InterConf
Scientific Publishing Center

THE ISSUE CONTAINS:

Proceedings of the 6th International Scientific and Practical Conference

SCIENTIFIC PARADIGM IN THE CONTEXT OF TECHNOLOGIES AND SOCIETY DEVELOPMENT

Geneva, Switzerland
26-28.11.2023

Scientific Collection
InterConf

№ 180
November, 2023

OPEN ACCESS

LIGHT INDUSTRY AND FOOD INDUSTRY

RADIO ENGINEERING, ELECTRONICS AND ELECTRICAL ENGINEERING

INFORMATION AND WEB TECHNOLOGIES

Proceedings of the 6th International
Scientific and Practical Conference
«Scientific Paradigm in the Context of
Technologies and Society Development»
(November 26-28, 2023).
Geneva, Switzerland

No
180

INFORMATION AND WEB TECHNOLOGIES

Впровадження конвеєру безперервної інтеграції та постачання для веб-застосунку

Голубничий Дмитро Юрійович[1], **Коломійцев Олексій Володимирович**[2], **Третяк Вячеслав Федорович**[3], **Діденко Сергій Сергійович**[4], **Рибальченко Аліна Олександрівна**[5], **Любченко Олексій Вікторович**[6], **Рудаков Ігор Сергійович**[7]

[1] кандидат технічних наук, доцент, доцент кафедри Інформаційних систем;
Харківський національний економічний університет імені Семена Кузнеця; Україна

[2] Заслужений винахідник України, доктор технічних наук,
професор кафедри комп'ютерної інженерії та програмування;
Національний технічний університет «Харківський політехнічний інститут»; Україна

[3] кандидат технічних наук, доцент, старший науковий співробітник,
науковий співробітник наукового центру Повітряних Сил;
Харківський національний університет Повітряних Сил імені Івана Кожедуба; Україна

[4] магістрант кафедри Інформаційних систем;
Харківський національний економічний університет імені Семена Кузнеця; Україна

[5] аспірантка кафедри комп'ютерної інженерії та програмування;
Національний технічний університет «Харківський політехнічний інститут»; Україна

[6] аспірант кафедри комп'ютерної інженерії та програмування;
Національний технічний університет «Харківський політехнічний інститут»; Україна

[7] аспірант кафедри комп'ютерної інженерії та програмування;
Національний технічний університет «Харківський політехнічний інститут»; Україна

Розглядається використання стеку MEAN для розробки веб-застосунків. Скорочення MEAN утворюється з наступних елементів: MongoDB, Express.js, Angular та Node.js. MEAN є уніфікованим стеком JavaScript, який переважно використовується для створення хмарних програм [1-6]. Використання стеку MEAN (MongoDB, Express.js, Angular і Node.js) має наступні переваги:

– *уніфікований стек мов програмування:* одна мова (JavaScript) використовується на усіх етапах розробки (від бази даних (БД) до клієнтського і серверного коду), що полегшує взаємодію між різними частинами застосунку та

INFORMATION AND WEB TECHNOLOGIES

спрощує процес розробки.

– *гнучкість та масштабованість:* MongoDB, яка використовується для зберігання даних, є гнучкою NoSQL БД, що сприяє легкості змін та розширенню структури даних;

– *швидка реакція на події:* Node.js забезпечує асинхронну обробку подій, що дозволяє ефективно взаємодіяти з багатьма запитами одночасно і поліпшує швидкодію застосунку;

– *розширюваність фронтенду:* Angular, як фреймворк для розробки клієнтського коду, надає потужні інструменти для створення складних та масштабованих користувацьких інтерфейсів;

– *швидка розробка:* Express.js дозволяє швидко створювати серверні застосунки та API, що сприяє швидкій розробці та внесенню змін;

– *підтримка спільноти:* всі компоненти MEAN (MongoDB, Express.js, Angular, Node.js) користуються активною та великою спільнотою розробників, що дозволяє отримувати швидку підтримку та розвивати проекти;

– *ідеальний для хмарних застосунків:* MEAN є популярним вибором для хмарних застосунків, оскільки усі компоненти підтримують легку масштабованість та деплоймент в хмарному середовищі.

