Національний технічний університет України
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»

Кафедра інженерії
програмного забезпечення
в енергетиці (ІПЗЕ)

Процеси неперервної інтеграції і деплойменту

Робоча програма навчальної дисципліни (Силабус)

Реквізити навчальної дисципліни

Рівень вищої освіти	**Перший (бакалаврський)**
Галузь знань	12 Інформаційні технології
Спеціальність	121 Інженерія програмного забезпечення
Освітня програма	Інженерія програмного забезпечення інтелектуальних кібер-фізичних систем в енергетиці
Статус дисципліни	Вибіркова
Форма навчання	очна(денна)
Рік підготовки, семестр	3 курс весняний семестр
Обсяг дисципліни	4 кредитів ЄКТС/120 год (36 год. лекції, 18 год. практичні заняття, 66 год. самостійна робота)
Семестровий контроль/ контрольні заходи	Залік
Розклад занять	https://schedule.kpi.ua/
Мова викладання	Українська
Інформація про керівника курсу / викладачів	Лектор: старший викладач, Колумбет В.П., kvplinux@gmail.com, тел. 093-405-35-39 Практичні: старший викладач, Колумбет В.П., kvplinux@gmail.com, тел. 093-405-35-39
Розміщення курсу	Кампус

Програма навчальної дисципліни

1. Опис навчальної дисципліни, її мета, предмет вивчання та результати навчання

В цій дисципліні детально вивчається концепції, принципи та методи інформаційних технологій. Етапи розробки програмного забезпечення. Міжнародні стандарти програмної інженерії. Моделі та процеси, які впроваджуються в розробленні програмного забезпечення. Вимоги до інформаційної системи. Документування розроблених вимог та моделей для створення інформаційної системи. Управління змінами до вимог протягом циклу розробки програмного забезпечення.

Метою дисципліни є опанування студентами теоретичних та практичних знань і навичок систем тестування, інструментальних засобів розробки програмних систем, розгортання та функціонування програмного забезпечення, розробляти, аналізувати та застосовувати ефективні алгоритми для розв'язання професійних завдань в області інформаційних технологій, та узгодження розробки й постачання програмного забезпечення із його використанням

Предмет дисципліни - вивчення принципів побудови, архітектури, основних функцій, режимів роботи і елементи та комплекси практик процесів неперервної інтеграції і деплойменту та розроблення адаптивних алгоритмів розв'язування задач.

Завдання. В результаті вивчення дисципліни у студентів повинні сформуватися наступні компетентності:

загальні:

- здатність використовувати математичні методи для прийняття ефективних рішень під час розв'язання професійних задач в процесі розробки ІС та ІТ (ПК-1);
- здатність використовувати сучасні комп'ютерні технології для системного, функціонального, конструкторського та технологічного проектування складних об'єктів і систем (ПК-7);

- здатність використовувати існуючі математичні моделі та методи при розв'язанні теоретичних і прикладних задач, що виникають при розробці ІТ та ІС (ПК-23);
- здатність до професійного володіння комп'ютерними технологіями (ПК-6с);
- здатність використання принципів структурного програмування, основних структур даних під час реалізації алгоритмів професійних завдань (ПК-6с);
- Здатність розв'язувати математичні, фізичні та економічні задачі шляхом створення відповідних застосувань (ПК-12с).
- здатність до застосування принципів, методів і алгоритмів обчислювальної математики до розробки підсистем моделювання інформаційних систем (ПК-12с).

Після засвоєння навчальної дисципліни студенти мають продемонструвати такі програмні результати навчання:

- дотримуватися морально-етичних та культурних норм, принципів академічної доброчесності та кодексу професійної етики, примножувати досягнення суспільства (ПРН 1),
- знати професійні стандарти та інші нормативно- правові документи в галузі інженерії програмного забезпечення (ПРН 10),
- використовувати методологію створення програмного забезпечення згідно поставленої задачі. Застосовувати на практиці ефективні підходи щодо проектування програмного забезпечення. Мотивовано обирати мови програмування та технології розробки для розв'язання завдань створення і супроводження програмного забезпечення (ПРН 16),
- володіти навичками командної розробки, погодження, оформлення і випуску всіх видів програмної документації (ПРН 17).

2. Пререквізити та постреквізити дисципліни (місце в структурно-логічній схемі навчання за відповідною освітньою програмою)

Пререквізити дисципліни. Знання, отримані при вивченні дисциплін: "Функціональне програмування" та "Архітектура та проектування програмного забезпечення".

