ІНТЕГРАЦІЯ СУЧАСНИХ ТЕХНОЛОГІЙ ШТУЧНОГО ІНТЕЛЕКТУ В ПРОЦЕСИ БЕЗПЕРЕРВНОЇ ІНТЕГРАЦІЇ ТА РОЗГОРТАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Адріан Наконечний,д.т.н., професор, Михаїл-Мішель Вигриновський, аспірант
Національний університет «Львівська політехніка», Україна
e-mail: mykhail-mishel.a.vyhrynovskyi@lpnu.ua

Анотація

У статті розглядаються сучасні підходи до організації процесів безперервної інтеграції (CI) та безперервного розгортання (CD) у розробці програмного забезпечення з використанням технологій штучного інтелекту (ШІ). Аналізується історичний розвиток CI/CD, їх роль у забезпеченні високої якості ПЗ, основні переваги та недоліки традиційних підходів, а також перспективи інтеграції AI-технологій для автоматизації та оптимізації даних процесів. Результати дослідження демонструють, що комплексне впровадження систем CI/CD із застосуванням ШІ сприяє скороченню циклів розробки, підвищенню стабільності систем та оптимізації використання ресурсів.

Ключові слова

CI/CD, безперервна інтеграція, безперервне розгортання, штучний інтелект, автоматизація, оптимізація, DevOps.

1. Вступ

За останні десятиліття методи розробки програмного забезпечення зазнали кардинальних змін. Традиційні каскадні моделі поступово відходять на задній план, а сучасні підходи, зокрема безперервна інтеграція (CI, Continuous Integration) та безперервне розгортання (CD, Continuous Delivery), стають основою ефективної роботи розробницьких команд. Сучасна модель розробки, заснована на принципах DevOps (development і operations), забезпечує тісну інтеграцію між командами розробників та експлуатаційними спеціалістами, що дозволяє оперативно впроваджувати нові функціональні можливості та підтримувати стабільну роботу застосунків у продуктивному середовищі. Сучасні технології штучного інтелекту відкривають додаткові можливості для оптимізації процесів CI/CD, що дозволяє досягати високої якості кінцевого продукту [1-6]. Ця стаття надає аналіз сучасних підходів до впровадження CI/CD, визначає їхні переваги і недоліки, а також розглядає перспективи застосування штучного інтелекту (ШІ) для автоматизації та оптимізації цих процесів, з акцентом на практичних технологіях та прикладах їх використання.

2. Недоліки

Незважаючи на численні переваги впровадження CI/CD, існують певні виклики, що ускладнюють їх повноцінне застосування у великих інформаційних системах:

1. Інтеграція новітніх технологій.
2. Впровадження інноваційних рішень, зокрема ШІ-технологій, потребує значних фінансових та кадрових ресурсів, а також модернізації існуючої інфраструктури. Це може стати серйозною перешкодою для малих і середніх компаній.
3. Обробка великих обсягів даних.
4. Прогнозування дефектів та оптимізація тестових сценаріїв вимагають аналізу великих обсягів історичних даних, що пов’язано з високими вимогами до обчислювальної потужності та застосуванням спеціалізованих алгоритмів.
5. Недостатня стандартизація. Різноманіття підходів і рішень, що використовуються в різних компаніях, створює труднощі для впровадження єдиних стандартів, що ускладнює інтеграцію інноваційних технологій у існуючі CI/CD-конвеєри. Для подолання зазначених викликів необхідно розробляти методології, що забезпечують інтеграцію новітніх технологій у традиційні системи CI/CD, оптимізувати методи обробки даних і впровадження стандартів [1,2].

3. Мета

Основною метою даного дослідження є розробка та обґрунтування методології інтеграції технологій штучного інтелекту у процеси безперервної інтеграції та розгортання, що дозволить:

підвищити ефективність і стабільність процесів розробки програмного забезпечення; скоротити час

реакції на виникнення дефектів завдяки автоматизованому прогнозуванню та оптимізації тестових сценаріїв. Оптимізувати використання ресурсів системи шляхом адаптивного управління навантаженням у режимі реального часу. Забезпечити інтеграцію різноманітних інструментів автоматизації в єдину екосистему за допомогою сучасних ШІ-технологій.

