЧЕНКА

Факультет радіофізики, електроніки та комп'ютерних систем

Кафедра комп'ютерної інженерії

АВТОМАТИЗАЦІЯ СИСТЕМ ГЕНЕРАЦІЇ WEB-САЙТУ

Кваліфікаційна робота бакалавра
студента 4 року навчання
спеціальність: 123 «Комп'ютерна інженерія»
Дмитра КОВАЛЬОВА

Науковий керівник:
Юрій ЮРЧИК,
асистент кафедри комп'ютерної інженерії

Рецензент::
канд. фіз.-мат. наук Олександр СУДАКОВ,
доцент кафедри медичної радіофізики

До захисту допускаю:
Завідувач кафедрою Юрій БОЙКО

Ухвалено на засіданні кафедри "16" червня 2022 р., протокол № 18

Київ - 2022

РЕФЕРАТ

Обсяг роботи 36 сторінки, 16 рисунків, 2 таблиці, 3 лістинги.

СТАТИЧНИЙ САЙТ, СТАТИЧНА ГЕНЕРАЦІЯ, СТАТИЧНИЙ ГЕНЕРАТОР NIKOLA, GITLAB, CI/CD, RESTRUCTUREDTEXT, DOCKER.

Об’єктом роботи є вебсайт кафедри комп’ютерної інженерії.

Метою роботи є створення вебсайту кафедри комп’ютерної інженерії з використанням засобів статичної генерації та його розгортання.

Методи розроблення: проектування вебсайту, автоматизація розгортання сайту засобами CI/CD.

Інструменти розробки: генератор статичних вебсайтів Nikola, система контейнерів Docker, автоматизовані засоби інтеграції і розгортання GitLab CI/CD

Результат роботи: створено вебсайт кафедри комп’ютерної інженерії з використанням статичного генератору сайтів Nikola, проведено реорганізацію контенту сайту. Автоматизація розгортання сайту реалізована засобами CI/CD за допомоги GitLab Pages. Сайт введений в експлуатацію, і доступний за посиланням http://ce.knu.ua/.

ЗМІСТ

ВСТУП

Оцінка сучасного стану об’єкта розробки. Наявний сайт кафедри комп’ютерної інженерії написано з використанням мови PHP, з залученням системи менеджменту контенту Joomla.

Актуальність роботи та підстави для її виконання. Joomla – це система керування вмістом (CMS), тобто програмне забезпечення для організації роботи вебсайтів. Це означає, що вміст сайту частково зберігається в базі даних, і сторінки сайту генеруються при кожному зверненні користувачів за допомоги рушію PHP. Це доволі повільно, та потребує певних ресурсів для постійної генерації сторінок. Крім того PHP поступово знижується в популярності серед веброзробників, тому є сенс перейти на більш сучасну та просту платформу. Також вміст сайту змінюватиметься доволі рідко, тому використання статичної генерації ніяк не ускладнить чи уповільнить подальшу його розробку.

Мета й завдання роботи. Реалізувати функціонал сайту кафедри комп’ютерної інженерії з використанням засобів статичної генерації. Це спростить процес подальшої розробки, зменшить використання обчислювальних та програмних ресурсів, необхідних для забезпечення роботи сайту. Провести реорганізацію контенту сайту. Автоматизувати процес розгортання та інтеграції сайту.

Об'єкт, методи й засоби розроблення. Об’єктом цієї роботи є створення вебсайту з використанням засобів статичної генерації. Для розробки було обрано інструмент Nikola, розгортання сайту реалізовано за допомогою засобів GitLab CI/CD.

1 СТАТИЧНА ГЕНЕРАЦІЯ WEB-ЗАСТОСУНКІВ

1.1 Статичні вебсайти

Статичні вебсайти складаються з певної кількості попередньо сформованих сторінок, які містять сталий вміст і структуру.

Веброзробники зазвичай використовують мову розмітки HTML для задання структури контенту, та CSS для задання параметрів візуального відображення елементів. На відміну від CMS, статичні сайти характеризуються відносно малими вимогами до апаратних та програмних ресурсів.

Після того як статичний вебсайт опубліковано, він залишається незмінним і не потребує повторної генерації вебсторінок у відповідь на запити користувача. Щоб внести зміни потрібно оновити HTML файли, часто для кожної сторінки вебсайту, що може бути дуже трудомістким, якщо потрібно оновити великий вебсайт.

Переваги статичних вебсайтів:

- статичні вебсайти потребують менше часу на збирання та розгортання ніж динамічні;
- можна копіювати базовий код між статичними сторінками, щоб зберігати одноманітність елементів, але додаючи певні корективи для їх виділення;
- статичні вебсайти зазвичай більш захищені;
- пошуковим системам легше ранжувати статичні сайти бо вони зазвичай швидше завантажуються;

- створення статичних сайтів зазвичай не потребує складного програмного стеку, як в динамічних;
- статичні вебсайти дешевші в розробці;
- статичний вебсайт просто відновити в разі програмного збою, або цілеспрямованої атаки на вебсервер.

Проте використання статичних, наперед згенерованих вебсторінок, має свої недоліки. До основних недоліків можливо віднести наступні:

- впровадження оновлення коненту сайту може бути складним і довгим, особливо якщо статичний сайт має великий обсяг;
- після створення базової структури, розширення та модифікація функціоналу сайту потербує додаткових зусиль;
- ускладнене застосування для відображення контенту, що має високу динаміку змін.

1.2 Динамічні вебсайти

Динамічні вебсайти генерують сторінки в реальному часі, за запитом корисувача. Гнучкість контенту і структури дозволяє динамічно оновлювати вміст сторінки, яку бачить користувач, або браузеру, який він використовує. Створення динамічного вебсайту зазвичай потребує знань відповідних мов програмування, як-от PHP, C# або Python. Нерідко промислове використання динамічних вебсайтів передбачає наявність систем керування контентом (CMS) та необхідність використання баз даних.

Програмний код, розміщений на сервері, використовується для генерувати HTML сторінки в реальному часі, які сконструйовані спеціально щоб задовольняти потреби певного користувача. В той час як статичні вебсайти відіграють роль інформування, динамічні вебсайти передбачають

взаємодію з користувачем і містять часто змінювані елементи. Як результат, веброзробники часто використовують комбінацію серверних і клієнтських програм, щоб створити дійсно інтерактивний вебсайт для відвідувачів.

Динамічні вебсайти генерують і показують контент в залежності від дій користувача. Рівень цих змін залежить від здібностей розробників і того, наскільки складними вони зроблять інтерактивні елементи сайту.

Переваги динамічних вебсайтів:

- надають сайту більше функціоналу і забезпечують динамічне оновлення контенту;
- показують різний контент в залежності від потреб користувача;
- забезпечують більшу гнучкість вебсайту завдяки можливості підключення системи керування вмістом, що спрощує оновлення контенту.

У якості основних недоліків динамічних вебсайтів можна зазначити:

- такі аспекти розробки, як створення основи сайту, створення підключень до бази даних та додавання інших функцій, можуть зробити динамічний веб-сайт дорожчим, ніж статичний;
- сайт може працювати повільніше та бути дорожчим в експлуатації через те, що сторінки потрібно кожен раз генерувати при запиті користувача.

1.3 Основні відмінності між статичними і динамічними сайтами

Основною різницею є те, що контент на статичному сайті залишається однаковим допоки розробник не внесе зміни у вихідний код, а інформація на динамічному сайті може динамічно змінюватись в залежності від потреб користувачів.

