МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ХАРКІВСЬКИЙ НАЦІОНАЛЬНИЙ ЕКОНОМІЧНИЙ УНІВЕРСИТЕТ
ІМЕНІ СЕМЕНА КУЗНЕЦЯ

ЗАТВЕРДЖЕНО
на засіданні кафедри
інформаційних систем
Протокол № 1 від 22.08.2023 р.

ПОГОДЖЕНО
Проректор з навчально-методичної роботи

_____Каріна НЕМАШКАЛО

СУЧАСНІ ТЕХНОЛОГІЇ ПРОГРАМУВАННЯ

робоча програма навчальної дисципліни (РПНД)

Галузь знань	**12 "Інформаційні технології"**
Спеціальність	**121 "Інженерія програмного забезпечення"**
Освітній рівень	**перший (бакалаврський)**
Освітня програма	**"Інженерія програмного забезпечення"**

Статус дисципліни	**вибіркова**
Мова викладання, навчання та оцінювання	**українська**

Розробник:
к.т.н., доцент підписано КЕП Андрій ПОЛЯКОВ

Завідувач кафедри
інформаційних систем _____ Дмитро БОНДАРЕНКО

Гарант програми _____ Олег ФРОЛОВ

Харків
2024

ВСТУП

Навчальна дисципліна "Сучасні технології програмування" діє на передовій лінії інновацій у сфері програмування, надаючи здобувачам найновіші технології та тенденції. У сучасному світі програмування ґрунтується на неперервному процесі вивчення та адаптації до змін, і ця дисципліна створена з метою підготувати здобувачів до такого динамічного середовища. Основу вивчення складає робота з RESTful API, особливостями технології віртуалізації та контейнеризації за допомогою Docker, використання хмарних сервісів AWS, засади роботи з процесами CI/CD та практику впровадження концепції "інфраструктури як код" через Terraform. Набуваються не тільки теоретичні знання, але й їх застосування на практиці, працюючи у реальних середовищах. Це дозволяє здобувачам розуміти, як функціонують сучасні програмні системи, та ефективно інтегруватись у професійну область програмування. Дисципліна надає навички та знання, які допоможуть орієнтуватися в майбутньому в динамічному світі сучасних технологій програмування.

Вивчення дисципліни "Сучасні технології програмування" передбачає освоєння навичок розробки та проектування з використанням сучасних технологій, таких як RESTful API, віртуалізацію та контейнеризації через Docker, використання хмарних сервісів AWS, роботи з процесами CI/CD та впровадження принципів "інфраструктури як код" через Terraform. Дозволяє здійснювати підготовку спеціалістів, здатних ефективно працювати в сучасному програмному середовищі.

Метою викладання навчальної дисципліни "Сучасні технології програмування" є формування у здобувачів теоретичних знань та практичних навичок у сфері сучасних технологій та методів програмування. Це передбачає вміння розробляти та застосовувати високоефективні технології програмування, а також створювати, проектувати та управляти системами на основі мікросервісної архітектури. Здобувачі вищої освіти повинні оволодіти навичками використання RESTful API, технології віртуалізації та контейнеризації (Docker), хмарних сервісів (AWS), технологіями CI/CD, працювати з базами даних та впроваджувати концепцію “інфраструктури як код”. Робота з навчальним матеріалом дозволяє розвивати навички самостійної роботи, критичного мислення, командної взаємодії та презентування власних ідей та програмних розробок.

Завданнями навчальної дисципліни є:

- формування у здобувачів сучасного погляду на процеси розробки програмного забезпечення, використовуючи найновітніші технології та практики;
- ознайомлення з принципами роботи та специфікою розробки RESTful API;
- навчити основам віртуалізації та контейнеризації застосунків з використанням Docker;

- вивчити процеси CI/CD та їх важливість у сучасній розробці програмного забезпечення;
- освоїти принципи роботи з хмарними сервісами на прикладі AWS та розробки безсерверних застосунків;
- вивчити інфраструктуру як код на прикладі Terraform;
- набути навичок командної роботи та використання систем контролю версій;
- набути критичного мислення і вироблення професійних навичок, необхідних для неперервного самовдосконалення у сфері IT.