4. Дослідження сучасних методів автоматизації та оптимізації процесів CI/CD із застосуванням штучного інтелекту

У сучасному IT-середовищі безперервна інтеграція та розгортання (CI/CD) стали ключовими складовими життєвого циклу розробки програмного забезпечення, що дозволяє командам оперативно впроваджувати інновації та підтримувати високий рівень якості продуктів. Традиційні методи розробки поступово відходять на другий план, а сучасні підходи, базовані на принципах DevOps, дозволяють створити безперебійні конвеєри збірки, тестування та розгортання.

Однак існує значна кількість викликів, пов'язаних із впровадженням CI/CD у великих системах: потреба у значних фінансових і кадрових ресурсах для інтеграції новітніх технологій, обробка великих обсягів історичних даних із високими вимогами до обчислювальної потужності та нестача уніфікованих стандартів для впровадження інноваційних рішень.

Використання сучасних DevOps-підходів сприяє глибокій автоматизації, а системи CI/CD можуть бути значно покращені за допомогою інтеграції ШІ-технологій. Аналіз передових DevOps-практик підтверджує необхідність удосконалення процесів шляхом впровадження штучного інтелекту, що підвищує ефективність та надійність конвеєрів CI/CD [7].

Подальші дослідження підкреслюють, як штучний інтелект може посилити безпеку та надійність CI/CD за допомогою виявлення аномалій та автоматизованих реакцій на інциденти [8].

Інтеграція ШІ у CI/CD-конвеєри відкриває такі перспективи: алгоритми, такі як рекурентні нейронні мережі, можуть аналізувати історичні дані змін у коді та результати тестування, формуючи модель прогнозування дефектів, що математично описується як $P(\text{дефект}) = f(x_1, x_2, \ldots, x_n)$, де $x_1, x_2, \ldots, x_n$ являють собою характеристики змін, а $f(\cdot)$ – функція, побудована на основі машинного навчання [9]. Такий підхід дозволяє заздалегідь визначати потенційні проблемні ділянки, що оптимізує розподіл ресурсів і скорочує час усунення дефектів. Крім того, оптимізація тестових сценаріїв за допомогою ШІ з використанням методів пріоритезації тестових випадків на основі машинного навчання значно прискорює процес перевірки критичних компонентів застосунків. Нарешті, оптимізація безперервного розгортання з використанням методів машинного навчання, зокрема, алгоритмів підкріплюючого навчання для адаптивного управління розгортанням, демонструє потенціал застосування ШІ для покращення стабільності і якості кінцевого продукту [10].

Контейнеризація, яку забезпечує інструментарій для управління ізольованими Linux-контейнерами Docker, є фундаментальною для сучасних CI/CD-конвеєрів. Завдяки Docker можна створювати єдині, відтворювані образи застосунків, що містять усі необхідні залежності, забезпечуючи сумісність середовищ на всіх етапах розробки. Dockerfile визначає базовий образ, встановлює робочу директорію, копіює файли залежностей та вихідний код застосунку, встановлює залежності, відкриває порт та запускає застосунок. Це дозволяє ефективно інтегрувати ШІ-модулі, які аналізують результати тестування в контейнерах та дають рекомендації щодо оптимізації конфігурацій [12].

Для управління великою кількістю контейнерів використовується відкрита система Kubernetes, яка забезпечує автоматичне масштабування та відмовостійкість застосунків. Завдяки стратегії розгортання rolling updates Kubernetes дозволяє здійснювати оновлення без зупинки роботи системи. Інтеграція ШІ з Kubernetes, зокрема через аналіз даних моніторингу, дозволяє оптимізувати налаштування масштабування та підвищити ефективність управління ресурсами. Додатково, ArgoCD як сучасний інструмент декларативного розгортання забезпечує синхронізацію фактичного стану застосунків із заданим конфігураційним описом у YAML-файлах. Використання ШІ для аналізу відхилень у стані системи допомагає оперативно реагувати на потенційні проблеми та забезпечує постійну відповідність до бажаного стану [13].

Для автоматизації процесів CI/CD часто використовують системи оркестрації, такі як Jenkins. Конфігурація детальньної послідовності (Pipeline) в Jenkins може бути описана за допомогою Jenkinsfile, який зберігається у репозиторії коду. Jenkinsfile визначає Pipeline з трьома стадіями: Build, Test та Deploy. На стадії Build виконується збірка проєкту за допомогою інструменту для автоматизації процесу зборки проєктів Maven, на стадії Test запускаються юніт-тести, а на стадії Deploy відбувається розгортання застосунку в Kubernetes через командний рядок kubectl. Хмарна інтеграція через Amazon Web Services (AWS) дає можливість реалізувати високоефективні CI/CD-конвеєри з автоматичним масштабуванням і високою безпекою. Сервіси AWS, такі як CodePipeline, CodeBuild та CodeDeploy, дозволяють автоматизувати весь життєвий цикл розробки, а сервіси ECS і EKS спрощують управління контейнерами у хмарному середовищі. Інтеграція ШІ з AWS, що забезпечує аналіз логів і прогнозування навантаження,

сприяє оптимальному розподілу ресурсів і підвищенню стабільності систем, що є критичним для підтримки високої якості застосунків [14-15].