Загалом, MEAN надає повний стек технологій для розробки веб-застосунків, що дозволяє ефективно поєднувати різні елементи та створювати високоефективні та гнучкі застосунки. Додатки, розроблені на базі MEAN, оптимізовані для ефективного використання переваг хмарного середовища, що призводить до збереження витрат та підвищення продуктивності [2-5] (рис. 1).

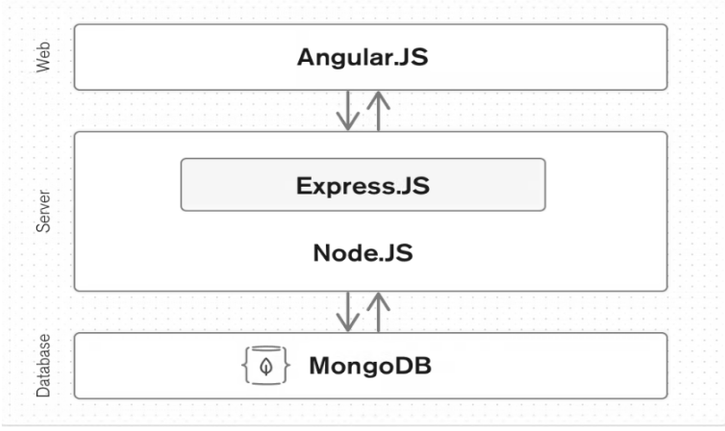

Рисунок 1

Взаємодія складових стеку MEAN

INFORMATION AND WEB TECHNOLOGIES

Незважаючи на багато переваг, стек MEAN (MongoDB, Express.js, Angular і Node.js) також має свої недоліки:

– *складність навчання:* особливо для початківців, вивчення усіх компонентів MEAN може бути складним. Відсутність досвіду з асинхронним програмуванням або з роботою з NoSQL БД може становити виклик для новачків;

– *висока складність коду:* створення та підтримка коду великих застосунків на MEAN може стати важкою задачею через велику кількість різних компонентів та їх взаємодію;

– *велика кількість залежностей:* кожен компонент стеку MEAN є окремим модулем з власним набором залежностей. Управління цими залежностями та їх оновлення може бути складним завданням;

– *складність відлагодження:* відлагодження програмних застосунків, особливо на великих проектах, може бути важким завданням через асинхронність Node.js та складність Angular;

– *великий обсяг вивчення:* для розробки на MEAN необхідно мати глибоке розуміння усіх його компонентів, що може вимагати часу та зусиль;

– *вибір альтернатив:* іноді може виникнути ситуація, коли інші технології або стеки підходять краще для конкретного проекту, але обмеження MEAN може ускладнити зміну технології серед шляхів розробки;

– *проблема з великою кількістю одночасних підключень:* хоча Node.js відомий своєю швидкодією, але для деяких великих проектів може виникнути проблема з великою кількістю одночасних підключень, особливо при великому обсязі обчислень.

Незважаючи на перелічені недоліки, багато з них можна подолати з досвідом та правильним плануванням розробки. Важливо обирати технології відповідно до конкретних потреб проекту і команди розробників.

Таким чином, практики, які ефективно використовуються в одному проєкті, можуть призвести до проблем у іншому.

Розроблений програмний застосунок складається із трьох основних складових: клієнт, сервер та БД. Відповідно до стеку MEAN, на стороні клієнта використовується фреймворк Angular. На стороні серверу використовується фреймворк NestJS, який базується на відомому Node.js фреймворку Express.js. Окрім того, на стороні серверу використано ряд додаткових компонентів (табл. 1). В якості БД використовується MongoDB, а загальну структуру компонентів проєкту представлено на рисунку 2.

На підготовчому етапі створено frontend папку та

INFORMATION AND WEB TECHNOLOGIES

додається у неї проєкт на фреймфоркі Angular. Після цього, створюються компоненти, які необхідні для аутентифікації користувача та головну сторінку застосунку. Структура frontend проєкту має наступний вигляд (рис. 3). На наступному кроці застосовується NestJS для створення backend проєкту.