Одною з вад динамічних сайтів є їх порівняно мала швидкодія та висока вартість експлуатації. Як вже було сказано, динамічні сайти при кожному запиті користувачів можуть звертатися до баз даних для отримання відповідної інформації, відповідно для комфортної взаємодії користувачів з сайтом потрібне досить потужне обладнання, інакше сайт працюватиме повільно. Статичні ж сайти фактично не мають такої проблеми, адже все згенеровано завчасно, і вебсерверу просто потрібно зчитати файл з диску, та надіслати користувачу (рис. 1.3.1), а якщо це популярний файл то з дискового кешу.

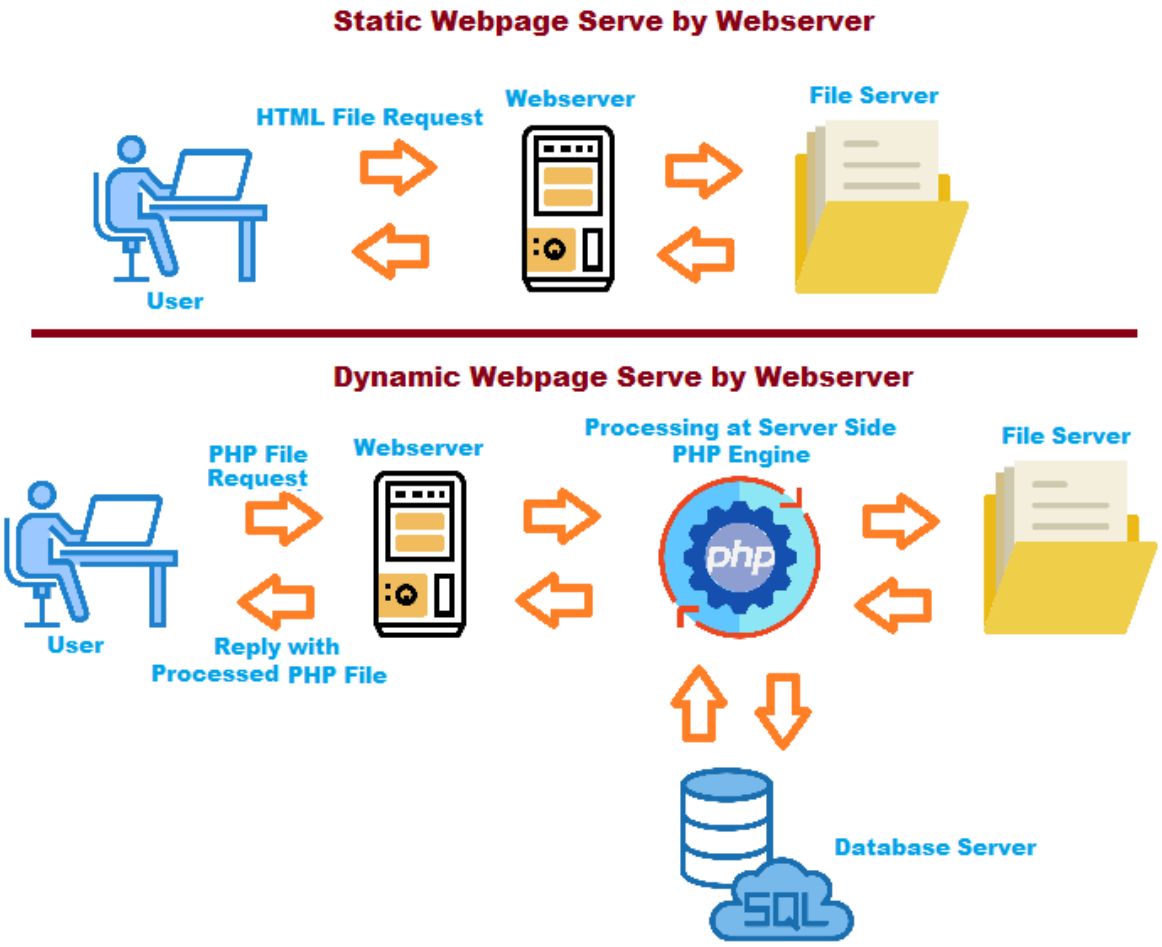

Рис. 1.3.1 – Типові схеми роботи статичних та динамічних сайтів при зверненні користувача

Програмний стек статичних сайтів набагато простіший – вебсервер, за оновленням якого простіше слідкувати. Сайт по суті являє собою звичайні HTML сторінки, тому розгортати такі сайти досить просто: не потрібна додаткова інсталяція середовищ, таких як PHP чи ASP.NET.

Статичний контент завантажується разом із самою вебсторінкою. Те що користувач бачить презентуєтсья в однаковому форматі не залежно від дій користувача. В динамічних вебсайтах контент змінюється в залежності від елементів керування встановлених адміністратором сайту та дій користувача.

Інформацію з статичних вебсайтів простіше кешувати. Зробити подібне для динамічного вебсайту складніше. Статичний контент може зберігатись на крайових серверах мережі доправлення контенту (CDN), що забезпечує вищу доступність і швидкість отримання інформації.

Кешування контенту пришвидшує завантаження сайту для користувачів, бо крайові сервери розподілені в різні географічні місця. Як результат, мережа доправлення контенту забезпечує швидше і надійніше підключення користувачам, які знаходяться поруч. Але зробити подібне для динамічного вебсайту менш практично через частоту зміни контенту, і крім того є небезпека пов’язана із кешуванням приватної інформації користувачів.

1.4 Область застосування статичних та динамічних вебсайтів

Статичні вебсайти найбільше підходять у випадках, коли на сайті не багато сторінок. Ці сайти зазвичай використовуютья для опублікування певної інформації. В динамічних вебсайтах інформація зазвичай приавтна – корситувачі входять під своїм обліковим записом і отримують доступ до неї, тому такі сайти зазвичай використовуються у різних приватних компаніях та підприємствах.

Якщо завдання – створити простий охайний блог, який доповнюватиме підприємство, то статичний вебсайт найкраще підійде для такої цілі. Інший випадок використання статичного сайту – це створення цільової сторінки, яка міститиме базову інформацію про компанію, продукти і сервіси, які надаються.

Якщо в планах створення інтернет-магазину з часто змінним інвентарем, то краще використовувати динамічний вебсайт. Це також дозволить робити рекомендації користувачам виходячи із їх попередніх покупок на сайті. Якщо користувач залишить якісь предмети в кошику, то сайт міг би надсилати сповіщення користувачу з нагадуванням завершити їх покупку. Динамічні вебсайти також можуть використовуватися при створенні прогресвних web-застосунків.

Рішення приймається в залежності від потреб: якщо сайт простий і бажано швидко його запустити – краще використовувати статичний; створення динамічного сайту - більш складний і довгий процес, але надає багато гнучкості для адаптації сайту під різні потреби. [13]

1.5 Статична генерація вебсайтів

Статична генерація дозволяє уникнути деякі вади класичних статичних сайтів, при цьому зберігаючи усі переваги в простоті і швидкості роботи вебсайту. Статичний генератор – це інструмент, який генерує HTML сторінки вебсайту і його структуру виходячи з заданих даних та шаблонів. Задаються вони часто у вигляді файлів мовами розмітки такою як Markdown або reStructuredText, або більш складними мовами розробки інтерфейсу користувача, як на приклад JavaScript. Фактично, статичний генератор сайтів автоматизує завдання розмітки окремих HTML сторінок, і наперед готує їх для

надсилання користувачам. Так як ці сторінки наперед згенеровані, вони дуже швидко завантажуються у браузерах користувачів.