У рамках дослідження автоматизації процесів CI/CD із застосуванням штучного інтелекту було розроблено архітектуру (рис. 1), яка ілюструє взаємодію між основними компонентами розгортання програмного забезпечення. Дана архітектура відображає розширену CI/CD-інфраструктуру, що включає інтеграцію ArgoCD, Docker Artifactory та ШІ-аналітики, що дозволяє забезпечити безперервний процес оновлення та оптимізації.

ArgoCD як ключовий елемент GitOps-розгортання.

ArgoCD виступає критичним компонентом у підході GitOps, який забезпечує автоматичну синхронізацію стану Kubernetes-кластерів із заданими конфігураціями. На відміну від традиційних методів розгортання, ArgoCD зменшує потребу у ручному втручанні та мінімізує ризики невідповідності версій застосунків. Основні можливості, які надає ArgoCD:

- Автоматизоване управління станом застосунку - порівняння реального стану кластеру з бажаним, визначеним у Git-репозиторії.
- Зворотне відновлення (rollback) - у разі виявлення некоректних змін ArgoCD автоматично повертає систему до стабільного стану.
- Поліпшена безпека та контроль доступу - інтеграція з RBAC (Role-Based Access Control) дозволяє гнучко налаштовувати права користувачів.
- Docker Artifactory для управління контейнерними образами
- Docker Artifactory використовується як централізоване сховище для контейнерних образів, що проходять через конвеєр CI/CD. Основні переваги цього підходу:
- Версіонування та контроль змін - Artifactory дозволяє відстежувати історію змін контейнерів та підтримувати різні версії.
- Кешування та оптимізація продуктивності - повторне використання образів зменшує навантаження на CI/CD-систему.
- Інтеграція з політиками безпеки - забезпечує автоматичну перевірку вразливостей у контейнерних образах перед їх розгортанням.

ШІ-аналітика у CI/CD.

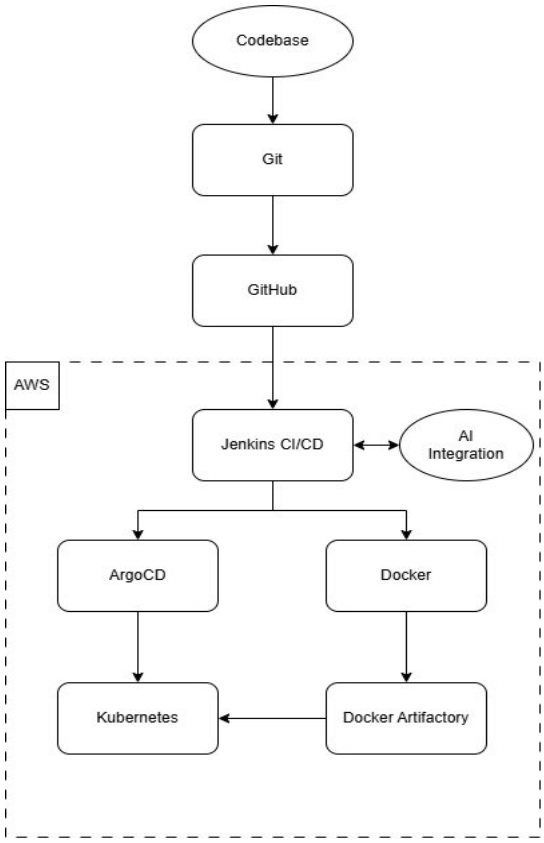

Рис. 1. Архітектурна схема взаємодії між основними компонентами розгортання програмного забезпечення

На структурі передбачена система ШІ Integration, яка підключається до Jenkins CI/CD для аналізу процесів

розгортання, що дозволяє:

- Прогнозувати можливі збої на основі аналізу історичних даних про невдалі деплої.
- Оптимізувати ресурси шляхом динамічного підлаштування конфігурацій контейнерів відповідно до реального навантаження.
- Автоматично аналізувати помилки тестів та пропонувати шляхи їх виправлення.