Для імпорту даних з сервісу The Movie DB надається команда, при запуску якої відбувається запит до стороннього API. Дані зберігаються у БД MongoDB. Слід зазначити, що для команди імпорту даних використовується модуль worker_threads, який дозволяє використовувати потоки для паралельного виконання JavaScript.

Таблиця 1

Додаткові компоненти на стороні серверу

Компонент	Мета використання
Google API	Використовується для реалізації аутентифікації користувача з використанням облікового запису на стороні Gmail
The Movie DB	Сторонній сервіс, який представляє собою базу з інформацією про фільми. Даний сервіс надає API, який використовується в програмному застосунку для порту даних
AWS S3	Використовується для зберігання env файлу, який надає змінні відповідного оточення
AWS Secrets Manager	Використовується для зберігання секретних даних, які потім додаються до змінних оточення
Swagger	Набір інструментів, який дозволяє автоматично описувати API на основі його коду
Postman	Платформа API для створення та використання API

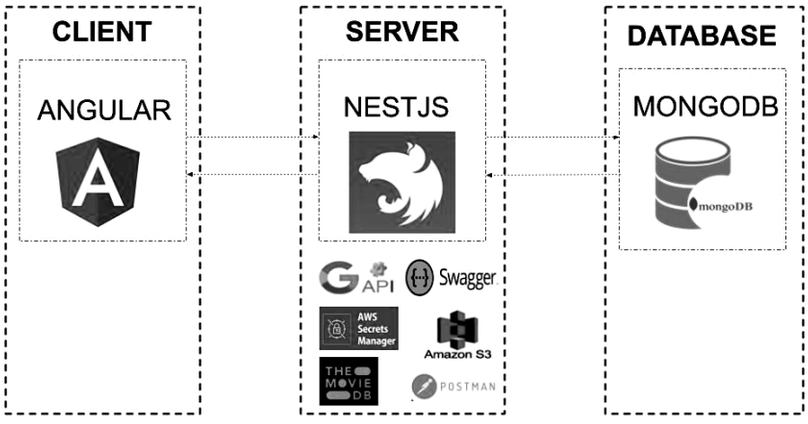

Рисунок 2

Загальна структура компонентів програмного застосунку

INFORMATION AND WEB TECHNOLOGIES

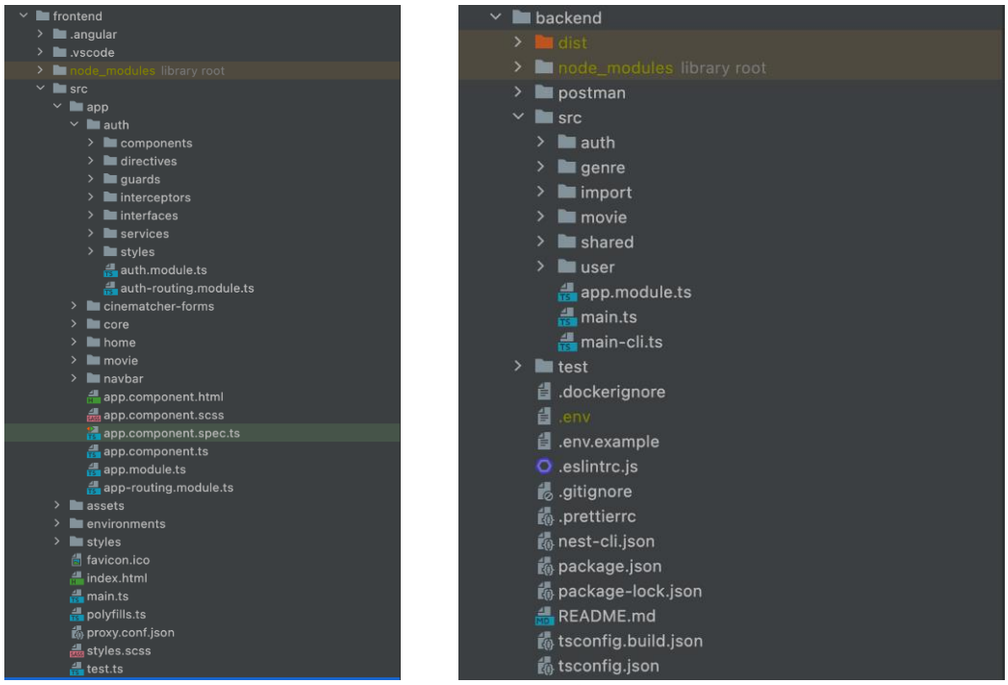

Рисунок 3

Структура frontend (зліва) та backend (зправа) проєкту

Аутентифікацію реалізовано двома способами: за допомогою email на пароль, а також за допомогою облікового запису Google. Для цього, на стороні клієнта створено два компоненти: LoginComponent та RegisterComponent. В результаті отримуємо сторінки для логіна та реєстрації.

На стороні серверу додається логіка для реєстрації та аутентифікації користувача. Процедура авторизації виконується за допомогою облікового запису Google та на основі протоколу OAuth 2.0. При цьому, на стороні Google розгортається клієнт OAuth 2.0 (рис. 4).

На стороні NestJS додається логіка опрацювання запиту на логін з використанням Google. Для цього, використовується бібліотека passport. Також, додається endpoints для обробки запитів від клієнта та запиту від Google.

Для локального оточення файл конфігурацій розміщується всередині папки backend. На основі файлу .env.example відбудовується .env файл.

Щодо інших оточень, то додана логіка встановлення змінних оточення у процесі запуску програмного застосунку. При цьому, дані зберігаються на AWS S3 та AWS Secrets Manager. Приклад файлу конфігурацій для локальної розробки наведений на рисунку 5.

