Vol. 7, No. 1, 2025

Назар Плесканка[1], Мар'яна Плесканка[2], Богдан Тимощук[3]

[1] Кафедра систем автоматизованого проектування, Національний університет "Львівська політехніка", вул. С. Бандери 12, Львів, Україна, E-mail: nazarii.m.pleskanka@lpnu.ua
[2] Кафедра Телекомунікації Національний університет "Львівська політехніка", вул. Професорська, 2, Львів, Україна, E-mail: mariana__p.m.v.9@ukr.net,
[3] Кафедра систем автоматизованого проектування, Національний університет "Львівська політехніка", вул. С. Бандери 12, Львів, Україна, E-mail: bohdan.tymoshchuk.knm.2020@lpnu.ua

АНАЛІЗ ТА РОЗРОБКА ПЛАТФОРМИ ДЛЯ АВТОМАТИЗОВАНОГО РОЗГОРТАННЯ ТА КЕРУВАННЯ СЕРВІСАМИ

Отримано: Березень 12, 2025 / Переглянуто: Березень 26, 2025 / Прийнято: Березень 31, 2025

https://doi.org/

Анотація. Автоматизація розгортання програмного забезпечення відіграє ключову роль у підвищенні ефективності та надійності робочих процесів у сфері розробки та впровадження IT-рішень. Існуючі сервіси, такі як Jenkins, TeamCity мають певні обмеження, зокрема недостатню гнучкість у візуалізації історії змін, обмежені можливості керування параметрами конфігурації та труднощі в координації командної роботи. Вдосконалення цих аспектів може значно підвищити продуктивність команд розробки та прискорити випуск оновлень. Метою дослідження є розробка платформи, яка інтегрується з системами Jenkins та контролю версій GitHub, здійснює запуск розгортання, надає розширені інструменти для візуалізації історії запусків та оновлень, забезпечує повторний запуск зі збереженими параметрами та спільну роботу команди. Проект базується на аналізі сучасних практик CI/CD, управління інфраструктурою як кодом та порівняльній оцінці платформ автоматизації.

Ключові слова: розгортання, інтеграція, автоматизація, CI/CD, Jenkins, програмне забезпечення

Вступ

Сучасний світ стрімко розвивається, та одним із ключових факторів прогресу стають інформаційні технології. Інтернет є невід'ємною частиною не лише бізнесу, але й побуту. У зв'язку з цим з'являються нові веб ресурси у різних галузях людського життя, кожен з яких потребує постійної підтримки, покращення та доповнення функціоналу. Крім того, зростає цінність автоматизації процесів управління, що зменшує людський вплив і, що більш важливо, підвищує ефективність роботи. Одним із цих процесів є автоматизоване розгортання.

Процес розгортання є важливим етапом у життєвому циклі будь-якого застосунку, що включає в себе перетворення програмного коду і потрібних ресурсів в робочий продукт, який в свою чергу розміщується на різних середовищах. Після цього, програмне забезпечення чи додаток готовий до використання користувачами. Важливо зауважити, що розгортання це не просто копіювання коду, а й налаштування середовища, баз даних, підключень, запуск тестів та інших процесів, що забезпечують повноцінну та безперебійну роботу застосунку. Сучасні програмні проекти характеризуються високим рівнем складності та динамічністю. Зростання частоти їх оновлень робить ручне виконання процесу розгортання трудомістким, схильним до помилок та нездатним йти в ногу з потребами сучасного світу. У даній ситуації автоматизація цього процесу стає ключовим фактором, що забезпечує ефективність та надійність розгортання програмного забезпечення та стабільну роботу сервісу.

Постановка проблеми

Перед тим, як зміни в додатку чи сервісі потрапляють до реальних користувачів, вони розгортаються на окремих тестових середовищах. Таким чином, команди розробників мають можливість перевірити зміни в окремому середовищі. Таких середовищ, що в свою чергу можуть містити декілька серверів, створюється декілька, що дозволяє одночасно тестувати різний функціонал. Наявні сервіси надають широкий спектр можливостей для створення і запуску розгортань, проте мають суттєві обмеження в контексті гнучкості візуалізації історії розгортань, повторного використання параметрів та командної взаємодії. Наприклад, відсутність можливості додавати користувацькі поля до історії або групувати дані за ключовими параметрами ускладнює пошук потрібних даних і зв'язків між ними. Це призводить до неефективного використання часу розробників, які змушені вручну агрегувати дані з різних джерел, а також збільшує ризик помилок через відсутність централізованого інтерфейсу для управління процесами. Перезапуск розгортань із такими ж параметрами вимагає повторного ручного введення даних, що знову ж таки збільшує час виконання задач та імовірність появи помилок. Таким чином, проблема полягає не лише в технічних недоліках існуючих інструментів, але й у відсутності екосистеми, яка поєднує автоматизацію, гнучку візуалізацію та інструменти для командної роботи, адаптовані до потреб різних учасників життєвого циклу розробки.

Огляд сучасних джерел інформації за тематикою публікації

Розгортання програмного забезпечення – це комплексний процес, який робить програмну систему готовою до роботи. Він охоплює всі дії, які потрібні для того, щоб застосунок став доступним кінцевим користувачам, гарантував безперебійну роботу та легко оновлювався новим функціоналом [1].