Таким чином, запропонована архітектура демонструє вдосконалений підхід до DevOps-автоматизації, де поєднання ArgoCD, Docker Artifactory та AI-аналітики створює ефективний, адаптивний і безпечний конвеєр безперервного розгортання програмного забезпечення.

У межах дослідження оптимізації CI/CD-конвеєрів було розроблено архітектурну схему (рис.2), що описує Pipeline автоматизованого збирання, тестування та розгортання програмного забезпечення.

Опис архітектури.

Дана схема демонструє повний цикл обробки коду в автоматизованому Pipeline CI/CD, починаючи з фіксації змін у систему керування версіями (SCM) і закінчуючи розгортанням нової версії продукту на сервер (deploy) та повідомленням про статус виконання.

Основні етапи Pipeline:

1. Фіксація змін (SCM Commit) у системі контролю версій. Автоматичний процес розпочинається після того, як розробник здійснює фіксація змін у систему контролю версій. Вибір гілки визначає цільове середовище розгортання.
2. Автентифікація (Authentication). Для забезпечення безпеки здійснюється перевірка користувача та авторизація дій перед виконанням наступних етапів.
3. Завантаження та встановлення залежностей (Dependency Download/Install). На цьому етапі виконується завантаження необхідних бібліотек і модулів, а також перевірка безпеки залежностей (Dependency Security Scan), що допомагає виявляти потенційні вразливості.
4. Клонування вихідного коду (SCM Checkout). Забезпечує отримання актуального стану репозиторію для подальшої обробки.
5. Статичний аналіз коду (Static Code Analysis). Виявляє помилки без необхідності виконання програми, аналізуючи структуру та можливі уразливості.
6. Фаза збирання (Build). Під час побудови проєкту може відбуватися паралельна компіляція та обробка декількох незалежних модулів (Build Module 1, Build Module N), що прискорює процес.
7. Збереження артефактів (Artifact Storage). Побудовані артефакти зберігаються для подальшого використання на наступних етапах, включаючи розгортання.
8. Автоматизоване тестування (Test). Включає виконання різних типів тестів у паралельному режимі для оптимізації часу:
   - Інтеграційні тести (Integration Test) перевіряють взаємодію між модулями.
   - Регресійні тести (Regression Test) виявляють вплив нових змін на існуючий функціонал.
   - Smoke-тести (Smoke Test) перевіряють базову працездатність додатку.
   - Тести доступності (Accessibility Test) забезпечують відповідність стандартам WCAG.
   - Тестування продуктивності (Performance Test) аналізує швидкодію системи.

   Результати тестування формуються у звіт (Tests Report).
9. Перевірка Docker-образів (Docker Image Scan). Автоматичний аналіз контейнерних образів для виявлення потенційних загроз безпеки.
10. Розгортання (Deploy). Відбувається доставка оновлених сервісів у цільове середовище.
11. Пост-деплойні тести та моніторинг (Post-Deployment Tests/Monitoring). Аналізують продуктивність, збирають метрики та перевіряють стабільність оновлень.
12. Оповіщення та звіти (Notifications, Build Report). Завершальний етап включає генерацію звітів про виконання Pipeline та повідомлення команд про статус.
13. Завершення Pipeline (End of Workflow). Остання точка процесу, яка означає успішне або неуспішне завершення розгортання.

Ця архітектурна схема демонструє сучасний підхід до автоматизації CI/CD-процесів із наголосом на оптимізацію часу, підвищення надійності коду та покращення якості релізів. Поєднання паралельного виконання тестів, автоматичного сканування безпеки та ефективного збереження артефактів створює гнучку та масштабовану систему безперервного розгортання.