Постреквізити дисципліни. Отримані знання при вивченні дисципліни «Архітектура системного програмного забезпечення» формує базові знання для вивчення наступних дисциплін: "Теорія ймовірностей та математична статистика", "Алгоритми та структури даних", "Бази даних", "Моделювання та аналіз програмного забезпечення", "Основи розробки трансляторів", "Системи штучного інтелекту", "Організація баз даних та знань", "Криптографія та шифрування". Також викладений матеріал може бути використаний при вивченні дисциплін "Математичне моделювання систем і процесів", "Дослідження операцій", які викладаються в наступних семестрах. Компетенції, отримані студентами в процесі вивчення цієї дисципліни, використовуються ними при виконанні дипломної роботи.

3. Зміст навчальної дисципліни

Тема 1. Основи та важливість CI/CD в сучасній розробці ПЗ.
Тема 2. Інструменти та технології в CI/CD.
Тема 3. Автоматизація тестування в CI/CD.
Тема 4. Безпека та команда в процесах CI/CD.
Тема 5. Стратегії розгортання та управління релізами.
Тема 6. Інфраструктура як код (IaC) та контейнеризація.
Тема 7. Майбутнє CI/CD та еволюція практик.

4. Навчальні матеріали та ресурси

Основна література

1. Інженерія програмного забезпечення: Посібник для студентів вищих навчальних закладів / І. Л. Бородкіна, Г. О. Бородкін ; М-во освіти і науки України, Національний університет біоресурсів та природокористування України. – Київ: , 2018. – с.
2. Kvaratskheliia T. E. Дослідження загроз та методів забезпечення безпеки хмарних обчислень [Електронний ресурс] / T. E. Kvaratskheliia, M. I. Lysenko // Topical issues of the development of modern

science. Abstracts of the 3rd International 88 scientific and practical conference. Publishing House “ACCENT”. Sofia, Bulgaria.2019.
3. Sommerville I. Software Engineering— 9th ed. / Ian Sommerville. – Addison-Wesley, 2011.
4. Сайт хмарного сервісу Amazon Web Services [Електронний ресурс] – Режм доступу до ресурсу https://aws.amazon.com/ru/devops/continuous-delivery/
5. Paul Duvall. Continuous Integration: Patterns and Anti-Patterns // DZone [Электронний ресурс]. URL: https://dzone.com/refcardz/continuous-integration.
6. Laplante, Phillip (2007). What Every Engineer Should Know about Software Engineering. Boca Raton: CRC. ISBN 9780849372285. Процитовано 2011-01-21.
7. Martin Fowler, "Continuous Integration", martinFowler, may 2006. URL: https://martinfowler.com/articles/continuousIntegration.html
8. Continuous Delivery: reliable software releases through build, test, and deployment automation. / Humble, Jez; Farley, David (2011). / Pearson Education Inc. ISBN 978-0-321-60191-9.

Додаткова література

1. PER BRINCH HANSEN, "DISTRIBUTED PROCESSES: A CONCURRENT PROGRAMMING CONCEPT", 1978. URL: http://brinchhansen.net/papers/1978a.pdf
2. “Jenkins User Handbook. Using Jenkins.” URL: https://www.jenkins.io/doc/book/using/
3. Martin Fowler. Continuous Integration // MartinFowler.com [Электронний ресурс]. URL: https://martinfowler.com/articles/continuousIntegration.html
4. Continuous integration. URL: https://en.wikipedia.org/wiki/Continuous_integration.
5. Визначення предмету-програмна інженерія.-/Лавріщева К. М.-Проблеми програмування.-Спецвипуск.-2008.-№ 2-3.-с.191-204.
6. Jenkins - Overview. URL: https://www.tutorialspoint.com/jenkins/jenkins_overview.htm
7. Lotfi Waderni, "HOW TO BUILD YOUR CUSTOM JENKINS DOCKER IMAGE", Yala-Labs, may 2020. URL: https://yallalabs.com/devops/jenkins/howto-build-custom-jenkins-docker-image/.
8. Peter, Naur; Brian Randell (7–11 October 1968). Software Engineering: Report of a conference sponsored by the NATO Science Committee (PDF). Garmisch, Germany: Scientific Affairs Division, NATO.
9. SWEBOK executive editors, Alain Abran, James W. Moore; editors, Pierre Bourque, Robert Dupuis. (2004). У Pierre Bourque and Robert Dupuis. Guide to the Software Engineering Body of Knowledge - 2004 Version. IEEE Computer Society. с. 1–1. ISBN 0-7695-2330-7.
10. Build automatio. URL: https://en.wikipedia.org/wiki/Build_automation.
11. The DevOps Handbook: How to Create World-Class Agility, Reliability, and Security in Technology Organizations / Gene Kim, Patrick Debois, John Willis, Jez Humble, John Allspaw / IT Revolution Press / October 6, 2016, 480p, ISBN-10 ： 1942788002 ISBN-13 ： 978-1942788003