Однією з важливих практик у контексті розгортання програмного забезпечення вважається інфраструктура як код (англ. Infrastructure as Code, IaC) – це підхід, який передбачає ідею використання високорівневої мови програмування для управління інфраструктурою, яка забезпечує роботу програм та сервісів(рис. 1). IaC розглядає інфраструктуру як програмне забезпечення, яке керується на основі системи контролю версій і відповідає тим ж принципам, що і звичайний застосунок [2].

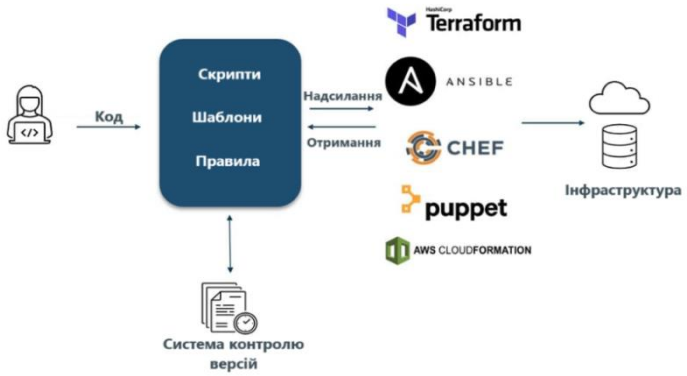

Рис. 1. Принцип роботи підходу IaC

Впровадження IaC дає змогу суттєво прискорити процес створення інфраструктури та внесення будь яких змін. Завдяки автоматизації рутинних завдань, пов'язаних з налаштуванням серверів, мережевих пристроїв та інших компонентів, IaC дозволяє значно зменшити час, необхідний для введення в експлуатацію нових систем або внесення змін до існуючих. IaC дає можливість швидко відновити працездатність систем у разі виникнення проблем, адже забезпечує автоматизовані процедури резервного копіювання та відновлення [3].

Під час розробки програмного забезпечення швидкість та надійність роботи стають ключовими факторами. Саме практики безперервної інтеграції та безперервного розгортання дають можливість скоротити час тестування та розгортання застосунку завдяки автоматизації цих

процесів. Безперервна інтеграція (Continuous Integration – CI) – це одна з DevOps практик розробки програмного забезпечення, що передбачає регулярне об'єднання розробниками змін коду в центральному репозиторії, після чого відбувається автоматична збірка, тестування та запуск. Головною метою є виявити та виправити помилки до того як продукт потрапить в робоче середовище, покращити якість та скоротити час випуску оновлень програмного забезпечення [4,5].

Використовуючи систему контролю версій, наприклад Git, розробники вносять зміни в спільний репозиторій. Перед модифікацією, розробники мають можливість виконати збірку та запуск модульних тестів в локальному середовищі для додаткової перевірки перед інтеграцією. Після того, як зміни коду потрапили в репозиторій сервіс безперервної інтеграції здійснює автоматичну збірку та запуск модульних тестів та повертає результат виконання (рис. 2).

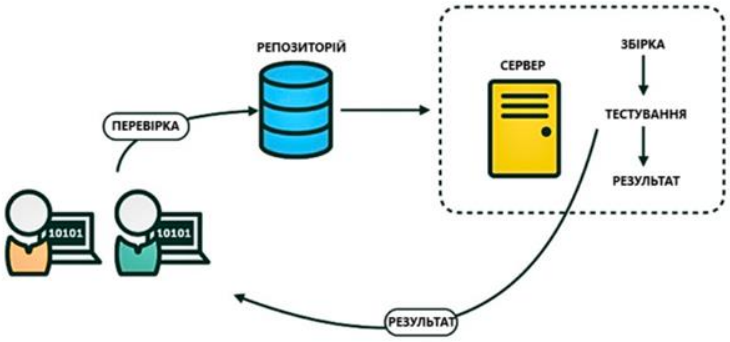

Рис. 2. Принцип роботи безперервної інтеграції

Безперервна доставка (Continuous Delivery або Deployment – CD) починається там, де закінчується безперервна інтеграція. Це ще одна DevOps практика розробки програмного забезпечення, яка використовує автоматизацію для прискореного випуску нового коду. Після всіх необхідних перевірок, таких як функціональне тестування, тестування продуктивності та інтеграції, безперервна доставка розгортає зміни в робочому середовищі [6].

Таким чином, безперервна інтеграція та безперервна доставка формують один повноцінний механізм, який забезпечує процес збірки, тестування та розгортання змін програмного забезпечення. Автоматизація і синхронізація різних етапів значно економить час та ресурси, робить процес випуску програмного забезпечення гнучким, динамічним та надійним.

Цілі та проблеми дослідження

Основна ціль дослідження – розробка платформи для автоматизованого розгортання та керування сервісами, яка пропонує більший набір функціоналу для гнучкої візуалізації та ефективної командної роботи, а саме:

1. Запуск розгортань: платформа інтегрована із сервісом Jenkins та забезпечує синхронізацію даних про розгортання сервісів, що дозволяє розробникам бачити повну картину процесу автоматизованого розгортання, а також здійснювати повторний запуск зі збереженими параметрами.
2. Гнучкість відображення історії розгортання: забезпечення можливості гнучкого налаштовування історії автоматизованих розгортань відповідно до особистих потреб, а також швидкого пошуку інформацію про всі доступні розгортання(деплойменти).
3. Підвищення ефективності командної роботи: платформа дозволить розробникам ділитися історією власних розгортання із іншими членами команди, щоб забезпечити кращу комунікацію та співпрацю.