Статичні генератори сайтів є альтернативою системам керування вмістом (CMS). CMS – інший вид інструменту для управління веб контентом, генерування веб-сторінок, та імплементації шаблонів. CMS прийшли на заміну найпершим статичним сайтам, де розробники мали вручну описувати кожну сторінку та розміщувати їх у відповідних директоріях. CMS прибирає необхідність ручної роботи, та дозволяє зберігати контент сторінок в базі даних, і коли користувач запитує сторінку, то вона генерується виходячи із даних в базі даних та заданого шаблону. При цьому є обмеження того, що контент має підходити під тип полів, які надаються базою даних CMS.

Статичні генератори сайтів пропонують компроміс між цими двома підходами. Як і CMS, вони дозволяють розробникам використовувати шаблони та автоматично генерувати сторінки, але вони це роблять наперед, а не у відповідь на запит користувача. [16]

1.6 Приклади статичних генераторів

Hugo – статичний генератор сайтів написаний мовою програмування Go компанії Google. Він використовує шаблони для компонування сторінок. Основною перевагою, яка відрізняє Hugo від інших генераторів – це швидкість генерації сторінок. Він надає хороші можливості керування вмістом, доступний для багатьох операційних систем. Підтримує таксономію, надає таблиці генерації контенту, підтримує чисті адреси, і може бути розгорнутим практично усюди. На додачу підтримує живу компіляцію під час розробки підтримує різні види контенту, має функціонал підрахування кількості слів і часу на прочитання.

Next.js – фреймворк, побудований на базі Node.js, який дозволяє вести розробку сайтів засобами React. При цьому на відміну від більшості подібних фреймворків, він дозволяє реалізовувати візуалізацію сторінок не тільки з боку клієнта, а й збоку сервера, і підтримує статичну генерацію. Такий підхід дозволяє уникнути деякі проблеми пов'язані з використанням React: довший час завантаження сторінки, потреба наявності у користувача доступу до виконання JavaScript у браузері, потенційні проблеми з безпекою. Все це негативно впливає на оптимізацію сайту для пошукових систем.

Jekyll написаний на Ruby, і зазвичай використовується для створення блогів і персональних проектів, через його близьку інтеграцію з GitHub. Jekyll використовує Markdown, Liquid, HTML та CSS як вхідні файли та генерує готовий для розгортання сайт. Цей генератор має велику спільноту розробників, які створили велику кількість плагінів, що полегшує роботу з ним для людей з досвідом роботи з WordPress та Drupal, дозволяючи імпортувати контент з цих форматів. [1]

Серед інших генераторів статичних сайтів для розробки сайту кафедри комп'ютерної інженерії було обрано саме Nikola, бо він забезпечує усіма потрібними інструментами і можливостями, при цьому залишаючись доволі простим як у використанні, так і у вивченні, завдяки своїй нескладній структурі. При цьому підтримка великої кількості вхідних форматів дозволяє розробникам застосовувати їх досвід роботи в інших сферах при подальшій підтримці сайту. Крім цього в ІОЦ вже є приклади вдалого використання саме цього генератору.

2 СТВОРЕННЯ САЙТУ СТАТИЧНИМ ГЕНЕРАТОРОМ NIKOLA

2.1 Загальна інформація

Nikola – статичний генератор сайтів написаний на Python 3 (3.5+). Він підтримує багато форматів вхідних файлів: reStructuredText, Markdown, IPython (Jupyter) Notebook, HTML а також має плагіни для багатьох інших форматів.

Nikola використовує утиліту автоматизації doit, що забезпечує інкрементну рекомпіляцію, через що пришвидшується розробка та редагування сайту, бо кожен раз компілюється тільки те, що змінилось, а не все. Крім того компіляція може відбуватись автоматично, після кожної зміни, що дуже корисно при активній розробці.

Наявна велика кількість плагінів, які розширюють функціонал Nikola, їх каталог можна переглянути на сайті генератору, а також за потреби можна написати свій плагін. Nikola має досить простий і зручний інтерфейс командного рядка, і створення сторінок та статей значно спрощено, бо метадані автоматично заповнюються, і не потрібно пам’ятати їх заголовки. Наявно багато вбудованого функціоналу: сторінки, статті з можливістю додати коментарі від третіх сторін (на приклад DISQUS, IntenseDebate, Facebook), прості галереї зображень. Підтримується майже 40 мов (в тому числі українська), з можливістю додавати більше. Для стилізації сайту використовується bootstrap4 з можливостями налаштувати власну тему. [9]

2.2 Робота з Nikola

Найважливіші та часто вживані команди nikola, необхідні для розробки подано в наступній таблиці:

Команда	Опис
`nikola init <mysite>`	Ініціює пустий проект у директорії mysite
`nikola new_post`	Створює нову статтю
`nikola new_page`	Створює нову сторінку
`nikola build`	Генерує сайт в директорії output
`nikola serve`	Запускає локальний тестовий сервер з сайтом
`nikola auto`	Автоматично перегенеровує сайт, коли детектує зміни у вихідних файлах

Для комфортної розробки з використанням Nikola спочатку потрібно встановити та активувати віртуальне середовище, для цього є зручний інструмент venv, який іде нативно з Python (рис. 2.2.1)

```
PS D:\Files\University\4th year\Diploma\nikola_test> python -m venv env
PS D:\Files\University\4th year\Diploma\nikola_test> .\env\Scripts\activate
```

Рис. 2.2.1 – Створення і активація віртуального середовища за допомоги інструменту venv

Після налаштування середовища потрібно встановити в нього Nikola, для цього використовується інсталятор пакетів pip (рис. 2.2.2)

```
(env) PS D:\Files\University\4th year\Diploma\nikola_test> python -m pip install -U pip setuptools wheel
(env) PS D:\Files\University\4th year\Diploma\nikola_test> python -m pip install -U "Nikola[extras]"
```

Рис. 2.2.2 – Встановлення Nikola за допомоги інсталятору пакетів pip

Тепер можна створити чистий проект командою nikola init (рис. 2.2.3). Запуститься майстер, який дозволить задати деякі базові налаштування сайту, такі як назва сайту, доменне ім’я сайту, інформація про автора, налаштування мови сайту, часової зони.

```
(env) PS D:\Files\University\4th year\Diploma\nikola_test> nikola init test
Creating Nikola Site
====================

This is Nikola v8.1.3.  We will now ask you a few easy questions about your new site.
If you do not want to answer and want to go with the defaults instead, simply restart with the `-q` parameter.
--- Questions about the site ---
Site title [My Nikola Site]: Test
Site author [Nikola Tesla]: Kovalov Dmytro
Site author's e-mail [n.tesla@example.com]: wresser1grr.la@gmail.com
Site description [This is a demo site for Nikola.]:
Site URL [https://example.com/]:
Enable pretty URLs (/page/ instead of /page.html) that don't need web server configuration? [Y/n] Y
--- Questions about languages and locales ---
We will now ask you to provide the list of languages you want to use.
Please list all the desired languages, comma-separated, using ISO 639-1 codes.  The first language will be used as the default.
Type '?' (a question mark, sans quotes) to list available languages.
Language(s) to use [en]: uk

Please choose the correct time zone for your blog. Nikola uses the tz database.
You can find your time zone here:
https://en.wikipedia.org/wiki/List_of_tz_database_time_zones

Time zone [Europe/Kiev]:
    Current time in Europe/Kiev: 14:05:35
Use this time zone? [Y/n] Y
--- Questions about comments ---
You can configure comments now.  Type '?' (a question mark, sans quotes) to list available comment systems.  If you do not want
any comments, just leave the field blank.
Comment system:

That's it, Nikola is now configured.  Make sure to edit conf.py to your liking.
If you are looking for themes and addons, check out https://themes.getnikola.com/ and https://plugins.getnikola.com/.
Have fun!
[2021-12-04 14:05:44] INFO: init: Created empty site at test.
```

Рис. 2.2.3 – Інтерактивний майстер початкового налаштування параметрів проекту

Далі треба перейти в створену директорію, звідки можна згенерувати сайт та запустити тестовий вебсервер (рис. 2.2.4).

```
(env) PS D:\Files\University\4th year\Diploma\nikola_test> cd test
(env) PS D:\Files\University\4th year\Diploma\nikola_test\test> nikola auto
```

Рис. 2.2.4 – Перехід в директорію проекту і запуск автоматичної генерації

Можна переглянути заготовку сайту за адресою http://127.0.0.1:8000/ (рис. 2.2.5).