Навчальний контент

5. Методика опанування навчальної дисципліни (освітнього компонента)

Тема 1. Основи та важливість CI/CD в сучасній розробці ПЗ.

Лекція 1. Вступ до неперервної інтеграції та неперервного розгортання
Основні поняття та переваги CI/CD.
Історія розвитку методологій CI/CD..

Тема 2. Інструменти та технології в CI/CD.

Лекція 2. Інструменти неперервної інтеграції
Огляд популярних інструментів CI: Jenkins, GitLab CI, CircleCI тощо.
Вибір інструменту для проекту.

Лекція 3. Конфігурація середовища для CI
Налаштування сервера CI.
Автоматизація тестування коду.

Лекція 4. Робочі процеси неперервної інтеграції
Створення ефективного CI пайплайну.
Управління залежностями та версіями.

Тема 3. Автоматизація тестування в CI/CD.

Лекція 5. Тестування в рамках CI
Автоматизоване тестування: юніт-тести, інтеграційні тести, системні тести.
Конфігурація середовища для автоматичного тестування.

Лекція 6. Інтеграція і робота з кодовими репозиторіями
Git workflows: feature branching, Gitflow, trunk-based development.
Автоматизація злиття коду та розв'язання конфліктів.

Тема 4. Безпека та команда в процесах CI/CD.

Лекція 7. Безпека в процесах CI/CD
Виявлення вразливостей у коді.
Керування доступом і секретами.

Лекція 8. Контейнеризація та CI/CD
Використання Docker у процесах CI/CD.
Основи Kubernetes для розгортання.

Тема 5. Стратегії розгортання та управління релізами.

Лекція 9. Неперервний розгортання (CD): основи
Відмінності між CD та CI.
Стратегії розгортання: blue-green deployment, canary release.

Лекція 10. Моніторинг і логування в CI/CD
Інструменти та практики моніторингу.
Аналіз логів та звітність.

Лекція 11. Оптимізація процесів CI/CD
Кешування та оптимізація швидкості побудов.
Автоматизація та скорочення затрат на обслуговування.

Лекція 12. Інфраструктура як код (IaC) у CI/CD
Використання Terraform, Ansible для автоматизації інфраструктури.
Переваги IaC у процесах розгортання.

Лекція 13. Керування конфігурацією в CI/CD
Інструменти та підходи до керування конфігурацією.
Автоматизація налаштувань середовищ.

Лекція 14. Cloud-native CI/CD
Використання хмарних платформ для CI/CD.
Специфіка розгортання у хмарі.

Тема 6. Інфраструктура як код (IaC) та контейнеризація.

Лекція 15. Мікросервісна архітектура та CI/CD
Особливості розгортання мікросервісів.
Управління залежностями та версіями в мікросервісному середовищі.

Лекція 16. Залучення команди до процесів CI/CD
Культура DevOps та її вплив на CI/CD.
Роль комунікацій та співпраці.

Лекція 17. Кейс-стаді з реального життя: впровадження CI/CD
Аналіз успішних кейсів впровадження CI/CD.
Перешкоди та виклики під час впровадження.

Тема 7. Майбутнє CI/CD та еволюція практик.
Лекція 18. Майбутнє CI/CD та останні тренди
Інновації та майбутній розвиток CI/CD.
Вплив штучного інтелекту та машинного навчання на CI/CD.

6. Самостійна робота студента

Тема 1. Аналіз і порівняння інструментів CI/CD
Дослідіть та порівняйте щонайменше три різні інструменти CI/CD (наприклад, Jenkins, GitLab CI/CD, CircleCI), зосередившись на їх основних характеристиках, перевагах та недоліках.

Тема 2. Встановлення та налаштування Jenkins на власному комп'ютері
Встановіть Jenkins, створіть простий проект і налаштуйте автоматичну збірку проекту з GitHub кожного разу, коли в репозиторій здійснюється push.