Виклад основного матеріалу

Існує багато платформ для реалізації безперервної доставки та безперервного розгортання, які дозволяють автоматизувати різні етапи розробки програмного забезпечення та організувати безпечний та стабільний випуск програмних продуктів. Вибір засобів для розгортання залежить від

потреб команди та специфікації програмної системи. До найбільш поширених платформ можна віднести: *Jenkins, TeamCity, Bamboo, Travis CI* .

З урахуванням особливостей функціонування та застосування кожної з представлених платформ представлено порівняльну таблицю, що відображає ключові критерії для вибору оптимального рішення для розгортання, такі як ціна, операційна система, хостинг (On Premise – на власному сервері, Cloud – на хмарному сервері) та кількість компаній, які використовують (табл.1) [7].

Таблиця.1.

Порівняльна таблиця платформ для розгортання

Порівняльний фактор	**Jenkins**	**TeamCity**	**Bamboo**	**Travis CI**
Вартість	Безкоштовно	$299-1999	$10-800	$69-489
Операційна система	Windows, Linux, Unix-like, macOS	Windows, Linux, macOS, Solaris, FreeBSD	Windows, Linux, MacOS, Solaris	Linux, MacOS
Хостинг	On Premise / Cloud	On Premise	On Premise / Bitbucket Cloud	On Premise / Cloud
Кількість компаній	95 615	15 790	2 827	4 316

Перед початком розробки будь-якого програмного продукту необхідно окреслити вимоги до інтерфейсу користувача. Інтерфейс користувача – це комплекс доступних інструментів, які дозволяють користувачеві взаємодіяти з програмним забезпеченням, системою тощо. Для користувачів, система повинна мати інтуїтивно зрозумілий та зручний інтерфейс, адже це одна з ключових частин, із якою безпосередньо взаємодіє користувач.

Для ефективнішого представлення функціональних можливостей платформи для розгортання та управління сервісами, які користувачі можуть використовувати для взаємодії з системою, було розроблено діаграму варіантів використання, яка зображена на рис. 3. Вона відображає можливі сценарії взаємодії з системою, надаючи чітке уявлення про те, як можна маніпулювати функціоналом.

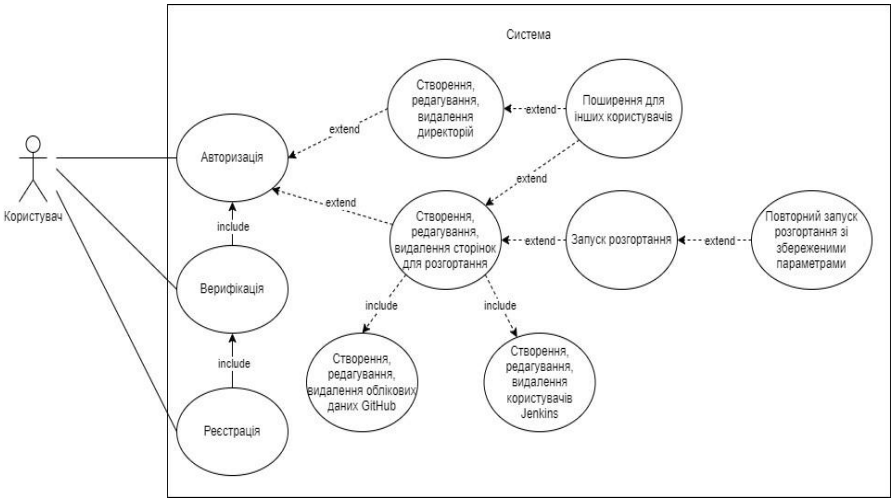

Рис. 3. Діаграма варіантів використання платформи автоматизованого розгортання

Однією з ключових функціональних можливостей платформи є процес запуску розгортання. Він розпочинається з моменту, коли користувач вводить необхідні параметри у відповідні поля та

ініціює процес розгортання сервісу, використовуючи для цього зручний веб інтерфейс. Далі система витягує дані для авторизації в Jenkins та надсилає запит на отримання інформації про останнє розгортання. Після цього надсилається запит на запуск нового розгортання. У випадку, якщо під час запиту виникає помилка, користувач отримує повідомлення з описом проблеми, і процес завершується. Якщо ж помилки не виявлено, система надсилає запит на отримання інформації про останнє розгортання і перевіряє, чи змінилася вона у порівнянні з першим запитом. Якщо інформація останнього розгортання не змінилась, проводиться повторний запит. У разі виявлення змін, запуск розгортання провівся успішно та нова інформація зберігається в базу даних. Процес завершується збереженням оновленої інформації в базу даних, або після надсилання повідомлення про помилку користувачу. На рис 4 представлено блок схему алгоритму запуску розгортання сервісу.