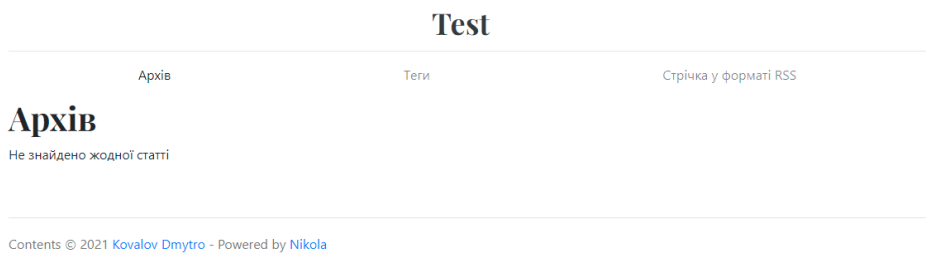

Рис. 2.2.5 – Початковий вигляд сайту

Щоб додати нову сторінку, потрібно спочатку створити її командою new_page (рис. 2.2.6)

```
(env) PS D:\Files\University\4th year\Diploma\nikola_test\test> nikola new_page
Creating New Page
-----------------

Title: Демонстраційна сторінка
Scanning posts........done!
[2021-12-04 14:20:23] INFO: new_page: Your page's text is at: pages\demonstratsiina-storinka.rst
```

Рис. 2.2.6 – Створення нової сторінки

Запускається мастер створення сторінки, який запитує її назву, та генерує відповідний файл сторінки в директорії pages. Вміст згенерованого файлу подано в лістингу 2.2.1.

Лістинг 2.2.1 – файл demonstratsiina-storinka.rst

```
.. title: Демонстраційна сторінка
.. slug: demonstratsiina-storinka
.. date: 2021-12-04 14:20:23 UTC+02:00
.. tags:
.. category:
.. link:
.. description:
.. type: text

Напишіть вашу сторінку тут
```

Бачимо, що Nikola автоматично задав деякі метадані, а саме назву, шлях до сторінки, дату створення, тип сторінки. Далі потрібно додати цю сторінку в навігацію, для цього треба відредагувати поле PAGES та NAVIGATION_LINKS у файлі налаштувань проекту config.py (фрагменти коду подано в лістингу 2.2.2).

Лістинг 2.2.2 – фрагменти файлу config.py

```
PAGES = (
    ("pages/*.rst", "", "page.tmpl"),
    ("pages/*.md", "", "page.tmpl"),
```

```
    ("pages/*.txt", "", "page.tmpl"),
    ("pages/*.html", "", "page.tmpl"),
)

NAVIGATION_LINKS = {
    DEFAULT_LANG: (
        ("/archive.html", "Архів"),
        ("/categories/", "Теги"),
        ("/rss.xml", "Стрічка у форматі RSS"),
        ("/demonstratsiina-storinka/", "Демонстраційна
сторінка")
    ),
}
```

Далі треба перегенерувати сайт, і можна побачити додану сторінку (рис. 2.2.7).

Рис. 2.2.7 – Вигляд демонстраційної сторінки та навігації сайту

Щоб додати статтю, достатньо створити її (рис. 2.2.8), і вона буде автоматично додана на сайт після його наступної генерації (2.2.9). Nikola так само, як і для сторінок, задає метадані для статей, для них час створення грає більшу роль, бо статті сортуються по ньому.

```
(env) PS D:\Files\University\4th year\Diploma\nikola_test\test> nikola new_post
Creating New Post
-----------------

Title: Демонстраційна стаття
Scanning posts........done!
[2021-12-04 14:38:10] INFO: new_post: Your post's text is at: posts\demonstratsiina-stattia.rst
```

Рис. 2.2.8 – Створення нової статті

Рис. 2.2.9 – Створена стаття відображається в розділі "Архів"

2.3 Формат reStructuredText

reStructuredText (RST, ReST, або reST) – простий для прочитання формат розмітки сторінок, а також парсер. Він використовується для написання документації прямо у файлах з програмним кодом. Парсер проходиться по стрічкам коду і виділяє серед них коментарі, написані на RST, після чого генерує із них документацію. Крім цього RST доволі гнучкий і може також використовуватися для створення простих вебсторінок і написання інших документів. Парсер reStructuredText – це компонент Docutils, а сама мова є покращенням мови StructuredTex. Основною ціллю RST було створення набору інструментів подібних до Javadoc для мови програмування Java або Plain Old Documentaion для Perl. [10]

Цей формат використовується в Nikola по замовчуванню, і з його використанням було виконано цю роботу. reStructuredText надає багато засобів розмітки тексту, найважливіші з них це: створення нумерованих та ненумерованих списків, таблиць, виділення заголовків, виділення важливого тескту, вставлення зображень, зовнішніх посилань, посилань на файли. Також є директиви, які надають можливість вставляти HTML-блоки, блоки з програмним кодом Python, створювати контейнери для тексту, які дозволяють застосовувати до них CSS класи. В Nikola директиви використовуються для

задання метаданих сторінок, таких як автор сторінки, час її створення, заголовок, посилання на сторінку та ін.

Наявні шорткоди (shortcodes), які спрощують додання більш складних елементів, таких як слайдшоу, аудіо. Шорткоди можна завантажувати з плагінами, або відповідно створювати свої. В наступній таблиці подано реалізацію деяких елементів на restructuredText та отриманий з них HTML, згенерований Nikola:

Елемент	RST	HTML
Параграф	Параграф	Параграф
Курсивний текст	*Курсивний текст*	 Курсивний текст
Виділений текст	**Виділений текст**	 Виділений текст
Заголовок	Заголовок *********	Заголовок
Посилання	`Посилання </index/>`__	 Посилання
Ненумерований список	- елемент 1 - елемент 2 - елемент 3	<ul class"simple"> елемент 1 елемент 2 елемент 3
Нумерований список	1. елемент 1 2. елемент 2 3. елемент 3	<ol class"arabic simple"> елемент 1 елемент 2

		      елемент 3
Зображення	.. image:: /images/building.jpg	<img alt="/images/building.jpg" src="../images/building.jpg">
Таблиця	.. list-table:: Title :widths: 25 25 50 :header-rows: 1 * - Heading 1 - Heading 2 - Heading 3 * - Row 1, column 1 - - Row 1, column 3 * - Row 2, column 1 - Row 2, column 2 - Row 2, column 3	<table> <caption>Title</caption> <colgroup> <col style="width: 25%"> <col style="width: 25%"> <col style="width: 50%"> </colgroup> <thead><tr> <th class="head"> Heading 1 </th> <th class="head"> Heading 2 </th> <th class="head"> Heading 3 </th> </tr></thead> <tbody> <tr> <td> Row 1, column 1 </td> <td></td> <td> Row 1, column 3 </td> </tr> <tr> <td> Row 2, column 1 </td> <td> Row 2, column 2 </td> <td> Row 2, column 3 </td> </tr> </tbody> </table>

2.4 Структура проекту

Структуру проекту розглянемо на прикладі розробленого сайту, її подано на рис. 2.4.1. В корені проекту наявні наступні директорії: files, galleries, images, listings, output, pages, posts. galleries та listings не використовувалися при розробці сайту, вони потрібні для створення галерей зображень та збереження відрізків програмного коду, які можуть показуватися на сайті.