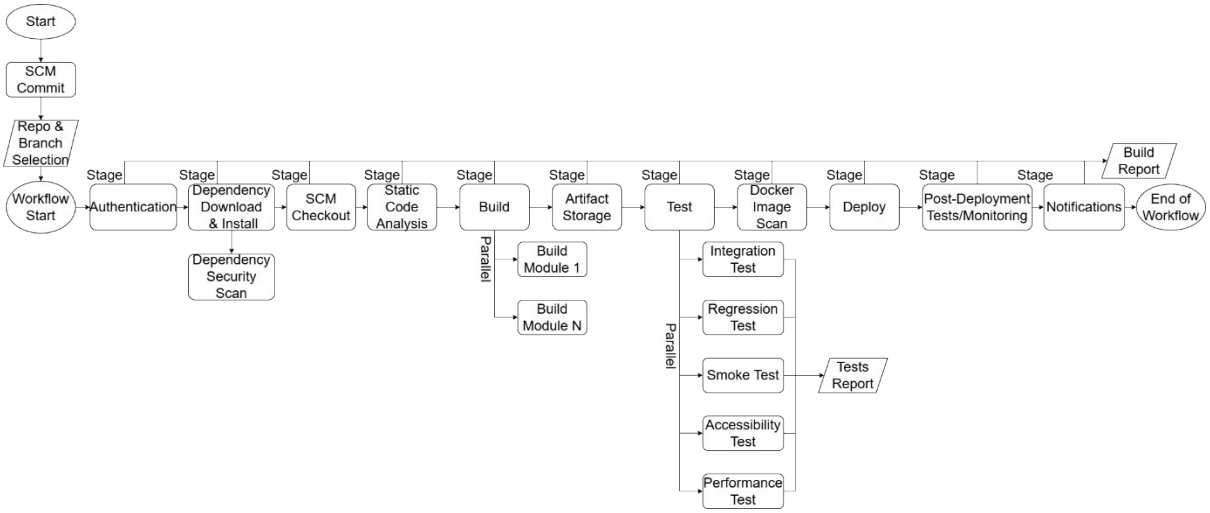

Рис.2. Архітектурна схема CI/CD-пайплайну з автоматизованим збиранням, тестуванням і розгортанням ПЗ

У підсумку комплексна інтеграція сучасних CI/CD-систем із технологіями Docker, Kubernetes, ArgoCD та AWS, доповнена можливостями штучного інтелекту, створює потужну екосистему, здатну забезпечити високоефективну автоматизацію, оптимізацію процесів тестування і розгортання, а також адаптивне управління ресурсами. Цей підхід не лише сприяє зниженню операційних витрат, але й дозволяє своєчасно виявляти дефекти та оперативно реагувати на зміни у навантаженні, що є ключовим фактором для успіху сучасних IT-систем.

5. Висновки

Проведене дослідження демонструє, що інтеграція систем безперервної інтеграції та розгортання (CI/CD) із сучасними технологіями, такими як Docker, Kubernetes, ArgoCD та Amazon Web Services, у поєднанні з алгоритмами штучного інтелекту створює ефективну, відтворювану та масштабовану екосистему розробки програмного забезпечення. Комплексний підхід дозволяє оптимізувати процеси розробки за рахунок завчасного прогнозування дефектів та оптимізації тестових сценаріїв, що знижує час на усунення помилок і підвищує якість кінцевого продукту. Автоматичне масштабування та адаптивне управління ресурсами, реалізоване через інтегровані моніторингові системи в Kubernetes та AWS, забезпечує стабільну роботу застосунків навіть у пікові періоди навантаження. Подальші дослідження, спрямовані на розробку ефективніших моделей машинного навчання та уніфікацію стандартів інтеграції AI-технологій у CI/CD, відкривають додаткові можливості для оптимізації розробницьких процесів, що є ключовим для забезпечення високої ефективності, якості та адаптивності сучасних IT-систем. Впровадження описаних підходів на практиці може значно покращити процеси розробки та розгортання програмного забезпечення.

Подяка

Автори висловлюють щиру подяку керівнику дисертаційних досліджень за цінні рекомендації та підтримку протягом усього процесу дослідження. Особлива подяка також адресується співробітникам кафедри комп’ютерних систем автоматики за надану допомогу та консультації.

Конфлікт інтересів

Автори заявляють, що конфлікт інтересів відсутній.

Список використаних літературних джерел

[1] S. Pattanayak, P. Murthy, A. Mehra. Integrating AI into DevOps pipelines: Continuous integration, continuous delivery, and automation in infrastructural management. International Journal of Science and Research, Vol. 13(01), 2024, pp. 2244-2256. doi:10.30574/ijsra.2024.13.1.1838
[2] Yara Maha Dolla Ali. Autonomous systems: Challenges and opportunities. Advances in Engineering Innovation. AEI, Vol.4, 2023 pp. 38-42 doi: 10.54254/2977-3903/4/2023031