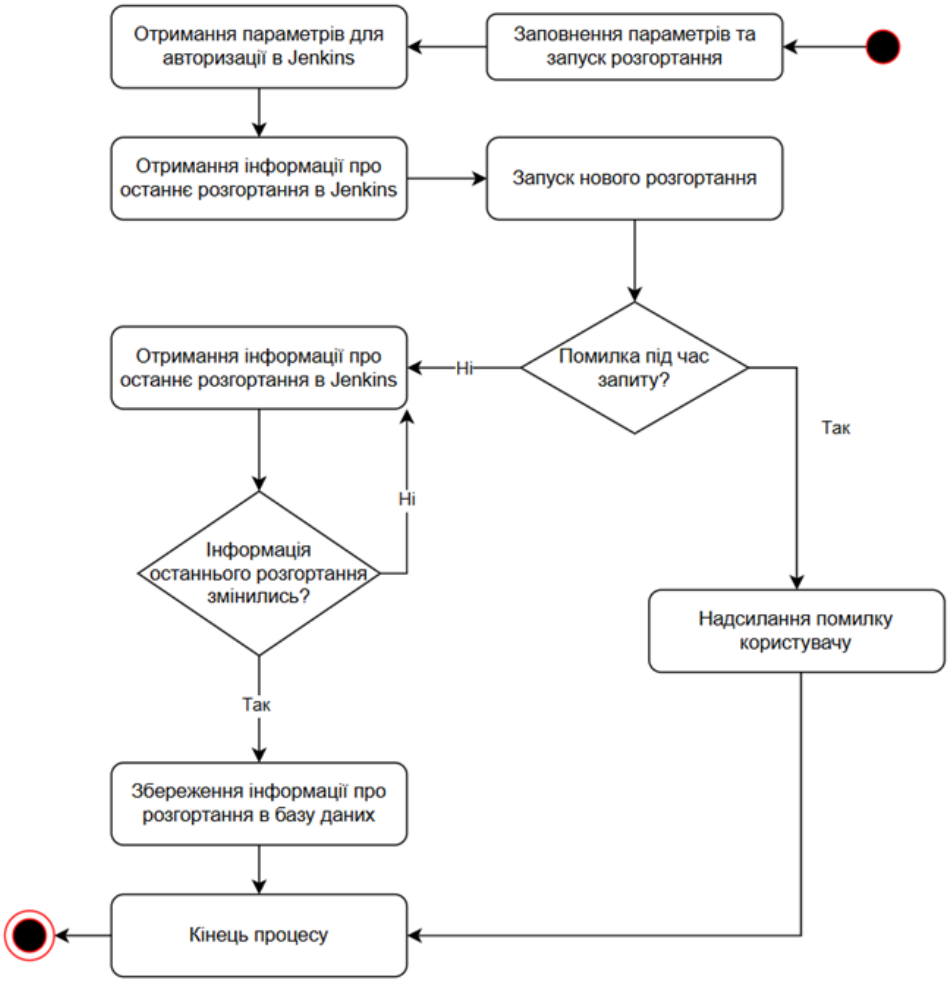

Рис.4. Блок-схема реалізації алгоритму запуску розгортання

Ще одним важливим процесом застосунку являється алгоритм оновлення статусу розгортання, який дозволяє показувати користувачу актуальний статус процесу розгортання в реальному часі. Система отримує список розгортань, які мають статус "у процесі", а потім надсилає запити на оновлення їхніх поточних статусів із Jenkins. У випадку виникнення помилок під час запиту, система надсилає користувачу повідомлення з описом проблеми та завершує процес. Якщо ж помилок не виявлено, система продовжує процес, оновлюючи статуси розгортань у базі даних, а потім на відповідних сторінках розгортання. Процес завершується після успішного оновлення статусів, або після надсилання повідомлення про помилку користувачу. На рис 5 представлено схематичне зображення алгоритму оновлення статусу розгортання.

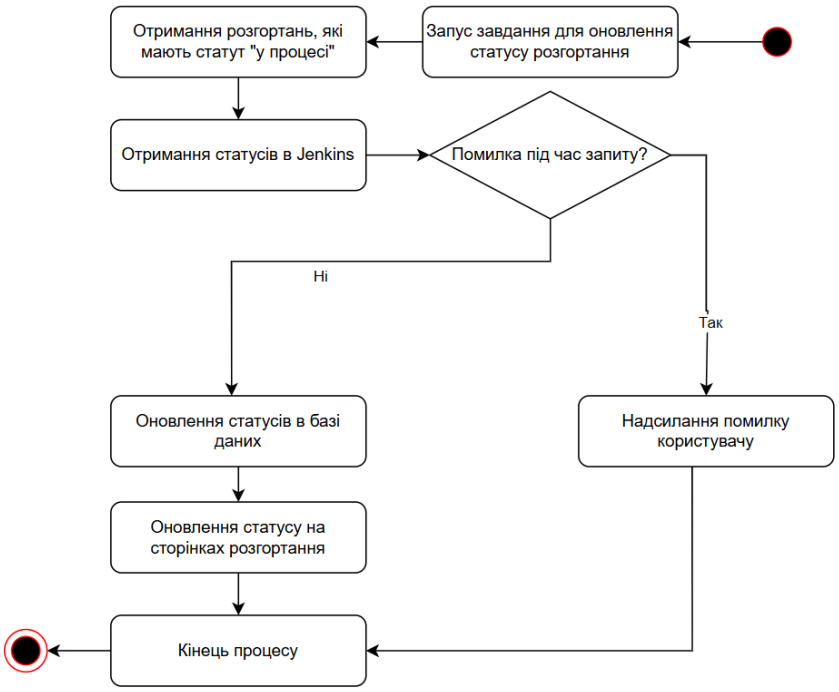

Рис. 5. Блок-схема реалізації алгоритму оновлення статусу розгортання

На рисунку 6 представлена архітектурна схема системи автоматизованого розгортання сервісів, яка забезпечує розуміння логіки та взаємозв'язків всередині застосунку. Ця діаграма демонструє, як компоненти системи взаємодіють між собою та із зовнішніми сервісами, створюючи єдиний функціональний механізм.