Тема 3. Створення Dockerfile для власного проекту
Напишіть Dockerfile для контейнеризації будь-якого простого веб-додатку або скрипта. Побудуйте образ і запустіть його локально.

Тема 4. Розгортання простого застосунку використовуючи Heroku
Використовуючи платформу Heroku, розгорніть простий веб-додаток і налаштуйте автоматичне розгортання з GitHub репозиторію.

Тема 5. Автоматизація тестування з використанням GitLab CI/CD
Створіть простий проект з юніт-тестами. Налаштуйте .gitlab-ci.yml файл для автоматичного запуску тестів при кожному push в репозиторій.

Тема 6. Практика з інфраструктурою як код (IaC) за допомогою Terraform
Використовуючи Terraform, напишіть конфігурацію для створення віртуальної машини в AWS або іншому хмарному провайдері.

Тема 7. Інтеграція та використання ELK Stack для моніторингу логів
Розгорніть Elasticsearch, Logstash та Kibana (ELK Stack) для збору та аналізу логів з вашого проекту або додатку.

Тема 8. Виконання blue-green розгортання за допомогою Kubernetes
Використовуючи мінімальний Kubernetes кластер (наприклад, Minikube), реалізуйте стратегію blue-green розгортання для простого веб-додатку..

Політика та контроль

7. Політика навчальної дисципліни (освітнього компонента)

Відвідування лекційних та лабораторних занять є обов’язковим за винятком поважних причин (хвороби, форс-мажорних обставин).
В разі пропущення занять з поважних причин викладач надає можливість студенту виконати усі або деякі лабораторні завдання (винятком є виконання деяких завдань у зв’язку із закінченням навчального процесу).

В разі пропущення занять без поважних причин, а також через порушення граничного терміну виконання завдання (deadline) студент може отримати 80% від максимальної оцінки відповідне завдання.

Протягом семестру студенти:

- виконують та захищають лабораторні роботи у відповідні терміни (на кожну лабораторну роботу відводиться два тижні для здачі),
- пишуть модульну контрольну роботу,
- повинні позитивно закрити дві атестації (в кінці березня та в середині травня),
- по закінченні навчального процесу складають екзамен.

8. Види контролю та рейтингова система оцінювання результатів навчання (РСО)

Система рейтингових (вагових) балів та критерії оцінювання

1) Робота на лекціях

На лекціях може бути проведено бліцопитування студентів. Такі опитування проводяться на довільних лекціях 5 разів протягом семестру, наприкінці лекції. Ваговий бал за вірну відповідь - 1. Максимальна кількість балів, що може отримати кожен студент за семестр - 5.

2) Лабораторні практикуми

Максимальна кількість балів за усі виконані комп'ютерні практикуми дорівнює 60 балів. Розподіл балів серед лабораторних практикумів наступний:

№ з/п	Назва лабораторного практикуму	Кількість балів
1	Робота з Git та реалізація Git Hooks	12
2	Контейнеризація за допомогою Docker	12
3	Використання Docker Compose для оркестрації контейнерів	12
4	Впровадження стратегії неперервного розгортання за допомогою GitLab CI/CD	12
5	Моніторинг та логування в CI/CD процесах	12
Всього:		60

Критерії оцінювання:

Виконання лабораторного практикуму:

- виконаний своєчасно (протягом двох тижнів з моменту видачі), у повному обсязі – відповідний бал згідно номеру комп'ютерного практикуму;
- виконаний із запізненням – знімається 10 – 30% від максимальної кількості балів в залежності від терміну запізнення;
- виконаний не самостійно, із запізненням – знімається 50% від максимальної кількості балів;
- невиконани1 протягом відведеного часу – 0 балів.

3) Модульна контрольна робота

Максимальна кількість балів за модульну контрольну роботу дорівнює 10 балів.

Якість виконання роботи:

- усі відповіді вірні та повні –10 балів,
- у відповідях допущені несуттєві неточності – 8 балів,
- половина відповідей вірна – 5 балів,
- відповіді з суттєвими неточностями, але без критичних помилок – 2 бали,
- менше половини відповідей вірна – 0 балів.

Штрафні та заохочувальні бали за:

- активність на комп'ютерних практикумах + 2 бали
- виконання комп'ютерного практикуму з використанням власного оптимального алгоритму + 1 бали

- відсутність на занятті без поважної причини – 2 бали
- несвоєчасна здача комп'ютерного практикуму (пізніше ніж за тиждень) – 0,5 балів;

4) Складання заліку

Максимальний ваговий бал $r_{icп}$=40

На іспиті студент виконує письмову контрольну роботу, яка містить два теоретичних питання і одне практичне питання. Теоретичні питання оцінюються максимально по 10 балів, практичне – 20 балів.