В директорії files містяться файли, в основному pdf документи, які опубліковано на сайті. В директорії images знаходяться зображення для

сторінок сайту, для зручності вони розподілені на піддиректорії для кожної сторінки, або групи сторінок, на яких використовуються зображення.

В директорії output містяться html файли, які генерує Nikola із вихідних rst файлів, та деякі допоміжні файли.

В директорії pages містяться rst файли всіх розділів сайту окрім новин. В директорії posts містяться rst файли кожної новини.

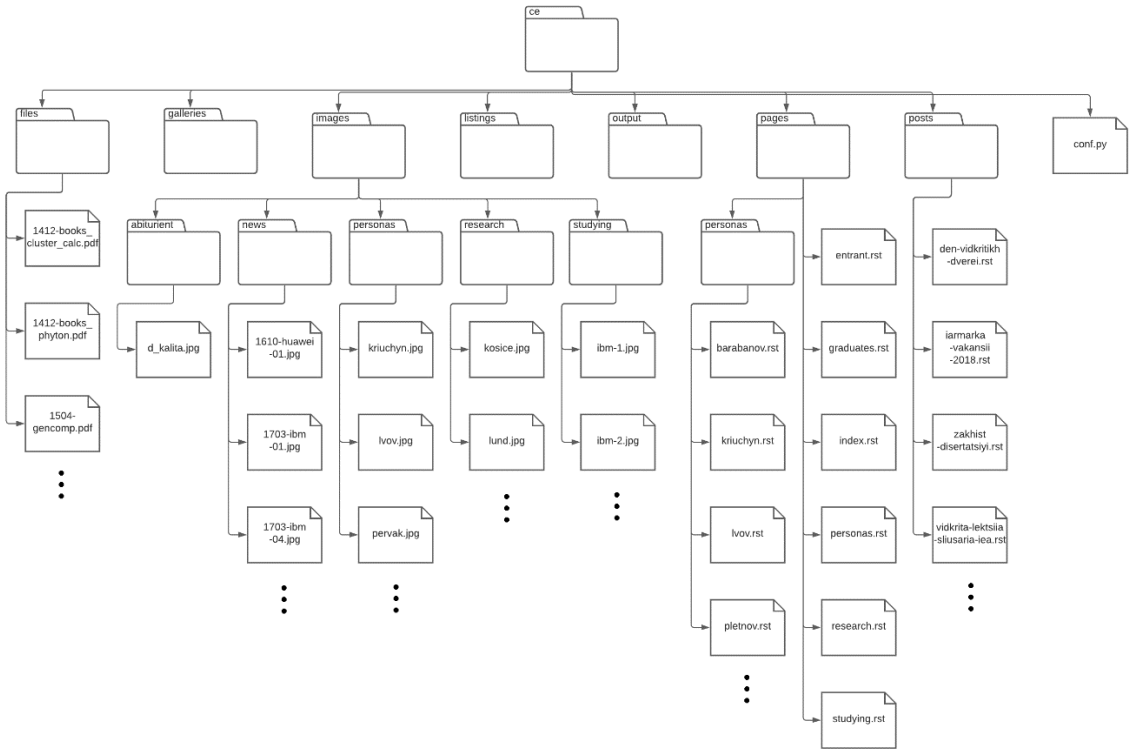

Рис. 2.4.1 – Файлова структура проекту

У файлі conf.py було задано назву сайту “Комп’ютерна інженерія”, обрано українську мову, як мову за замовчуванням, дозволено використання тизерів для статей, увімкнено спрощені посилання, вимкнено показ джерел сторінки, обрано часову зону. Крім того налаштовано навігацію та обрано шлях /news/ для виводу статей.

Сторінки “Комп’ютерна інженерія”, “Навчальна робота”, “Дослідження”, “Абітурієнту” та “Наші випускники” реалізовані подібним чином: їх rst файли знаходяться в директорії pages, контент на них досить простий у вигляді тексту, відцентрованих зображень з підписами, перелічень, посилань на додаткові ресурси та на файли. Шлях до rst файлу вказаний в налаштуваннях навігації.

Сторінка новин виконана у вигляді блогу, тобто кожна новина створюється як стаття командою new_post, і автоматично додається на сторінку новин. Новини відсортовані від найновіших до найстаріших, тому все нове автоматично з’являтиметься на початку сторінки. Також наявна навігація між новинами у вигляді кнопок “Наступна стаття” та “Попередня стаття”. Для того, щоб на сторінці новин окрім заголовків статей відображався ще додатковий текст або зображення, в кожній статті задано тизер – верхня частина статті, яка обмежується директивою TEASER_END.

На сторінці персоналій засобами bootstrap кожен співробітник кафедри поданий у вигляді фото з лівого боку та імені й посади з правого. Ім’я при цьому слугує посиланням на особисту сторінку людини (якщо така наявна, інакше – назад на сторінку персоналій). Rst файли особистих сторінок містяться у піддиректорії personas директорії pages, де сторінка кожної людини має назву їх фамілії.

3 РОЗГОРТАННЯ САЙТУ ЗА ДОПОМОГИ CI/CD ТА GITLAB

3.1 Docker

Docker – це програмна платформа, яка дозволяє швидко збирати, тестувати та розгортати додатки. Docker організує програмне забезпечення в стандартизовані одиниці, які називаються контейнерами. Які містять все потрібне для роботи програм, включають в себе бібліотеки, системні інструменти, програмний код та ін. Docker надає зручні команди для керування контейнерами, автоматизації їх запуску і розгортання, керує життєвим циклом, дозволяє запускати декілька контейнерів на одній хостовій машині.

Контейнери – це спосіб упакувати додаток і всі його залежності в один образ. Цей образ запускатиметься в ізольованому середовищі. Контейнери дозволяють відділити додаток від інфраструктури: розробникам не потрібно брати до уваги середовище, в якому оточенні працюватиме їх додаток. Усі залежності та налаштування компонуються в єдиний образ. Потім цей образ можна запускати на інших системах, без додаткового налаштування середовища виконання.

Контейнеризація схожа на віртуалізацію, але це не одне й те ж. Віртуалізація працює як окремий комп’ютер, зі своїм віртуальним обладнанням і операційною системою. При цьому на базі хостової ОС можливо розгорнути декілька гостьових ОС, відмінних від хостової. У випадку контейнеризації віртуальна середа запускається як процес у рамках основної операційної системи і не віртуалізує обладнання. Це означає, що контейнер може працювати тільки в тій же ОС, що і основна. При цьому, через те, що

контейнери не віртуалізують обладнання, вони споживають набагато менше ресурсів. [14]

3.2 CI/CD

CI/CD – це поширений метод розгортання застосунків, який передбачає використання автоматизації на різних стадіях розробки. Основні концепції, які приписуються CI/CD, – це неперервна інтеграція (continuous integration), неперервне розгортання (continuous delivery, continuous deployment). CI/CD вирішує проблеми, спричинені інтеграцією нового коду, для команд розробників.

Неперервна інтеграція означає, що нові зміни в коді регулярно компілюютсья, тестуються та інтегруються в спільний репозиторій. Це вирішує проблему, яка з'являється в застосунках, які активно розроблюються, коли наявна велика кількість гілок, які можуть конфліктувати між собою.

Неперервне розгортання передбачає, що після того, як зміни в коді були протестовані та заватажені в репозиторій, вони будуть автоматично додані до кінцевого продукту. В контексті розробки web-застосунків, це означає, що зміни в репозиторії переносяться на веб сервер, і відповідно їх можна побачити на сайті.