INFORMATION AND WEB TECHNOLOGIES

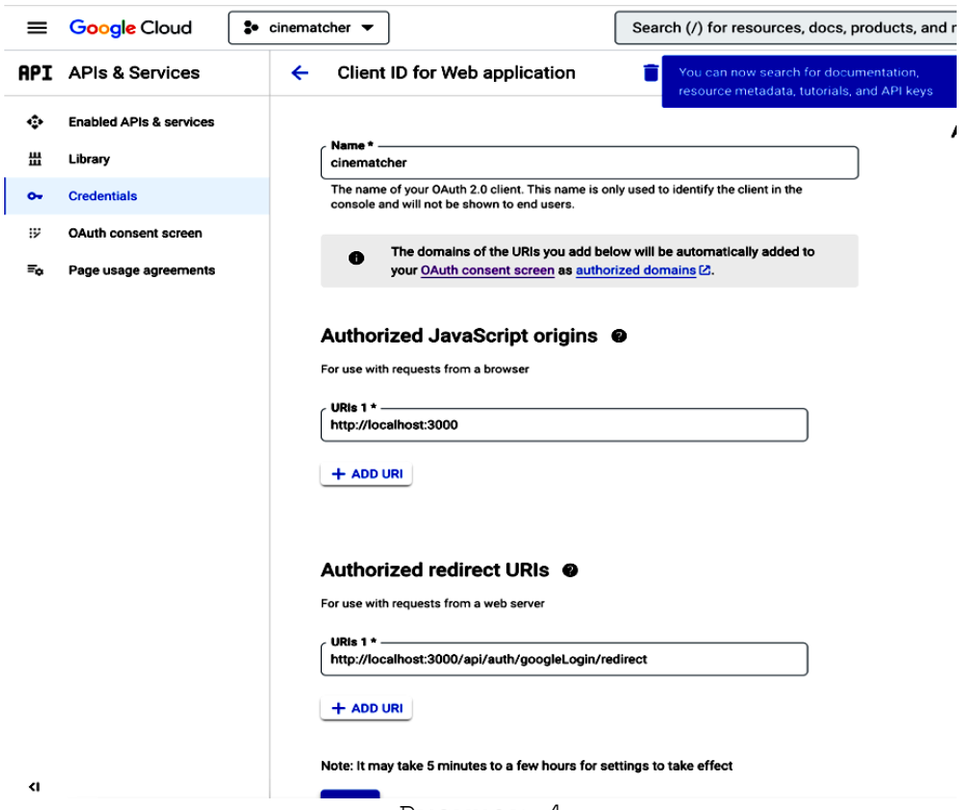

Рисунок 4

Створення OAuth 2.0 клієнта

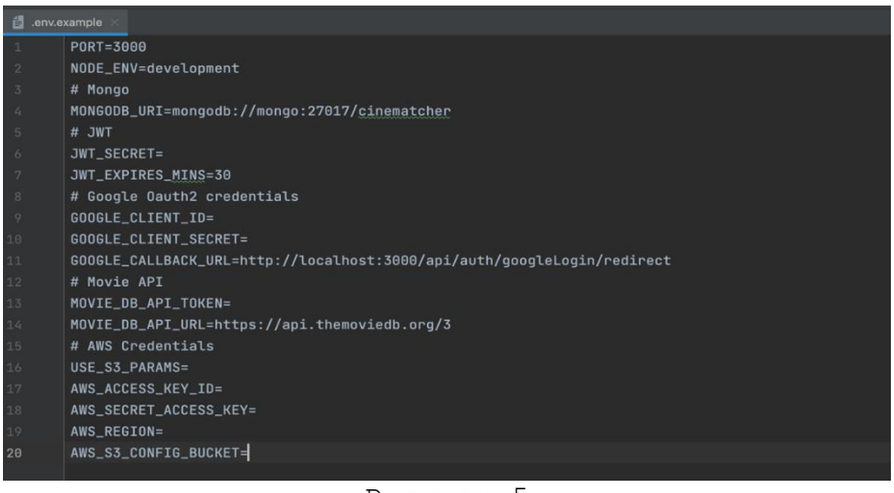

Рисунок 5

Приклад файлу конфігурацій для локальної розробки

З використанням даного файлу формується .env, який додається у .gitignore для виключення файлу з репозиторію.