Предметом навчальної дисципліни є вивчення основних принципів, підходів та методів проектування, розробки та тестування сучасних програмних систем, інтеграції різних компонентів системи, використання хмарних технологій та безсерверних архітектур, а також впровадження циклу неперервної інтеграції та доставки.

Об'єктом навчальної дисципліни є сучасні методи, технології та інструменти, які застосовуються при проектуванні, розробці, випуску та підтримці програмних продуктів та систем. Це включає такі компоненти, як RESTful API, робота з Docker, процеси CI/CD, розробка базових об'єктів інфраструктури через код за допомогою Terraform, використання хмарних сервісів AWS і втілення концепції serverless.

Результати навчання та компетентності, які формує навчальна дисципліна визначено в табл. 1.

Таблиця 1

Результати навчання та компетентності, які формує навчальна дисципліна

Результати навчання	Компетентності, якими повинен оволодіти здобувач вищої освіти
РН07	ЗК05, СК13.
РН12	СК03, СК14.
РН17	ЗК02, СК12.
РН18	ЗК06, СК12.
РН21	ЗК02, СК08.

де, РН07. Знати і застосовувати на практиці фундаментальні концепції, парадигми і основні принципи функціонування мовних, інструментальних і обчислювальних засобів інженерії програмного забезпечення.

РН12. Застосовувати на практиці ефективні підходи щодо проектування програмного забезпечення.

РН17. Вміти застосовувати методи компонентної розробки програмного забезпечення.

РН18. Знати та вміти застосовувати інформаційні технології обробки, зберігання та передачі даних.

РН21. Знати, аналізувати, вибирати, кваліфіковано застосовувати засоби забезпечення інформаційної безпеки (в тому числі кібербезпеки) і цілісності даних відповідно до розв'язуваних прикладних завдань та створюваних програмних систем.

ЗК02. Здатність застосовувати знання у практичних ситуаціях.

ЗК05. Здатність вчитися і оволодівати сучасними знаннями.
ЗК06. Здатність до пошуку, оброблення та аналізу інформації з різних джерел.
СК03. Здатність розробляти архітектури, модулі та компоненти програмних систем.
СК08. Здатність застосовувати фундаментальні і міждисциплінарні знання для успішного розв'язання завдань інженерії програмного забезпечення
СК12. Здатність здійснювати процес інтеграції системи, застосовувати стандарти і процедури управління змінами для підтримки цілісності, загальної функціональності і надійності програмного забезпечення.
СК13. Здатність обґрунтовано обирати та освоювати інструментарій з розробки та супроводження програмного забезпечення.
СК14. Здатність до алгоритмічного та логічного мислення.

ПРОГРАМА НАВЧАЛЬНОЇ ДИСЦИПЛІНИ

Зміст навчальної дисципліни

Змістовий модуль 1. Проектування, розробка та управління API RESTful сервісів.

Тема 1. Розвиток сервісів RESTful, їх обмеження та дизайн.

1.1.Вступ до дисципліни. Контрольні заходи та результати навчання.

1.2.Що таке API?: Еволюція REST/JSON API.

1.3.Знайомство з RESTful API, приватні, публічні та партнерські API, ланцюжок цінності API.

1.4.Вступ до обмежень архітектури REST.

Тема 2. Шаблони REST API.

2.1.Шаблони обробки помилок REST API.

2.2.Шаблони керування версіями REST API: обробка змін, керування версіями API, реалізація керування версіями ACME API.

2.3.Шаблон керування кеш-пам'яттю REST API: проектування та концепція API кешування, директива з керування кешем, реалізація API кешування за допомогою директив керування кеш-пам'яттю.

2.4.Шаблон обробки даних відповіді REST API: часткові відповіді та їх реалізація, розбиття на сторінки (pagination) та їх реалізація.