Pipeline – високорівневий компонент CI/CD. Pipeline скаладаються з завдань (jobs), які визначають які саме дії виконувати, на приклад компілювати та тестувати програмний код; та етапів (stages), які визначають коли запускати завдання, на приклад спочатку йде етап компіляції коду, а потім етап його тестування.

Завдання виконуються за допомогою виконавців (runners). Декілька завдань в одному етапі можуть виконуватися паралельно, якщо є достатня

кількість одночасних виконавців. Тільки після того, як всі завдання в етапі виконані, pipeline переходить до наступного етапу. Якщо ж виконання якогось із завдань не увінчується успіхом, то наступний етап зазвичай не починає виконуватись, і pipeline зупиняється передчасно.

Зазвичай, pipeline запускається автоматично і не потребує додаткового втручання після його створення, але передбачені можливості ручної взаємодії з ним.

Артефакти – це можливий результат роботи pipeline, який складається із архіву файлів і директорій. На приклад це можуть бути бінарні файли скомпільованої програми, або файли статичного сайту, створені методом статичної генерації. [17]

3.3 Git

Git – це сучасна і розповсюджена система контролю версій із відкритим вихідним кодом. Вона розроблена для керування малими та великими проектами із високою швидкістю та ефективністю, дозволяє координувати роботу між кількома розробниками. Контроль версій дозволяє відстежувати роботу команді розробників в межах робочого простору. Git – це основна складова таких сервісів як GitHub та GitLab, але може бути використаним і окремо від них. Git легко вивчаєтсья, і має високу продуктивність в порівнянні з іншими подібними інструментами, такими як Subversion, CVS, Perforce та ClearCase. Основними перевагами Git є:

- Відкритість коду.
- Масштабованість, тобто підтримка великої кількості користувачів.
- Розподіленість, яка означає, що проект зберігається не тільки в центральному репозиторії, а й на локальній машині кожного

користувача, і містить повну історію всіх комітів, тому працювати можна не підключаючись до віддаленого репозиторію, і синхронізовувати зміни тільки на ключових моментах розробки.

- Захищеність, git використовує SHA1 для ідентифікації об'єктів в репозиторії. Файли і коміти перевіряються по чексумі при закінченні роботи. Історія зберігається таким чином, що ідентифікатор кожного коміту залежить від повної історії розробки до того моменту. Після опублікування неможливо внести зміни в старі версії, але їх можна отримати по ідентифікатору і далі з ними працювати, якщо є така потреба, такий функціонал за замовчуванням відсутній у багатьох системах контролю версій.
- Висока швидкодія. Git дуже швидко виконує всі завдання, більшість операцій виконуються в локальному репозиторії, і на відміну від централізованих систем контролю версії, Git не підтримує постійне з'єднання з віддаленим сервером. Тести швидкодії, проведені Mozilla показали, що Git набагато швидше за інші подібні системи, в часності отримання історії проекту набагато швидше із локального репозиторію. Основна частина Git написана мовою C, яка фактично не має накладних втрат при роботі в порівнянні з високорівневими мовами програмування.
- Підтримує нелінійну розробку. Git підтримує непомітні розгалудження і злиття, що допомагає візуалізувати нелінійну розробку. Гілка в Git репрезентує один коміт. Розгалудження і злиття роблять Git відмінним від інших інструментів, він дозволяє створювати декілька гілок, які не впливають одна на одну; можна виконувати такі дії, як створення, видалення і злиття гілок, які при цьому виконуютсья дуже швидко. Розгалудження надає такі переваги: можна створити окрему гілку для нового модуля проекту, закомітити чи видалити її коли завгодно; можна

мати окрему виробничу гілку, в якій міститься проект, який розгортається і власне працює, і з цієї гілки завжди можна розгалуджитись в окрему гілку для тестування; можна мати демонстраційну гілку для експериментів і перевірок роботи, яку також можна видалити у разі потреби; одна з головних переваг розгалудженості віток, це те, що коли треба завантажити наробки у віддалений репозиторій, немає потреби надсилати усі гілки, а можна вибрати тільки ті, які необхідні.

- Проміжна зона також є унікальним функціоналом Git, його можна вважати попередньою версією наступного коміту. В цій зоні майбутні коміти можна форматувати і переглядати перед закінченням. Коли виконується коміт, Git бере зміни, які знаходяться в проміжній зоні і оформлює їх як новий коміт. Можна додавати і прибирати зміни в цій зоні, по суті вона є місцем, де Git зберігає всі поточні зміни. Git не використовує окрему директорія, де можуть зберігатися файли, які репрезентують внесені зміни, натомість використовується файл index (рис. 2.3.1). Крім того можна занести в проміжну зону деякі файли і виконати коміт, при цьому не комітити інші модифіковані файли в робочій директорії.

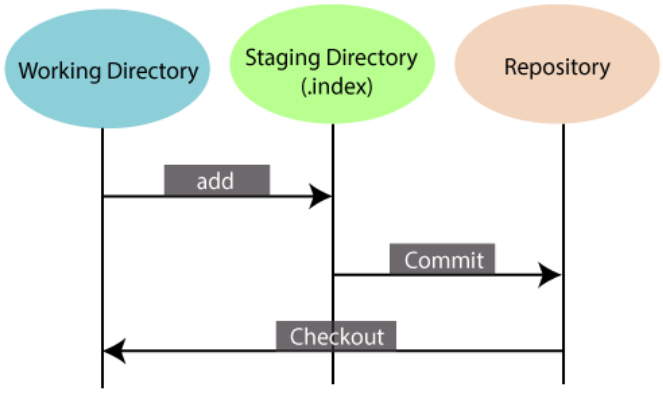

Рис. 2.3.1 – Діаграма взаємодії з проміжною зоною

- Збереження чистої історії. Git полегшує роботу завдяки Git Rebase – це одна з найзручніших функцій Git, яка дозволяє завантажити найостанніші коміти з віддаленого репозиторію, і ставить поточні зміни в робочій директорії поверх них, що забезпечує чистоту історії проекту.
- Контроль версій дозволяє відслідковувати зміни в файлах проекту. Кожен раз коли вносяться зміни, вони можуть бути завантажені у спільний віддалений репозиторій. Це дозволяє виправляти помилки, шляхом відкату до попередньої версії, дає можливість спостерігати за процесом розробки, перевіряти статус, порівнювати файли з різних гілок. [4]

3.4 GitLab

GitLab – DevOps платформа з відкритим вихідним кодом, яка об'єднує в собі контроль версій з застосуванням git, систему тікетів, засоби CI/CD та ін. В Інформаційно Обчислювальному Центрі наявний локальний сервер зі встановленим GitLab, який використовується для розробки більшості програмних продуктів, в тому числі сайту кафедри комп'ютерної інженерії.

GitLab, як і інші подібні платформи підтримує створення Pages сайту. Це дуже зручний інструмент, який дозволяє розгортати сайти в GitLab та швидко оновлювати їх при комітах в репозиторій. Це виконується засобами CI/CD, обирається Docker образ, на якому виконуватимуться подальші скрипти, створюється завдання, зі спеціальною назвою "pages", в якому зазначаються дії, які необхідно зробити із файлами з репозиторію, щоб отримати робочі файли сайту, після чого ці артефакти потрібно помістити в спеціальну директорію "public". Таким чином при новому коміті в репозиторій запускатиметься створений pipeline, який має 2 етапи: виконання дій, заданих нами по генерації потрібних артефактів; розгортання сайту. [8]

3.5 Розгортання сайту

Програмний код сайту зберігається в репозиторії https://gl.icc.knu.ua/ci_site/ci-dep-site-static-nikola/. В проект додано .gitlab-ci.yml файл (Лістинг 3.6.1), в якому задані інструкції для GitLab CI/CD. При оновленні файлів проекту запускається обробник, який виконує заданий сценарій, який включає компіляцію проекту, та опублікування отриманих файлів на веб сервер.