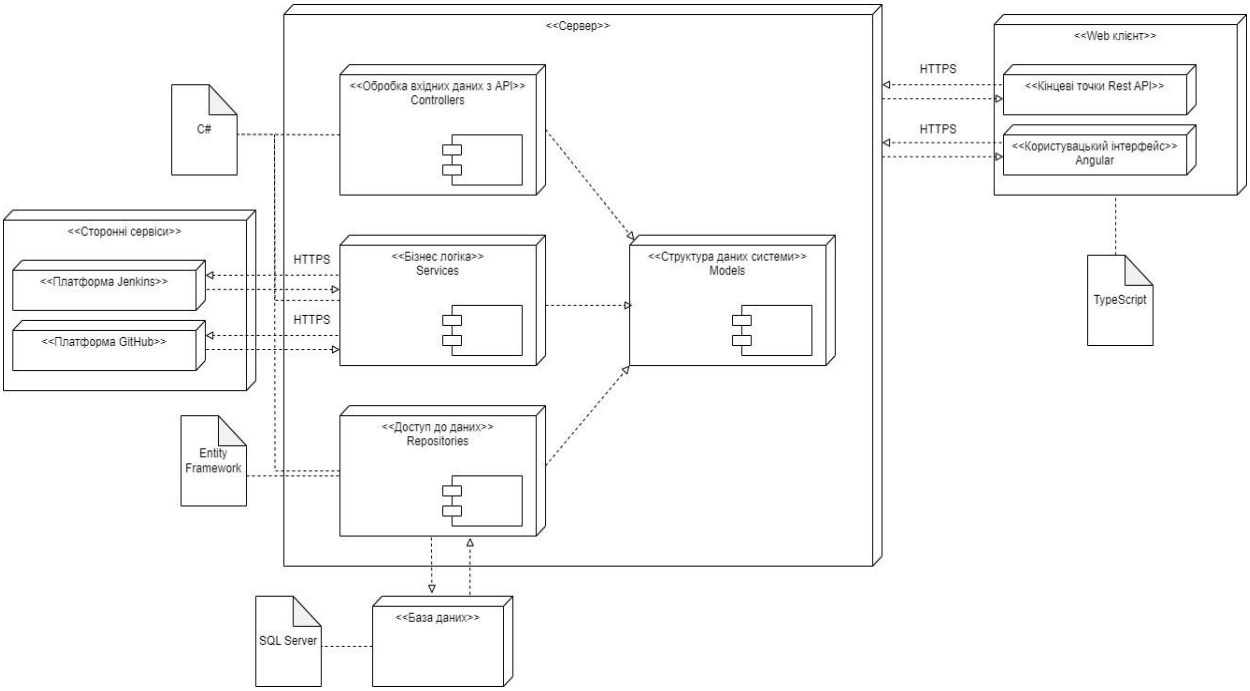

Рис. 6. Діаграма архітектурних компонентів системи

Результати та обговорення

Представлений ресурс, як і більшість сучасних цифрових продуктів, вимагає проходження процесу авторизації для отримання доступу до інтерфейсу управління. Це забезпечує безпеку

системи, захист персональних даних та персоналізований доступ користувача. Авторизація дозволяє ідентифікувати кожного користувача, надаючи йому відповідний рівень доступу до функціоналу, що запобігає несанкціонованому використанню сервісу.

Після успішної авторизації, користувач потрапляє на основну сторінку, яка виконує роль централізованого місця для навігації та управління даними користувача (рис. 7). На цій сторінці відображаються основні директорії із списком сервісів, функціонал розгортання та навігація по всьому продукту, яка також дозволяє користувачам бачити свою поточну позицію в структурі директорій та швидко переходити до попередніх рівнів або інших частин системи.