INFORMATION AND WEB TECHNOLOGIES

Процес формування змінних оточення представлено на рисунку 6.

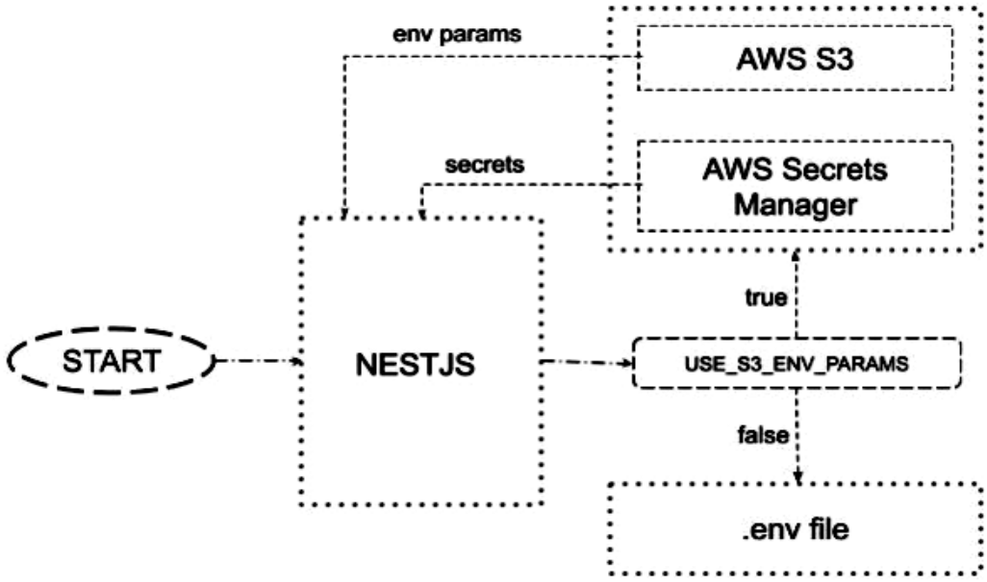

Рисунок 6
Процес формування змінних оточення

За допомогою Swagger генерується документація для API (рис. 7).