Лістинг 3.6.1 – файл .gitlab-ci.yml

```
image: registry.gitlab.com/registryimage/nikola

pages:
  script:
    - cd ce
    - nikola build
    - mv output ../public
  artifacts:
    paths:
    - public
  only:
  - main
```

Як базовий образ було обрано спеціальний образ nikola, який зберігається в репозиторії https://gitlab.com/registryimage/nikola. Його було обрано, бо для роботи nikola потрібно встановити певний перелік залежностей, який включає мову програмування Python, і сам статичний генератор nikola. Це можна було б зробити вручну, прописавши свій Dockerfile, але було прийнято рішення використати саме цей образ, бо він регулярно оновлюється, у випадку виходу нової версії nikola, має встановленими всі потрібні залежності, включаючи ті, які дозволяють генерувати сторінки сайту із Jupyter Notebooks.

В даному контейнері виконується генерація сайту командою nikola build, після чого згенерований сайт із диреткорії output переписується в директорію

public, яка зберігається у вигляді артефакту. Ці дії відбуваються тільки для вітки main, тому можна вести розробку в інших вітках не змінюючи стан сайту, і тільки по закінченню всієї роботи і її перевірки виконати злиття віток, і оновити сайт свіжими доробками.

Блок-схему оновлення і розгортання сайту подано на рис. 3.6.1.

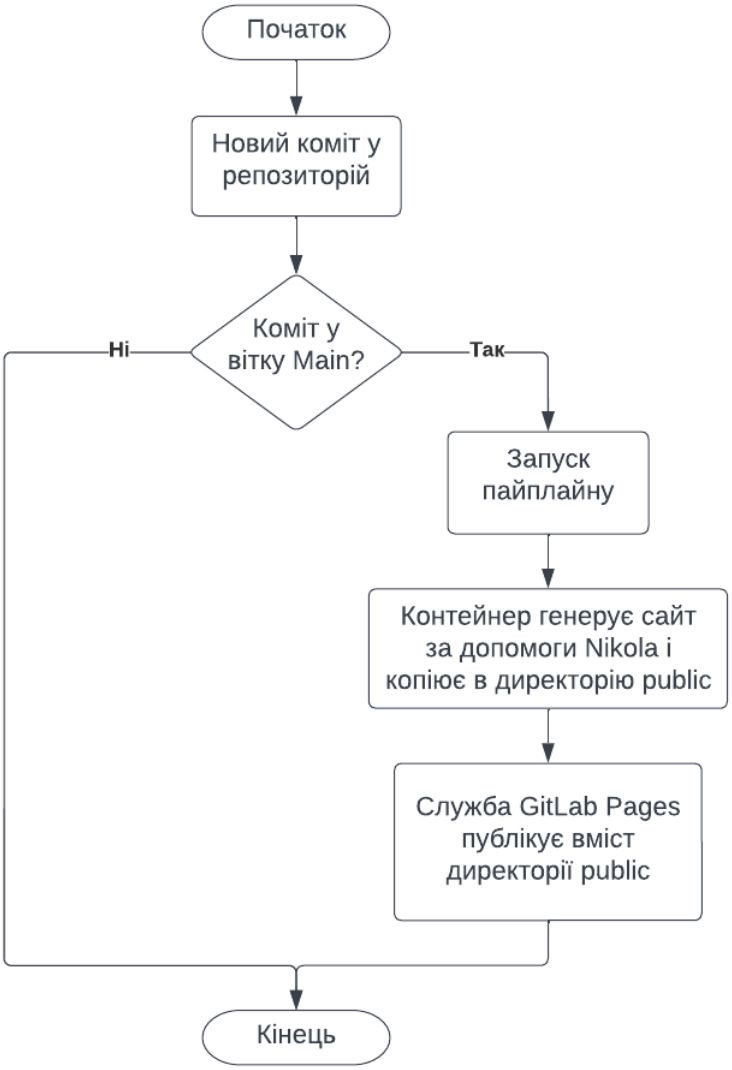

Рис. 3.6.12 – Алгоритм оновлення і розгортання сайту

При виконанні цього pipeline є два етапи (рис. 3.6.2): перший було описано в файлі .gitlab-ci.yml, який включає в себе завдання pages по генерації сайту; після цього запускається спеціальне завдання pages:deploy, яке підготовлює файли сайту для служби GitLab Pages.

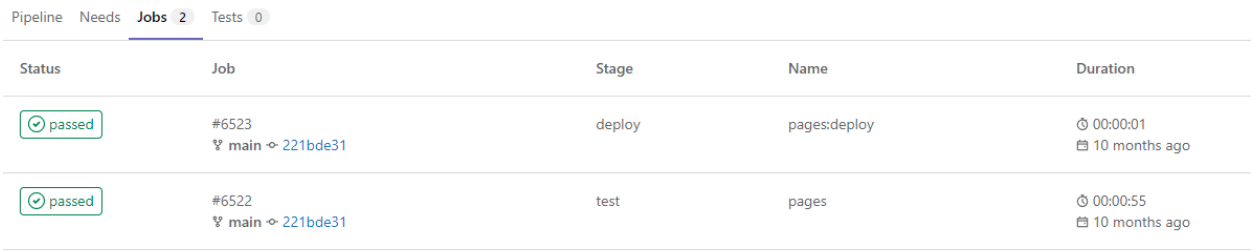

Рис. 3.6.2 – Результат успішного виконання pipeline в GitLab

Завдяки тому, що сайт генерується в контейнері, немає потреби зберігати в репозиторії директорії cache, та output, тому було прийняте рішення прибрати їх. Для цього потрібно оновити вміст файлу .gitignore, додати в нього ці директорії, а також виконати очистку історії. Це можна досягнути наступною командою, де path_to_file заміняється шляхом до диреторії або файлу, який треба видалити:

```
git filter-branch -f --index-filter "git rm -rf --cached --ignore-unmatch path_to_file" HEAD
```

`git filter-branch` дозволяє переписувати історію версій проекту в гілках репозиторію, застосовуючи спеціальні фільтри для кожної версії. Ці фільтри можуть модифікувати дерево, наприклад видалити файл, або змінити інформацію кожного коміту. Вся інша інформація, включаючи оригінальний час комітів, або інформація про злиття, зберігається без змін. Переписана історія матиме інші назви об'єктів, і не буде збігатися з оригінальною віткою, тому, щоб запушити її у віддалений репозиторій, потрібно використовувати примусову опцію `--force`, перед чим дозволити такі дії в налаштуваннях

репозиторію, адже за замовчуванням вони заборонені в захищених вітках. Для того щоб визначити, які саме об'єкти потрібно модифікувати використовуються фільтри, в нашому випадку було обрано фільтр `--index-filter`, який вносить зміни в проміжний каталог git. Цей фільтр схожий на `--tree-filter`, але він не перемикається між гілками, через що набагато швидший. [6]

Після зазначення фільтру слідує сама команда, яка виконуватиметься в оболонці операційної системи функцією eval. Якщо виконання команди поверне ненульовий результат, то вся операція відміняється. В нашому випадку команда має вигляд `git rm -rf --cached --ignore-unmatch path_to_file`. Команда `git rm` видаляє файли з робочого дерева та з проміжного каталогу; опція `--cached` дозволяє видаляти файли лише з проміжного каталогу; опція `-r` дозволяє рекурсивне видалення файлів всередині директорії; опція -f дозволяє обійти обмеження, яке забороняє видаляти файли, які не ідентичні кінцю гілки (тобто старі версії фалів); `--ignore-unmatch` зазначає, що навіть якщо не було знайдено файлу, то виходити із кодом 0, таким чином якщо ми видаляємо із історії файл, який з'явився не на самому початку проекту, то в старіших версіях де його не було програма не відзвітує про помилку, і вся операція по видаленню не відміниться.