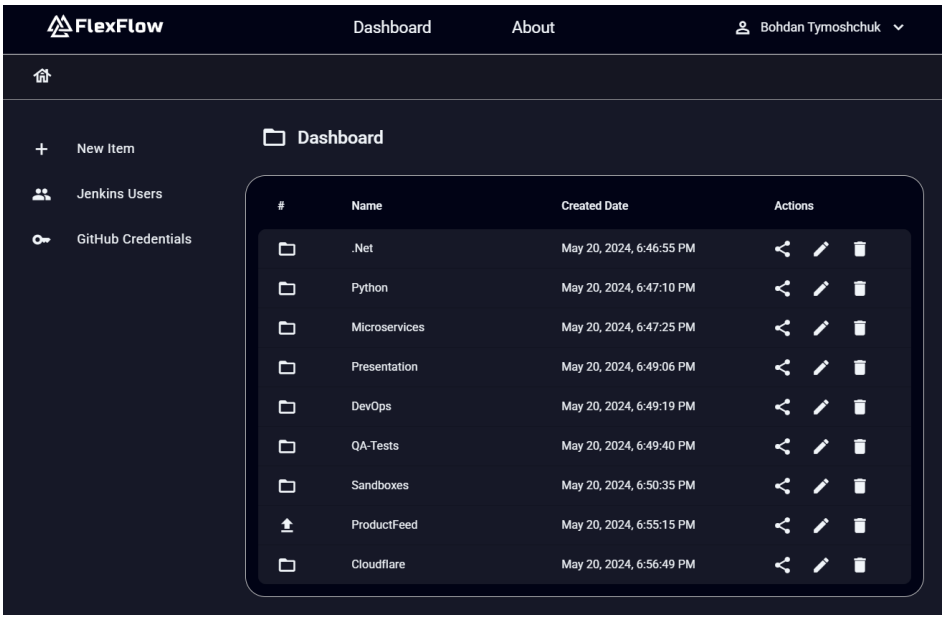

Рис. 7. Сторінка з каталогами сервісів та навігації

Також, оскільки платформа підтримує авторизацію та створення користувачів, також передбачено сторінку керування профілем користувача, яка дозволяє користувачам редагувати свої особисті дані, а саме, ім’я, прізвище, електронну пошту та пароль (рис. 8).

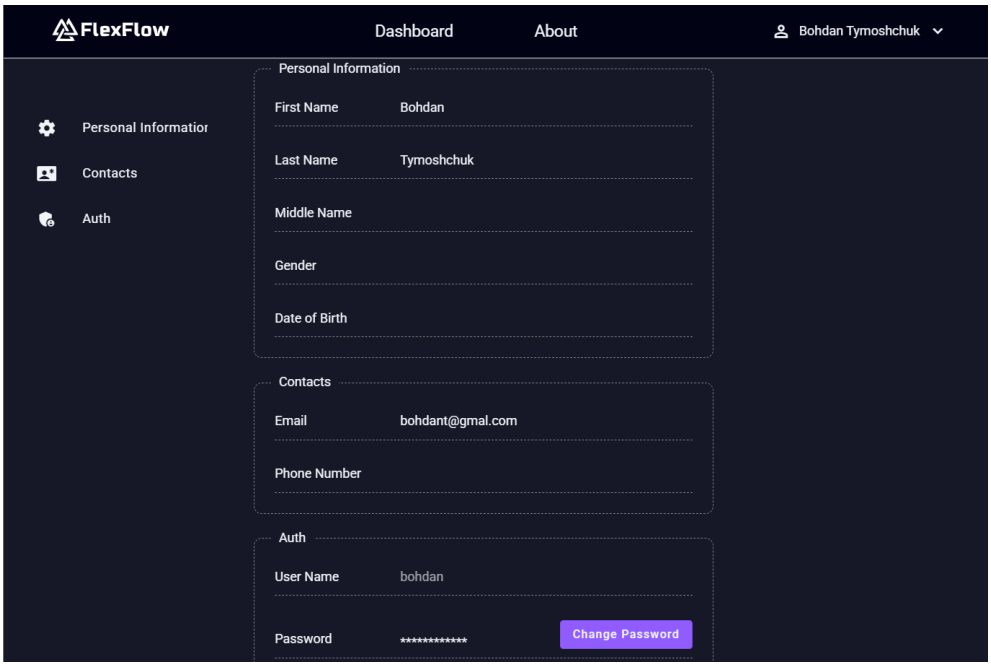

Рис. 8. Сторінка керування профілем користувача

Користувач має одну сторінку, на якій є можливість вибрати що саме він хоче створити: окрему директорію чи сторінку для розгортання. Якщо він обирає створення директорії, йому достатньо ввести її назву та опис. В свою чергу, для створення сторінки розгортання потрібно заповнити наступні поля: назва, опис, користувач Jenkins, посилання на сторінку з розгортанням в Jenkins, дані GitHub та параметри розгортання.

Основний функціонал застосунку розміщений на сторінці для відображення та запуску розгортань (рис. 9). Дані про розгортання відображаються у вигляді таблиці, яка складається з параметрів, створених користувачем. Атрибути, які відображаються на всіх сторінках: час запуску розгортання, статус розгортання та унікальний ідентифікатор, який містить пряме посилання на процес в системі Jenkins. Також є можливість редагування даних та повторного запуску розгортання із збереженими даними.