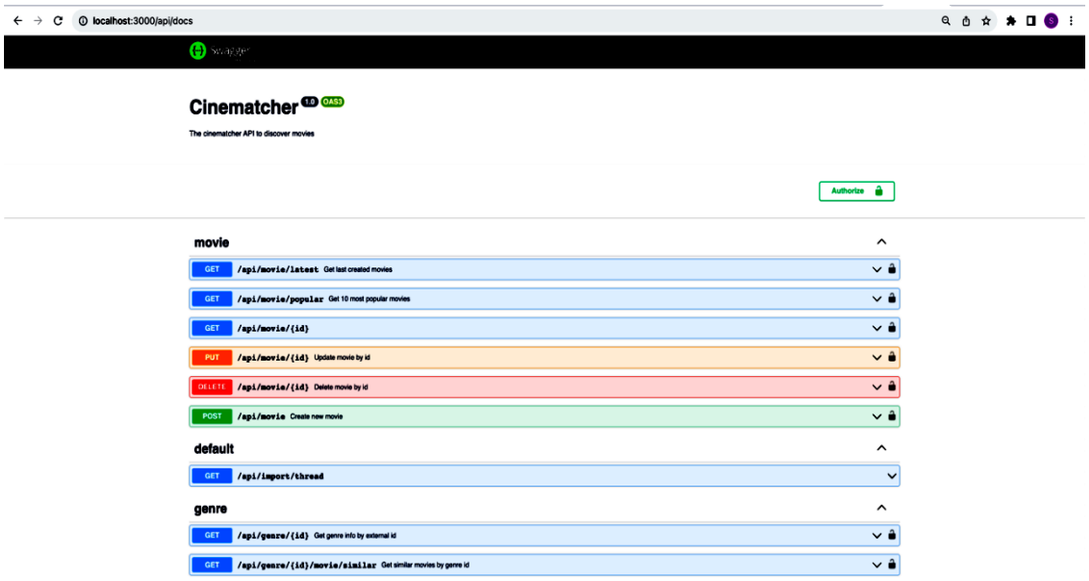

Рисунок 7
Сторінка з документацією для API

INFORMATION AND WEB TECHNOLOGIES

Далі використовується Postman колекція для роботи з API, яка може бути корисною для локальної розробки або для тестування. Postman колекція колекція додається в репозиторій (рис. 8).

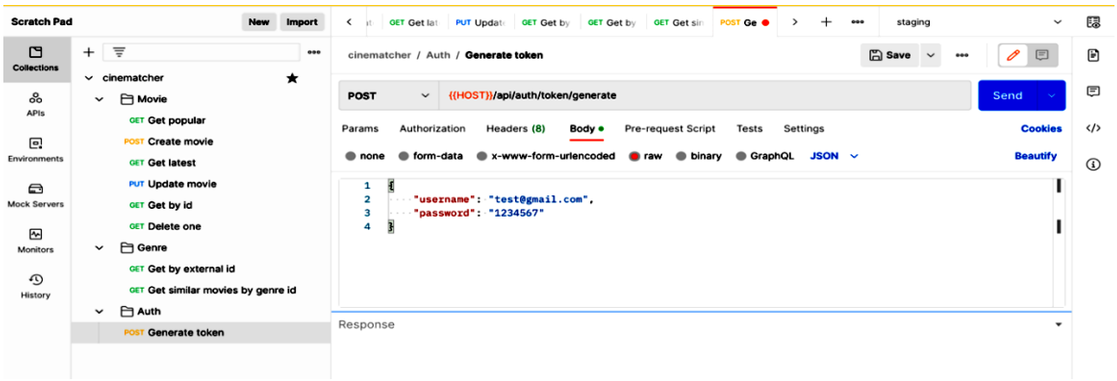

Рисунок 8
Postman колекція

Для програмного застосунку додається Dockerfile, а також docker-compose.yml. Передбачається створення 3-х контейнерів: застосунок, БД MongoDB та інтерфейс адміністратора Mongo Express. Також, створено Makefile для спрощення запуску проєкту та взаємодії з Docker контейнером застосунку. Після логіну, у систему відкривається головна сторінка, яка відображає вітання та список з 10 популярних фільмів. Тобто, застосунок працює, дані створюються після імпорту з стороннього сервісу, система аутентифікації працює.