Тема 3. Безпека REST API та OpenAPI специфікація v3.1.0, Swagger, Управління API.

3.1.Захист API за допомого: базової автентифікації, токенів і JWT, ключа та секрету API, авторизація API за допомогою OAuth2.0, функціональні атаки.

3.2.Процес аналізу вимог, вступ до специфікацій REST, специфікацій Swagger/OAI.

3.3.Впровадження управління API, життєвий цикл API, портал розробників API, керування безпекою / трафіком API, API Analytics, продаж API та монетизація API.

Змістовий модуль 2. Основи віртуалізації та контейнеризації застосунків та сервісів.

Тема 4. Основи віртуалізація та контейнеризація. Основи платформи Docker.

4.1. Контейнеризація та віртуалізація, вступ до Docker, архітектура Docker, контейнери та образи, docker cli, управління контейнерами, життєвий цикл.

4.2. Основи створення образів (Dockerfile): команди FROM; WORKDIR, COPY, ADD; RUN; ENV, LABEL, USER; VOLUME і EXPOSE; VOLUME і EXPOSE; VOLUME і EXPOSE.

4.3. Образи для виробничого середовища: параметризація файлів Docker за допомогою ARG, створення та запуск багаторазових зображень, час збірки проти часу виконання, створення менших зображень, багатоетапне створення зображення, створення базового зображення з нуля, сервери додатків для запуску, обладнання баз даних робочого рівня, впровадження проксі-сервера, необхідність автоматизації.

Тема 5. Основи роботи з даними у контейнері та репозитарії образів.

5.1. Томи Docker-у: проблеми збереження, типи томів.

5.2. Реєстри Docker, налаштування локального реєстру Docker, Docker HUB, пошук та запуск образів.

Тема 6. Оркестровка контейнерів. docker composer.

6.1. Запуск контейнерів у Docker: запуск виробничих контейнерів у Docker, основи Docker Compose.

6.2. Docker Compose File: версія та томи, мережі, служби, керування зображеннями та життєвий цикл програми з Docker Compose.

6.3. Docker Swarm: архітектура та сервіси Swarm, підготовка Swarm за допомогою Docker Machine, автономні контейнери та послуги в Swarm, режими обслуговування та сітка маршрутизації входу, стек застосунків у Swarm, середовище та життєвий цикл програми в Swar, стислий опис Docker Runtime Environment.

Змістовий модуль 3. Проектування та розробка процесів CI/CD.

Тема 7. Концепція CI. Jenkins.

7.1. Концепція безперервної інтеграції, доставки та розгортання (CI/CD).

7.2. Архітектура Jenkins-а: конфігурування та налаштування, плагіни, безпека, резервне копіювання, сховище секретів.

7.3. Jenkins Job: налаштування Job, сценарії від Jenkins, параметризація Job, логічний ввід у Jenkins Job, безперервна інтеграція (CI).

7.4. Pipeline у Jenkins: декларативні та скриптові pipeline,

Тема 8. Типи розгортання. Jenkins Pipeline.

8.1. Розгортання: Blue/Green, Canare, A/B.

8.2. Безперервне розгортання (CD) із Jenkins. Інфраструктура як код.

8.3. Робота Jenkins DSL, Jenkins як конвеєр коду, Розподілені збірки в Jenkins, Інтеграція Jenkins з Docker, Аспекти безпеки Jenkins.

Змістовий модуль 4. Хмарні технології та Cloud-native застосунки. Інфраструктура як код (IaC).

Тема 9. Введення до AWS, керування ідентифікацією та доступом, AWS Computing.

9.1. Початок роботи з AWS Cloud: Що таке AWS? Глобальна інфраструктура AWS. Взаємодія з AWS. Створення облікового запису AWS

9.2. Безпека в AWS: Безпека та модель спільної відповідальності AWS. Захист root-користувача AWS. Підходи до захисту вашого облікового запису AWS.