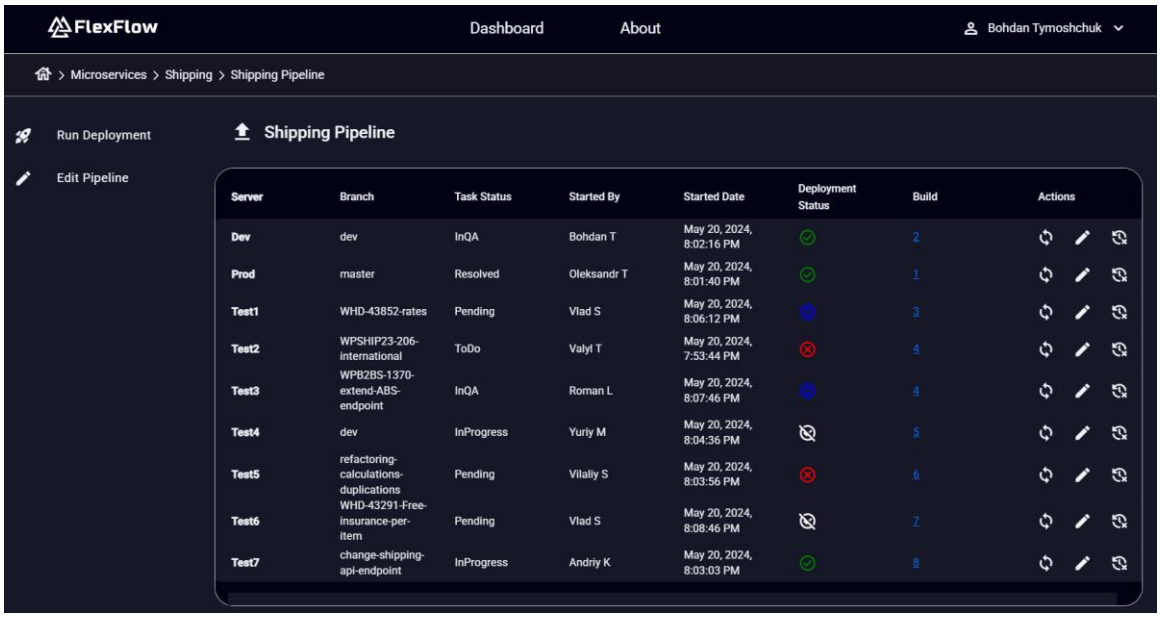

Рис. 9. Сторінка для відображення та запуску розгортань

При запуску розгортання користувачу потрібно ввести параметри, які він вказав при його створенні. Параметри відображаються згідно типу, який обрав користувач, як показано на рис. 10.

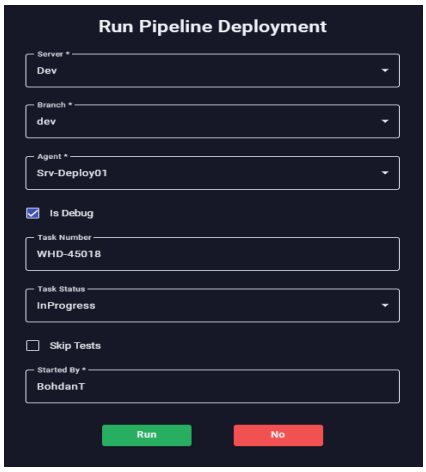

Рис. 10. Вікно для запуску розгортань

Оскільки розроблена платформа передбачає інтеграцію із системами Jenkins та GitHub, відповідно розроблено функціонал для забезпечення можливості керування цими інтеграціями. Такий підхід централізованого управління дозволяє користувачам ефективно контролювати дані,

зменшуючи час на впровадження змін та оновлень, і забезпечує більшу гнучкість та адаптивність системи. Приклад такої сторінки представлено на рисунку 11.

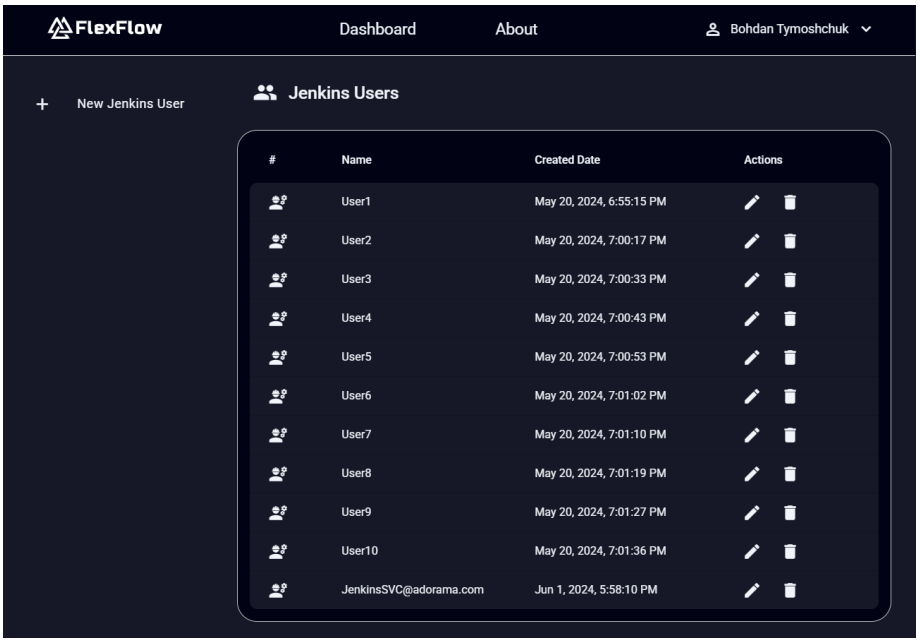

Рис. 11. Сторінка для налаштувань користувачів Jenkins

Подібна сторінка створена і для управління даними GitHub, де відображаються такі дані як назва репозиторію, його власник, а також дані про токен, що забезпечує доступ до нього, та його статус.