Для впровадження конвеєру безперервної інтеграції та постачання обрано 2-а підходи: GitLab CI/CD + Heroku та GitHub Actions + AWS EC2. Перш за все необхідно визначити, які етапи мають бути реалізовані для CI/CD. Для цього створюються 5-ть етапів з різною кількість завдань на кожному з них (рис. 9).

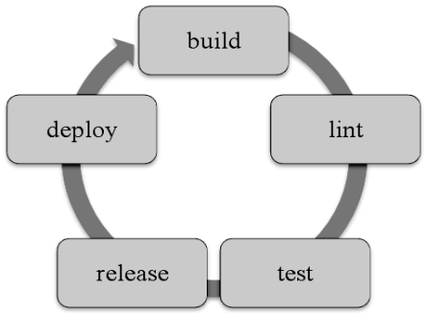

Рисунок 9
Етапи для конвеєру

INFORMATION AND WEB TECHNOLOGIES

Для впровадження CI/CD на GitLab додається файл .gitlab-ci.yml у корені проєкту. У .gitlab-ci.yml файли визначають:

– сценарії, які необхідно запустити;

– інші файли конфігурації та шаблони, які необхідно включити;

– залежності та кеші;

– команди, які потрібно виконувати послідовно, і ті, які потрібно виконувати паралельно;

– місце для розгортання програми.

Сценарії згруповані в завдання, та завдання виконуються як частина більшого конвеєра. Можна групувати кілька незалежних завдань у етапи, які виконуються у визначеному порядку. Для конфігурації CI/CD потрібно принаймні одне завдання, яке не буде прихованим.

Після того, як додали .gitlab-ci.yml файл до свого сховища, GitLab виявляє його, і програма під назвою GitLab Runner запускає сценарії, визначені у завданнях.

Для початку вказується image. Тобто, образ Docker, у якому будуть виконуватися завдання. У якості image використовується node:16.20.2-alpine. Далі визначається stages. Дані етапи використовуються для групування завдань.

Формується порядок команд, які необхідно виконати для створення білду проєкта. Проєкт складається з двох елементів: клієнт (Angular) та сервер (NestJS). Тому, для формування білду проєкту необхідно зробити білд клієнту та розмістити дані файли на сервері. На стороні Angular за директорію, де має розміститись білд відповідає outputPath з файлу angular.json.

Для створення білду клієнта використовується команда: npm run frontend:build, яка запускає команду ng build. В результаті створюється білд та розміщається в папці, яку зазначено у outputPath.

Далі, сервер оберає index.html файл та відображає даний файл у браузері.

Головна мета етапу lint перевірити код на наявність помилок відповідно до встановлених правил лінтера, що дозволяє покращити якість коду (табл. 2).

Таблиця 2

Правила для lint файлу

Правило	Опис
eqeqeq	Вимагає використання === та !==. Вважається хорошою практикою використання типобезпечних операторів рівності === та !== замість їхніх звичайних аналогів == та !=.

INFORMATION AND WEB TECHNOLOGIES

Продовження табл. 2

semi	Вимагає крапки з комою. JavaScript не вимагає крапки з комою в кінці кожного оператора. У багатьох випадках механізм JavaScript може визначити, що крапка з комою має стояти в певному місці, і автоматично додає її. Ця функція відома як автоматична вставка крапки з комою (ASI) і вважається однією з найбільш суперечливих функцій JavaScript.
no-trailing-spaces	Це правило забороняє кінцеві пробіли (пробіли, табуляції та інші пробільні символи Unicode) у кінці рядків.