9.3. Управління ідентифікацією та доступом до AWS: Вступ до управління ідентифікацією та доступом в AWS. Доступ на основі ролей в AWS. Демонстрація AWS IAM. Розміщення додатку довідника співробітників в AWS. Образ машини Amazon за замовчуванням (AMI) для Amazon EC2.

9.4. AWS Compute: Обчислення як послуга на AWS. Вступ до хмари еластичних обчислень Amazon. Життєвий цикл екземпляра Amazon EC2. Контейнерні сервіси на AWS.

Тема 10. AWS мережи, сховища та бази даних.

10.1. Робота в мережі AWS: Робота в мережі на AWS. Вступ до Amazon VPC. Маршрутизація Amazon VPC. Захист мережі за допомогою Amazon VPC Security. Гібридне підключення з AWS.

10.2. Створення VPC і перезапуск програми корпоративного каталогу на Amazon EC2.

10.3. Типи сховищ на AWS: Amazon EC2 Instance Storage та Amazon Elastic Block Store. Об'єктне сховище з Amazon S3. Вибір правильного сервісу для зберігання даних. Створіть Amazon S3 Bucket.

10.4. Бази даних на AWS. Вивчення баз даних на AWS. Служба реляційних баз даних Amazon. Спеціалізовані бази даних на AWS. Вступ до Amazon DynamoDB. Вибір правильного сервісу баз даних AWS.

Тема 11. Інфраструктура як код (IaC). Terraform.

11.1. Що таке інфраструктура як код (IaC)? Переваги інфраструктури як коду, основи Terraform. Порівняння Terraform з іншими засобами IaC.

11.2. Робота з Terraform. Підготовка вашого облікового запису в AWS. Встановлення Terraform. розгортання одного сервера, одного веб-сервера, конфігурованого веб-сервера, кластера веб-серверів, балансувальника навантаження. Видалення непотрібних ресурсів.

11.3. Повторне використання інфраструктури за допомогою модулів Terraform: що таке модуль? Вхідні параметри модуля, локальні та вихідні змінні модуля. Особливості використання модулів. Управління версіями.

11.4. Робота з Terraform: цикли, умовні вирази, розгортання та особливості використання: цикли, умовні вирази, розгортання з нульовим часом простою.

Тема 12. Безсерверні застосунки. Введення до AWS Lambda.

12.1. Фактори хмарного застосунку.

12.2. AWS CLI та API. Вступ до AWS API Management Console CLI SDK. Вступ до AWS CLI. Cloud9, AWS API, AWS CLI. Вивчення AWS SDK (Java). Використання тимчасових облікових даних в AWS Cloud9. Модель безсерверних додатків. Інструментарій AWS для IntelliJ. Cloud9 Temporary

Credentials, AWS SDK, AWS Toolkits, AWS SAM - Java. Налаштування та вивчення SDK.

12.3. Вступ до проектування на основі API. Розробка на основі API. Що таке API-шлюз? Dragon API: Термінологія API-шлюзу. Що таке API Gateway Notes, API Driven Dev Notes. Моделі та відображення. Створення GET API за допомогою імітації інтеграції. Публікація API. Використання Postman для створення запитів. Етапи шлюзу API, розгортання, виклик, Postman.

12.4. Аутентифікація API. Вступ до автентифікації та шлюзу API. Управління доступом до шлюзу API. Автентифікація та авторизація шлюзу API. Вступ до Amazon Cognito. Використання Amazon Cognito для входу та виклику шлюзу API. Використання Amazon Cognito для входу та виклику API-шлюзу за допомогою JavaScript.

12.5. Безсерверні обчислення та лямбда. Вступ до AWS Lambda. Виконання AWS Lambda. Дозволи AWS Lambda. Вступ до Лямбда, виконання Лямбда, дозволи Лямбда. Тригери, модель Push, Pull. Повторне використання контексту лямбда-виконання. Відповідність AWS Lambda. Асинхронні та синхронні відповіді.

12.6. Псевдоніми та версії. Створення лямбда-функції AWS - Java. Створення та налагодження лямбда за допомогою AWS Toolkit для IntelliJ. Створення лямбда-функції, версії та псевдоніми.