[3] Venkata Mohit Tamanampudi. AI-Augmented Continuous Integration for Dynamic Resource Allocation. World Journal of Advanced Engineering Technology and Sciences, Vol. 13(01), 2024 pp. 355-368 doi:10.30574/wjaets.2024.13.1.0425
[4] Osinaka Chukwu Desmond. AI-Powered DevOps: Leveraging machine intelligence for seamless CI/CD and infrastructure optimization. International Journal of Science and Research Archive, Vol. 06(02), 2022, pp. 94-107, doi: 10.30574/ijsra.2022.6.2.0150
[5] M. Steidl, M. Felderer, R. Ramler. The pipeline for the continuous development of artificial intelligence models-Current state of research and practice. Journal of Systems and Software, Vol. 199, 2023, 111615, doi.org/10.1016/j.jss.2023.111615
[6] Yue Zhou, Yue Yu, Bo Ding. Towards MLOps: A case study of ML pipeline platform. International Conference on Artificial Intelligence and Computer Engineering, ICAICE, IEEE (2020), pp. 494-500, doi:10.1109/ICAICE51518.2020.00102
[7] Aliyu Enemosah. Enhancing DevOps Efficiency through AI-Driven Predictive Models for Continuous Integration and Deployment Pipelines. International Journal of Research Publication and Reviews, Vol 6(1), 2025, pp. 871-887, doi:10.55248/gengpi.6.0125.0229
[8] Abdul Sajid Mohammed, Venkata Ramana Saddi, Santhosh Kumar Gopal, Dhanasekaran Selvaraj. AI-Driven Continuous Integration and Continuous Deployment in Software Engineering. 2nd International Conference on Disruptive Technologies (ICDT), 2024. pp. 531-536, doi:10.1109/ICDT61202.2024.10489475
[9] Kim, G., Humble, J., Debois, P., Willis, J. The DevOps Handbook: How to Create World-Class Agility, Reliability, and Security in Technology Organizations. Portland: Publisher IT revolution press, Book: 2016. – 480 p. URL: https://www.amazon.com/DevOps-Handbook-World-Class-Reliability-Organizations/dp/1942788002
[10] B. S. Satapathy, S. S. Satapathy, S. I. Singh, and J. Chakraborty, "Continuous Integration and Continuous Deployment (CI/CD) Pipeline for the SaaS Documentation Delivery," in International Conference on Information Technology, Singapore, 2023, pp. 41-50, doi.org/10.1007/978-981-99-5994-5_5
[11] N. Poulton. The Kubernetes Book: 2023 Edition. – JJNP Consulting Limited, 2022. – 3rd edition. – 355 p.
[12]. N. Poulton. Getting Started with Docker: Master the Art of Containerization with Docker. Book: 2024, Edition. – Packt Publishing Limited – 116 p.doi.org/10.0000/9781835462058-001
[13] Hussain, M. Application of Docker and Kubernetes in large-scale cloud environments. International Journal of Engineering Research & Technology, Vol 12(5), 2023. pp. 450–457,https://doi.org/10.5281/zenodo.8135267
[14] Kubernetes Documentation. Horizontal Pod Autoscaler [Електронний ресурс]. – https://kubernetes.io/docs/tasks/run-application/horizontal-pod-autoscale/ (дата звернення: 28.03.2025).
[15] Sandeep Pochu, Sai Rama Krishna Nersu, Srikanth Reddy Kathram. Scaling Kubernetes Clusters with AI-Driven Observability for Improved Service Reliability. Journal of AI-Powered Medical Innovations. Vol. 3(1), 2024. pp. 39-52, doi.org/10.60087/Japmi.Vol.03.Issue.01.Id.003

Наконечний Адріан Йосифович
доктор технічних наук , професор
Кафедра «Комп'ютеризованих систем автоматики»
Національний університет «Львівська політехніка»
вул. С.Бандери, 12, м.Львів, Україна,.79013
E-mail: Adrian.Y.Nakonechnyi@lpnu.ua
Контактний телефон: +38 067 704 75 95
Кількість статтей у загальнодержавних базах даних – 123
Кількість статтей у міжнародих базах даних – 39
Номер ORCID: https://orcid.org/0000-0002-1873-6337

Вигриновський Михаїл-Мішель Андрійович
аспірант
Кафедра «Комп'ютеризованих систем автоматики»
Національний університет «Львівська політехніка»
вул. С. Бандери, 12, м. Львів, Україна, 79013
E-mail: nazarkulik99@gmail.com
Контактний тел.: +38 097 301 80 08
Кількість статей у загальнодержавних базах даних – 4
Кількість статей у міжнародних базах даних – -

Номер ORCID: -https://orcid.org/0009-0008-3357-3418