Додаються файли лінтеру для клієнта та серверу. Для запуску лінтеру використовується команда npm run lint, яка запускає eslint "{src,apps,libs,test}/**/*.ts" на стороні серверу та ng lint на стороні клієнта.

Дані команди беруть правила з зазначених вище файлів та перевіряють код. Формується етап lint та додається задача. Також, використовується before_script для додавання необхідних бібліотек.

На етапі test проводиться тестування коду. Для цього, додається декілька тестів для серверу та клієнту. У якості фреймворку для тестування використовується jasmine. Jasmine – це BDD фреймворк для JavaScript. Він не покладається на браузери, DOM або будь-який фреймворк JavaScript.

Таким чином, він підходить для веб-сайтів, проєктів Node.js або будь-де, де може працювати JavaScript.

Для запуску тестів використовуються команди: npm run backend:test та npm run frontend:test. Додаються дві задачі до етапу test, кожна з яких відповідає за тестування клієнту та серверу відповідно.

Етап release передбачений для створення docker image та розміщення даного image у репозиторії Heroku. Heroku – це платформа як послуга (PaaS) для розгортання, керування та масштабування веб-додатків та сервісів у хмарі. Heroku дозволяє розробникам створювати, використовувати та публікувати додатки в Інтернеті без необхідності керувати інфраструктурою або серверами.

Таким чином, наведений методологічний підхід базується на ітераціях, а не на послідовному процесі, що дає змогу командам DevOps писати код, інтегрувати його, проводити тести, випускати нові версії та впроваджувати зміни у програмне забезпечення у спільному та реальному часі.

Конвеєр повинен забезпечити, що вихідний сигнал завжди

INFORMATION AND WEB TECHNOLOGIES

стабілізується з однаковим входом без коливань під час виконання. Автоматизований процес CI/CD значно покращує процес розробки програмного забезпечення, надаючи командам швидкий зворотний зв'язок щодо їхньої останньої роботи і сприяючи покращенню якості коду та прискоренню доставки.

References:

[1] What Is the MEAN Stack? : [Електронний ресурс]. – Режим доступу : https://www.mongodb.com/mean-stack.
[2] What is Continuous Delivery? : [Електронний ресурс]. – Режим доступу: https://aws.amazon.com/devops/continuous-delivery/.
[3] Best CI/CD Tools for DevOps: A Review of the Top 10 : [Електронний ресурс]. – Режим доступу : https://bluelight.co/blog/best-ci-cd-tools
[4] CALMS Framework for DevOps Transformation : [Електронний ресурс]. – Режим доступу : https://www.xenonstack.com/insights/calms-in-devops.
[5] Continuous Integration : [Електронний ресурс]. – Режим доступу : https://martinfowler.com/articles/continuousIntegration.html.
[6] Третяк, В. Ф., Голубничий, Д. Ю., & Челенко, Ю. В. (2005). Тиражирование данных в системе управления базами данных. Управління розвитком. – Х.: ХНЕУ, (3), 94-95.

SCIENTIFIC EDITION

SCIENTIFIC COLLECTION «INTERCONF»

№ 180 | November, 2023

The issue contains:

Proceedings of the 6th International
Scientific and Practical Conference

SCIENTIFIC PARADIGM IN THE CONTEXT OF TECHNOLOGIES AND SOCIETY DEVELOPMENT

Geneva, Switzerland
26-28.11.2023

All materials are reviewed.
The editorial office did not always agree with the position of authors.

Sighed for online publication: November 28, 2023.

Printed: December 26, 2023. Circulation: 200 copies. Format 60×84/8.
Batang & Courier New typefaces. Offset paper 100gsm. Digital color printing.

Contacts of the editorial office:

LLC Scientific Publishing Center «InterConf»
info@interconf.center
https://www.interconf.center

Certificate on the entry of publishing business subject in the State Register of Publishers, Manufacturers and Distributors of Publishing Products of Ukraine: ДК № 7882 of 10.07.2023.