Перелік лабораторних занять за навчальною дисципліною наведено в табл. 2.

Таблиця 2

Перелік лабораторних занять

Назва теми	Зміст
Тема 1 – 3.	Проектування та розробка API для RESTful з використанням Swagger та сучасних фреймворків
Тема 4 – 6.	Дослідження засобів контейнеризації та створення образу застосунку
Тема 7, 8.	Побудова процесу автоматизації CI/CD
Тема 9 – 11.	Розробка інфраструктури у AWS з Terraform.
Тема 12.	Розробка Cloud-native застосунку у AWS

Перелік самостійної роботи за навчальною дисципліною наведено в табл. 3.

Таблиця 3

Перелік самостійної роботи

Назва теми	Зміст
Тема 1 – 12	Вивчення лекційного матеріалу
Тема 1 – 12	Підготовка до лабораторних занять
Тема 1 – 12	Підготовка до поточних контрольних робіт
Тема 1 – 12	Підготовка до екзамену

Кількість годин лекційних та лабораторних занять, а також годин самостійної роботи наведено в робочому плані (технологічній карті) з навчальної дисципліни.

МЕТОДИ НАВЧАННЯ

У процесі викладання навчальної дисципліни для набуття визначених результатів навчання, активізації освітнього процесу передбачено застосування таких методів навчання, як:

Словесні (лекція (Тема 1, 2, 4, 7, 9,12), проблемна лекція (Тема 1 – 3, 7, 8), лекція-візуалізація (Тема 1 – 12)).

Наочні (демонстрація (Тема 1 – 12)).

Лабораторна робота (Тема 1 – 12), кейс-метод (Тема 1 – 3, 6, 8, 10, 11).

ФОРМИ ТА МЕТОДИ ОЦІНЮВАННЯ

Університет використовує 100 бальну накопичувальну систему оцінювання результатів навчання здобувачів вищої освіти.

Поточний контроль здійснюється під час проведення лекційних, лабораторних занять і має на меті перевірку рівня підготовленості здобувача вищої освіти до виконання конкретної роботи і оцінюється сумою набраних балів:

– для дисциплін з формою семестрового контролю екзамен (іспит): максимальна сума – 60 балів; мінімальна сума, що дозволяє здобувачу вищої освіти складати екзамен (іспит) – 35 балів.

Підсумковий контроль включає семестровий контроль та атестацію здобувача вищої освіти.

Семестровий контроль проводиться у формі семестрового екзамену (іспиту). Складання семестрового екзамену (іспиту) здійснюється під час екзаменаційної сесії.

Максимальна сума балів, яку може отримати здобувач вищої освіти під час екзамену (іспиту) – 40 балів. Мінімальна сума, за якою екзамен (іспит) вважається складеним – 25 балів.

Підсумкова оцінка за навчальною дисципліною визначається сумуванням балів за поточний та підсумковий контроль.

Під час викладання навчальної дисципліни використовуються наступні контрольні заходи:

Поточний контроль: захист лабораторних робіт (32 балів), поточні контрольні роботи (10 балів), презентацій (18).

Семестровий контроль: Екзамен (40 балів)

Більш детальну інформацію щодо системи оцінювання наведено в робочому плані (технологічній карті) з навчальної дисципліни.

Приклад екзаменаційного білета та критерії оцінювання для навчальної дисципліни.

Приклад екзаменаційного білета

Харківський національний економічний університет імені Семена Кузнеця
Перший (бакалаврський) рівень вищої освіти
Спеціальність «Інформаційні системи та технології»
Освітньо-професійна програма «Інформаційні системи та технології»
Семестр I
Навчальна дисципліна «Сучасні технології програмування»

ЕКЗАМЕНАЦІЙНИЙ БІЛЕТ № 1

Завдання 1. Тест.