Перш ніж виконувати дії, які будуть переписувати історію репозиторію варто зробити його резервну копію командою `git clone`.

Щоб порівняти розміри репозиторіїв було виконано наступні команди в резервній копії репозиторію, та в поточній версії:

```
git gc
```

```
git count-objects -vH
```

Команда `git gc` використовується з метою очистки локального репозиторію від непотрібних файлів. Вона запускає ряд обслуговуючих завдань в поточному репозиторії, як наприклад стискання ревізій файлів (для зменшення занятого місця на диску та покращення продуктивності); прибирання недосяжних об'єктів, які могли бути створеними від попередніх виконань `git add`; упакування посилань, підчищання reflog, метаданих або устарівших робочих дерев; може також оновити допоміжні переліки, як наприклад граф комітів. [7]

Після приведення репозиторіїв до однакових умов виконується `git count-objects -vH`, яка рахує кількість незапакованих файлів об'єктів, і об'єм на диску, який вони займають. Опція -v зазначає, що звіт після роботи команди має бути розширеним, і включати такі пункти: count – кількість вільних об'єктів; size – об'єм на диску, який зайнято вільними об'єктами; in-pack – кількість запакованих об'єктів; size-pack – об'єм на диску, який зайнятий запакованими об'єктами; prune-packable – кількість вільних об'єктів, які також присутні в пакунках; garbage – кількість об'єктів в базі даних які не являються ні валідними вільними об'єктами, ні валідними запакованими; size-garbage – об'єм на диску, який займають невалідні об'єкти. Опція -H зазначає, що розміри, які займають файли на диску будуть показуватися у зручних одиницях вимірювання, а не просто в кілобайтах. Серед перелічених параметрів найважливішим є size-pack, який по суті показує розмір репозиторію. [5]

Отримані результати подано на рис. 3.6.3 для резервної копії та на рис. 3.6.4 для поточної версії.

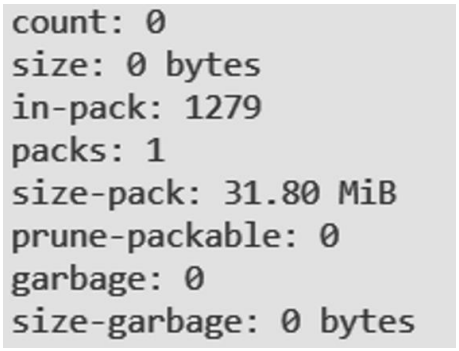

Рис. 3.6.3 – Результат підрахунку об'єму репозиторію для резервної копії

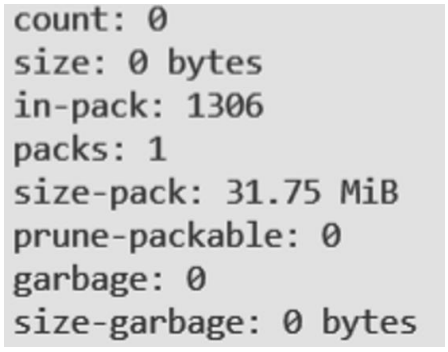

Рис. 3.6.4 – Результат підрахунку об'єму репозиторію для поточної версії

Бачимо, що після виконаного видалення директорій cache, та output розмір зменшився на 0.05 МБ, і тепер в майбутньому зміни в цих директоріях не будуть відслідковуватись взагалі.

4 ВИСНОВКИ

Засоби статичної генерації сайтів можуть використовуватись у тих випадках, коли контент сайту змінюється рідко. Використання таких сайтів має певні переваги: для їх роботи не потрібно встановлювати додаткове програмне забезпечення, таке як PHP або ASP.NET, загалом не потребує залучення баз даних. Звернення до сторінки, на відміну від динамічних сайтів, не вимагає повторної генерації сторінки. Це зменшує витрати обчислювальних ресурсів вебсерверу, спрощує структуру програмної частини вебсерверу і є більш економічним в експлуатації.

Використання статичних генераторів дозволяє автоматизувати процес генерації вебсайту з вхідних даних. Це дозволяє зменшити витрати часу на оновлення контенту та технічне обслуговування сайту.

Розгортання сайтів, створених на базі статичної генерації, доцільно реалізовувати засобами CI/CD та Docker, використовуючи операцію commit у певну вітку репозиторію як сигнал для запуску ПЗ статичного генератору.

Сайт кафедри комп'ютерної інженерії відповідає вимогам керівництва, і наразі введений в експлуатацію.

Наведені в цій роботі матеріали є достатніми для подальшого технічного обслуговування і оновлення контенту сайту кафедри комп'ютерної інженерії, а також можуть використовуватись для створення нових сайтів університету, для яких підходить технологія статичної генерації.

СПИСОК ЛІТЕРАТУРИ

1. Best Static Site Generators [Електронний ресурс] – режим URL: https://websitesetup.org/best-static-site-generators/
2. Building a site using Nikola [Електронний ресурс] – Режим доступу URL: https://adriaanrol.com/posts/building-a-site-using-nikola/
3. Create Documentation with RST, Sphinx, Sublime, and GitHub [Електронний ресурс] – Режим доступу URL: https://sublime-and-sphinx-guide.readthedocs.io/en/latest/
4. Git Tutorial [Електронний ресурс] – режим доступу URL: https://www.javatpoint.com/git
5. Git-count-objects [Електронний ресурс] – режим доступу URL: https://git-scm.com/docs/git-count-objects
6. Git-filter-branch [Електронний ресурс] – режим доступу URL: https://git-scm.com/docs/git-filter-branch
7. Git-gc [Електронний ресурс] – режим доступу URL: https://git-scm.com/docs/git-gc
8. GitLab Pages [Електронний ресурс] – режим доступу URL: https://docs.gitlab.com/ee/user/project/pages/
9. Nikola — Static Site Generator [Електронний ресурс] – Режим доступу URL: https://getnikola.com/
10. reStructuredText Markup Syntax and Parser Component of Docutils [Електронний ресурс] – режим доступу URL: https://docutils.sourceforge.io/rst.html
11. reStructuredText Primer CheatSheet [Електронний ресурс] – Режим доступу URL: https://www.sphinx-doc.org/en/master/usage/restructuredtext/basics.html

12. Sphinx and RST syntax guide (0.9.3) [Електронний ресурс] – Режим доступу URL: https://thomas-cokelaer.info/tutorials/sphinx/rest_syntax.html

13. Static and dynamic websites – what's the difference? [Електронний ресурс] – Режим URL: https://www.mlytics.com/blog/static-and-dynamic-websites-whats-the-difference/

14. Use containers to Build, Share and Run your applications [Електронний ресурс] – режим доступу URL: https://www.docker.com/resources/what-container/#:~:text=A%20Docker%20container%20image%20is,tools%2C%20system%20libraries%20and%20settings.

15. venv — Creation of virtual environments [Електронний ресурс] – Режим доступу URL: https://docs.python.org/3/library/venv.html

16. What is a static site generator? [Електронний ресурс] – Режим URL: https://www.cloudflare.com/learning/performance/static-site-generator/#:~:text=A%20static%20site%20generator%20is,to%20users%20ahead%20of%20time.

17. What is CI/CD? [Електронний ресурс] – Режим доступу URL: https://www.redhat.com/en/topics/devops/what-is-ci-cd