Умови позитивної проміжної атестації

Для отримання „зараховано" з першої проміжної атестації студент матиме не менше ніж 30 балів (за умови, що за 8 тижнів згідно з календарним планом контрольних заходів „ідеальний" студент має отримати 12 +12 +12 = 36 бал).

Для отримання „зараховано" з другої проміжної атестації студент матиме не менше ніж 50 балів (за умови, що за 14 тижнів згідно з календарним планом контрольних заходів „ідеальний" студент має отримати 36 +12 +12 =60 балів).

Умови допуску до *залікуу*

Необхідною умовою допуску до заліку є зарахування усіх комп'ютерних практикумів та виконання модульної контрольної роботи, а також стартовий рейтинг (R_c) не менше 40 балів.

Розрахунок шкали (R) рейтингу:

Сума вагових балів контрольних заходів протягом семестру (шкала рейтингу) складає:

$$R = r_{лек} + r_{практ} + r_{мод} + r_{icп} = 5+45 + 10 + 40= 100 \text{ балів.}$$

Максимальний стартовий рейтинг становить $R_c = r_{лек} + r_{практ} + r_{мод} =$ 60 балів.

Рейтинг заліку дорівнює 40 балів. Мінімальний рейтинг допуску до заліку становить 40 балів.

Таким чином, рейтингова шкала з кредитного модуля складає

$$R = 60+40=100 \text{ балів.}$$

Для отримання студентом відповідних оцінок рейтингова оцінка студента переводиться згідно таблиці:

Бали	**Оцінка**
95 - 100	Відмінно
85 - 94	Дуже добре
75 - 84	Добре
65 - 74	Задовільно
60 - 64	Достатньо
Менше 60	Незадовільно
R < 40 є незараховані роботи комп'ютерного практикуму або не виконані інші умови допуску до екзамену	Не допущено

1. Додаткова інформація з дисципліни (освітнього компонента)

Перелік питань, які виносяться на семестровий контроль:

1. Що таке неперервна інтеграція (CI)? Опишіть основні принципи.
2. Що таке неперервне розгортання (CD)? Як воно відрізняється від неперервної доставки?
3. Які переваги впровадження CI/CD для проекту з розробки програмного забезпечення?
4. Назвіть та описіть функціонал трьох інструментів для реалізації CI/CD.
5. Які основні компоненти CI/CD пайплайну?
6. Що таке "pipeline as code"? Наведіть приклад.

7. Опишіть процес налаштування автоматичних тестів у CI/CD пайплайні.
8. Як контейнеризація впливає на процеси CI/CD?
9. Опишіть стратегію blue-green розгортання. Які її переваги?
10. Які виклики можуть виникати під час впровадження CI/CD?
11. Які методи забезпечення безпеки ви знаєте для CI/CD пайплайну?
12. Що таке IaC (Infrastructure as Code) і яку роль він відіграє в CI/CD?
13. Назвіть приклади інструментів для IaC та опишіть їх основні функції.
14. Як здійснюється моніторинг і логування в CI/CD?
15. Опишіть процес ручного vs автоматичного розгортання. Коли варто використовувати кожен з них?
16. Як використання мікросервісної архітектури впливає на CI/CD пайплайн?
17. Як забезпечити високу доступність та масштабованість CI/CD інфраструктури?
18. Що таке Canary releases? Як вони впроваджуються у CI/CD?
19. Як управління конфігурацією інтегрується з CI/CD?
20. Як вибрати між самостійним розгортанням CI/CD інструментів проти використання хмарних сервісів?
21. Які критерії ви б використали для оцінки ефективності CI/CD пайплайну?
22. Які практики DevOps можуть підтримувати успішне впровадження CI/CD?
23. Що таке GitOps і як воно пов'язане з CI/CD?
24. Як зміни в коді просуваються через CI/CD пайплайн від розробки до продакшну?.

Робочу програму навчальної дисципліни (силабус) «Процеси неперервної інтеграції і деплойменту»:

Складено старшим викладачем ІПЗЕ, Колумбетом Вадимом Петровичем

Ухвалено кафедрою ІПЗЕ (протокол № 28 від 15.05.2023 р.)
Погоджено Методичною комісією ННІАТЕ КПІ ім. Ігоря Сікорського (протокол № 9 від 26.05.2023 р.)