Питання № 1	
Яке з наступних тверджень є правильним в AWS	
Виберіть один з 4 варіантів відповіді:	
1)	EC3 - це постачальник аналітики як послуги
2)	Amazon Elastic Cloud - система для створення віртуальних дисків
3)	SimpleDB взаємодіє з Amazon EC2 та Amazon S3
4)	Нічого з перерахованого вище

Питання № 2	
Припустимо, що створено підмережу і в ній запущено екземпляр EC2 з налаштуваннями за замовчуванням. Які з наведених нижче опцій будуть готові до використання в екземплярі EC2, щойно його буде запущено?	
Виберіть один з 4 варіантів відповіді:	
1)	Elastic IP
2)	Публична IP-адреса
3)	Приватна IP-адреса
4)	Internet Gateway

Питання № 3	
У визначенні Swagger, чим визначається кожна кінцева точка в розділі, який містить всі кінцеві точки?	
Виберіть один з 4 варіантів відповіді:	
1)	абсолютна URL-адреса
2)	рядок запиту
3)	відносна URL-адреса
4)	Метод HTTP

Питання № 4	
Docker контейнер часто описують як покращення порівняно з якою іншою технологією?	
Виберіть один з 4 варіантів відповіді:	
1)	віртуальні машини
2)	хмарні обчислення
3)	DevOps
4)	мікросервіси

Питання № 5	
Яке речення краще описує Docker?	
Виберіть один з 4 варіантів відповіді:	
1)	Нічого з перерахованого вище
2)	Побудувавши один раз, запускай всюди
3)	Просто керувати - набагато приємніше
4)	Будуй один раз, запускай двічі

Питання № 6	
Яке HTTP-дієслово зазвичай використовується для оновлення або створення ресурсу в API?	
Виберіть один з 4 варіантів відповіді:	
1)	POST
2)	WRITE
3)	SUBMIT
4)	CREATE

Питання № 7	
У чому полягає одна з переваг GraphQL над REST-підходами?	
Виберіть один з 4 варіантів відповіді:	
1)	більш безпечні за замовчуванням
2)	гнучкість запитів/відповідей
3)	сумісність з великою кількістю шлюзів
4)	більш стабільні API

Питання № 8	
Які переваги автомасштабування?	
Виберіть один з 4 варіантів відповіді:	
1)	Відмовостійкість
2)	Краща доступність
3)	Краще управління витратами
4)	Все вищезазначене

Питання № 9	
_____ - це екземпляри образів Docker, які можна запустити за допомогою команди Docker run	
Виберіть один з 4 варіантів відповіді:	
1)	Container
2)	File

3)	Cloud
4)	Hub

Питання № 10	
Які типи вхідних параметрів запиту можна використовувати в Swagger?	
Виберіть один з 4 варіантів відповіді:	
1)	Використовуйте 'host' для вхідного параметра host, 'port' для вхідного параметра port, 'method' для вхідного параметра method
2)	У визначенні Swagger немає вхідних параметрів запиту
3)	Використовуйте 'path' для вхідного параметра URL-адреси, 'query' для вхідного параметра рядка запиту, 'body' для вхідного параметра тіла запиту
4)	Використовуйте 'url' для вхідного параметра URL, 'querystring' для вхідного параметра рядка запиту, 'content' для вхідного параметра тіла запиту

Завдання 2.

Робота з контейнеризованими застосунками

1. Розгортання контейнера:

a. Завантажити зображення httpd:2.4.57-alpine у локальний репозитарії. (Зробити скріншот процесу та результату, навести команду).

b. Вивести список доступних зображень на docker сервері (Зробити скріншот результату)

c. Створити контейнер з зображення httpd:2.4.57-alpine, надайте йому назву перши букви вашого ПІБ та запустити його. (Навести команду, зробити скріншот результату)

d. Вивести список контейнерів, що виконуються. (Навести команду, зробити скріншот результату)

e. Підключитися до контейнеру та виконати його дослідження (bash або sh. Навести команди, зробити скріншот результату):

i. вивести інформацію відносно версії та назві операційної системи з файлів (/etc/*release);

ii. отримати назву контейнера(hostname);

iii. отримати інформацію по версію apach (httpd -v).