Загалом, інтерфейс платформи побудований таким чином, щоб бути уніфікованим, інтуїтивно зрозумілим і зручним у використанні. Його структура мінімізує інформаційне навантаження на користувача, забезпечує швидку навігацію та оперативний доступ до ключових функцій. Завдяки оптимізації API-запитів і використанню механізмів кешування вдалося суттєво скоротити час виконання операцій, що позитивно впливає на загальну продуктивність системи. Перспективи розвитку платформи включають розширення можливостей інтеграції з додатковими CI/CD-системами, що покращить процеси автоматизації розгортання. Також передбачено можливість експорту історії розгортань у формати CSV та JSON, що дозволить зручно аналізувати дані та вести облік змін. Окрім цього, адаптація інтерфейсу для мобільних пристроїв забезпечить комфортну роботу з платформою незалежно від місцезнаходження користувача.

Висновки

В даній роботі проведено дослідження та аналіз розгортання програмного забезпечення і його ключові аспекти. Розглянуто підходи до управління інфраструктурою як кодом та проаналізовано інструменти, що використовуються для їхньої реалізації. Виділено основні аспекти організації безперервної інтеграції та безперервної доставки сервісів. Представлено результати порівняльного аналізу різних платформ для автоматизованого управління програмним забезпеченням.

Розроблено комплексне програмне рішення, яке спрощує роботу з процесами автоматизації та розгортання, а також містить веб інтерфейс користувача, який забезпечує функціонал для запуску і керування процесами розгортання із використанням сервісу Jenkins, та гнучкого відображення історії даних процесів.

Використання розробленої платформи забезпечує гнучкий функціонал для відображення історії розгортань, з можливістю додавати додаткові поля, які не відносяться безпосередньо до самого розгортання, та надають більше уявлення про деталі робочого процесу. Крім того, завдяки інтеграції з системою контролю версій GitHub, користувачі мають змогу синхронізувати роботу з віддаленими репозиторіями, отримуючи доступ до актуальних гілок. Для забезпечення ефективної

командної роботи, застосунок надає інструменти для спільного використання директорій і сторінок розгортання різними користувачами, та можливість повторного запуску з попередньо вказаними параметрами.

Перелік використаних джерел

[1] Розгортання програмного забезпечення. [Електронний ресурс] – Режим доступу до ресурсу: https://www.wikidata.uk-ua.nina.az/Впровадження_програмного_забезпечення.html

[2] Young A. Infrastructure as Code: A Comprehensive Guide to Managing Infrastructure as Code Kindle Edition / Austin Young., 2019. – 118 с.

[3] Dev Ops Culture [Електронний ресурс] : [Веб-сайт]. – Електронні дані. – Режим доступу до ресурсу: https://martinfowler.com/bliki/DevOpsCulture.html

[4] .Fleming S. DevOps And Microservices Handbook: Non-Programmer's Guide to DevOps and Microservices (Continuous Delivery) / Fleming., 2018. – 246 с

[5] Morris K. Infrastructure as Code: Managing Servers in the Cloud. 1st Edition / Kief Morris., 2016. – 362с.

[6] A Schaefer , M Reichenbach: Continuous Integration and Automation for DevOps, 2012. – 345-349 – с. https://doi.org/10.1007/978-94-007-4786-9_28

[7] Continuous Integration: CircleCI vs Travis CI vs Jenkins vs Alternatives [Електронний ресурс] – Режим доступу до ресурсу: https://djangostars.com/blog/continuous-integration-circleci-vs-travisci-vs-jenkins/

Nazar Pleskanka[1], Maryana Pleskanka[2], Bogdan Tymoshchuk[3]

[1] Computer Design Systems Department , Lviv Polytechnic National University
St. S. Bandery 12, Lviv, Ukraine, E-mail: nazarii.m.pleskanka@lpnu.ua
[2] Telecommunication Department , Lviv Polytechnic National University
St. S. Bandery, Lviv, Ukraine, E-mail: mariana__p.m.v.9@ukr.net,
[3] Computer Design Systems Department , Lviv Polytechnic National University
St. S. Bandery 12, Lviv, Ukraine, E-mail: bohdan.tymoshchuk.knm.2020@lpnu.ua

ANALYSIS AND DEVELOPMENT OF A PLATFORM FOR AUTOMATED SERVICES DEPLOYMENT AND MANAGEMENT

Received: March 12, 2025 / Revised: March 26, 2025 / Accepted: March 31, 2025

Abstract. Software deployment automation plays a key role in improving the efficiency and reliability of workflows in the field of development and implementation of IT solutions. Existing services such as Jenkins, TeamCity have certain limitations, including insufficient flexibility in visualizing the history of changes, limited capabilities for managing configuration parameters, and difficulties in coordinating teamwork. Improving these aspects can significantly increase the productivity of development teams and accelerate the release of updates. The goal of the research is to develop a platform that integrates with Jenkins and GitHub version control systems, performs deployment launches, provides advanced tools for visualizing the history of launches and updates, provides relaunch with saved parameters, and team collaboration. The project is based on the analysis of modern CI/CD practices, infrastructure as code management, and comparative evaluation of automation platforms.

Keywords: deployment, integration, automation, CI/CD, Jenkins, software.