f. Зупинити контейнер та вивести список усіх контейнерів. (Навести команду, зробити скріншот результату).

g. Відновити роботу контейнера. (Навести команду, зробити скріншот результату)

2. Розгортання сервісу:

a. Зробить ще один контейнер з зображення httpd:2.4.57-alpine з налаштуванням пробросу порта 80 з контейнеру на 8083 хостової OS. (Зробити скріншот процесу та результату, навести команду)

b. Перевірити у браузері, що сервіс nginx доступний за адресою http://localhost:8083 або з хостової консолі командою “curl http://localhost:8083”

Затверджено на засіданні кафедри інформаційних систем
протокол № ____ від «___»_____20___р.

Екзаменатор _____ к.т.н., доц. Поляков А. О.

Зав. кафедрою _____ к.т.н., доц. Бондаренко Д. О.

Критерії оцінювання

Підсумкові бали за екзамен складаються із суми балів за виконання всіх завдань, що округлені до цілого числа за правилами математики.

Алгоритм вирішення кожного завдання включає окремі етапи, які відрізняються за складністю, трудомісткістю та значенням для розв'язання завдання. Тому окремі завдання та етапи їх розв'язання оцінюються відокремлено один від одного таким чином:

Завдання 1 (тест).

Перше питання присвячене перевірці теоретичних знать з дисципліни. На тест виноситься 10 питань по 1.5 балу за кожне. Загалом 15 балів. Тривалість тесту 20 хвилин.

Завдання 2 (евристичне).

Друге питання присвячене розробці процесу розгортання застосунку з використанням контейнеризованого середовища Docker. Конфігураційні файли пишуться з використанням мови YAML та bash команд. Розробляється набіл команд, що дозволяє керувати життєвим циклом контейнерів зі застосунком, та проводити моніторинг середовища. Здобувач повинен використовувати середовище Visual Code, Dokker, Linux та bash. При цьому здобувачу дозволяється користування існуючою довідковою літературою. Після перевірки програми здобувач отримує K_1 балів за наступними вимогами (табл. 4).

Таблиця 4

Критерії оцінювання за діагностичним завданням

Бали K_1	Вимоги
25	Завдання виконане в повному обсязі.
24	Завдання виконане в повному обсязі. Є невеликі зауваження до організації структури коду та інтерфейсу.
22 – 23	Завдання в основному виконано. Є зауваження до організації структури коду та організації середовища розгортання.
20 – 21	Завдання виконане, але не в повному обсязі. Проект контейнеру працює, але не реалізовані деякі вимоги або можливості до середовища розгортання, зазначені в завданні.
18 – 19	Завдання виконане, але не в повному об'ємі. Проект контейнеру працює, але не реалізовані дві або три вимоги або можливості до середовища розгортання, зазначені в завданні.
16 – 17	Завдання виконане, але не в повному об'ємі. Проект контейнеру працює, але не реалізовані чотири або п’ять вимог або можливостей до середовища розгортання, зазначені в завданні.
11 – 15	Завдання виконане, але не в повному об'ємі. Проект контейнеру працює, але не реалізовані більш ніж п’ять вимог або можливостей до середовища розгортання, зазначені в завданні.
7 – 10	Проект контейнеру як мінімум дозволяє правильно виконати одну функціональність середовища розгортання.
3 – 6	Контейнер запускається, але містить грубі помилки, повністю не виконано жодної вимоги до середовища розгортання, що наведені у завданні.
2	Контейнер або середовище не відповідає постановці завдання.
1	Проект контейнеру або середовища не містить програмного коду, розробленого студентом. Проект контейнеру або середовища має явні ознаки несамостійності її розробки.
0	Програма відсутня.

РЕКОМЕНДОВАНА ЛІТЕРАТУРА

Основна

1. Varanasi B. and others Spring REST: Building Java Microservices and Cloud Applications / B. Varanasi, M. Bartkov. — Berkeley, CA : Apress, 2022. — 251 p.

2. Gkatziouras E. A Developer's Essential Guide to Docker Compose Simplify the Development and Orchestration of Multi-Container Applications / E. Gkatziouras. — Birmingham : Packt Publishing, Limited, 2022. — 264 p.

3. Nickoloff J. and others Docker in action / J. Nickoloff, S. Kuenzli, B. Fisher. — Shelter Island, NY : Manning Publications Co, 2019. — 310 p.

4. Laster B. Jenkins 2: Up and Running: Evolve Your Deployment Pipeline for Next Generation Automation / B. Laster. — Sebastopol, CA : O'Reilly Media, 2018. — 577 p.

5. Culkin J. and other. AWS cookbook: recipes for success on AWS / J. Culkin, M. Zazon. — Beijing Boston Farnham Sebastopol Tokyo : O'Reilly, 2021. — 330 p.

6. Winkler S. and others Terraform in action / S. Winkler, A. Dadgar. — Shelter Island, NY : Manning, 2021. — 380 p.

Додаткова

7. Walls C. Spring in action / C. Walls. — Shelter Island, NY : Manning Publications Co, 2022. — 492 p.

8. Leszko R. Continuous delivery with Docker and Jenkins: delivering software at scale / R. Leszko. — Birmingham Mumbai : Packt Publishing, 2017. — 309 p.

9. Wittig M. and others Amazon Web Services in action / M. Wittig, A. Wittig, B. Whaley. — Shelter Island, NY : Manning, 2019. — 497 p.

10. Wadia Y. and others Mastering AWS Lambda. Learn how to build and deploy serverless applications / Y. Wadia, U. Gupta. — Birmingham : Packt Publishing, Limited, 2017. — 296 p.

11. Brikman Y. Terraform: up & running: writing infrastructure as code / Y. Brikman. — Beijing Boston : O'Reilly, 2021. — 341 p.

Інформаційні ресурси

12. API Documentation & Design Tools for Teams | Swagger [Електроний ресурс]. — Режим доступу: https://swagger.io/.

13. About Swagger Specification | Documentation | Swagger [Електроний ресурс] — Режим доступу: https://swagger.io/docs/specification/about/.

14. Docker: Accelerated Container Application Development [Електроний ресурс] — Режим доступу: https://www.docker.com/.

15. Overview of get started [Електроний ресурс] // Docker Documentation. — Електрон. дані. — Режим доступу: https://docs.docker.com/guides/get-started/.

16. Jenkins [Електроний ресурс] // Jenkins. — Електрон. дані. — Режим доступу: https://www.jenkins.io/.

17. User Handbook Overview [Електроний ресурс] // User Handbook Overview. — Електрон. дані. — Режим доступу: https://www.jenkins.io/doc/book/getting-started/.
18. Terraform by HashiCorp [Електроний ресурс] // Terraform by HashiCorp. — Електрон. дані. — Режим доступу: https://www.terraform.io/.
19. Build a Serverless Web Application with AWS Lambda, Amazon API Gateway, AWS Amplify, Amazon DynamoDB, and Amazon Cognito [Електронний ресурс] // Amazon Web Services, Inc. — URL: https://aws.amazon.com/getting-started/hands-on/build-serverless-web-app-lambda-apigateway-s3-dynamodb-cognito/.
20. Cloud Computing Services - Amazon Web Services (AWS) [Електронний ресурс]. – Режим доступу: — URL: https://aws.amazon.com/.
21. Welcome to AWS Documentation [Electronic resource] [Електронний ресурс]. – Режим доступу: — URL: https://docs.aws.amazon.com
22. Сучасні технології програмування (6.04.121) [Електронний ресурс] / Розробники Андрій Поляков, Олег Фролов // Персональні навчальні системи ХНЕУ ім. С. Кузнеця — Електрон. дані. — Режим доступу: https://pns.hneu.edu.ua/course/view.php?id=8632.