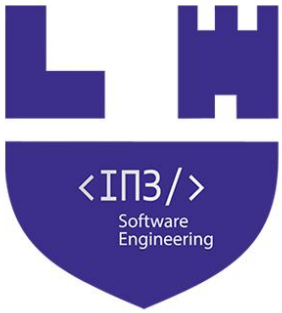
МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Луцький національний технічний університет
Факультет комп'ютерних та інформаційних технологій
Кафедра інженерії програмного забезпечення



АРХІТЕКТУРА ТА ПРОЕКТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Методичні вказівки до виконання самостійної роботи для здобувачів першого (бакалаврського) рівня вищої освіти освітньої програми «Інженерія програмного забезпечення» галузі знань 12 Інформаційні технології спеціальності 121 Інженерія програмного забезпечення денної та заочної форм навчання

2026

УДК 004.9; 004.4(07)
А38

До друку
Голова вченої ради ФКІТ _____І. С. Кондіус
(підпис)
Електронна копія друкованого видання передана для внесення в репозитарій ЛНТУ
директор бібліотеки _____Н. П. Поліщук
(підпис)
Затверджено вченою радою ФКІТ,
протокол №__ від «__» _____ 2026 року.

Розглянуто та схвалено на засіданні кафедри інженерії програмного забезпечення ЛНТУ,
протокол №__ від «__» _____2026 року.
Завідувач кафедри ІПЗ_____Н. М. Ліщина

Укладач _____ (підпис) В. О. Ліщина, кандидат технічних наук, доцент, доцент кафедри інженерії програмного забезпечення ЛНТУ
М. В. Хвищун, кандидат фізико-математичних наук, доцент, доцент кафедри інженерії програмного забезпечення ЛНТУ

Рецензен:_____ (підпис) Ю. Й. Тулашвілі, доктор педагогічних наук, професор, професор кафедри комп'ютерних наук ЛНТУ

Архітектура та проектування програмного забезпечення: методичні вказівки до виконання самостійної роботи для здобувачів першого (бакалаврського) рівня освітньо-професійної програми «Інженерія програмного забезпечення» галузі знань 12 Інформаційні технології спеціальності 121 Інженерія програмного забезпечення денної та заочної форм навчання / уклад. В. О. Ліщина, М. В. Хвищун. Луцьк: ЛНТУ. 2026. 22 с.

У методичних вказівках наведений перелік питань, що охоплюють зміст дисципліни, запропоновано завдання для самостійного опрацювання, завдання для індивідуальних завдань, описано порядок поточного і підсумкового оцінювання знань з дисципліни.

Призначені для студентів спеціальності 121 «Інженерія програмного забезпечення» денної та заочної форми навчання.

ЗМІСТ

ВСТУП

Метою викладання навчальної дисципліни «Архітектура та проектування програмного забезпечення» є формування науково-професійного світогляду бакалавра спеціальності 121 – Інженерія програмного забезпечення в області проектування архітектури програмного забезпечення.

Предмет вивчення – процес проектування архітектури програмного забезпечення, тобто принципи, каркаси, стилі та інструменти створення архітектури програмного забезпечення. Одним з найважливіших етапів створення програмного забезпечення є етап створення архітектури програмного забезпечення. Саме від вибору структурних елементів, їх інтерфейсів, а також особливостей їх взаємодії в першу чергу і залежить надійність, якість, а також безпека створюваного програмного забезпечення. Успішне засвоєння дисципліни дозволяє бакалавру з програмної інженерії розширити коло застосування набутих раніше знань та практичних навичок для вирішення широкого кола задач, пов'язаних з проектуванням архітектури програмного забезпечення.

Найменування та опис компетентностей, формування котрих забезпечує вивчення дисципліни.

Інтегральна компетентність

Здатність розв’язувати складні спеціалізовані завдання або практичні проблеми інженерії програмного забезпечення, що характеризуються комплексністю та невизначеністю умов, із застосуванням теорій та методів інформаційних технологій.

Загальні компетентності

К02. Здатність застосовувати знання у практичних ситуаціях;

К05. Здатність вчитися і оволодівати сучасними знаннями;.

Спеціальні (фахові, предметні) компетентності:

К14. Здатність брати участь у проектуванні програмного забезпечення, включаючи проведення моделювання (формальний опис) його структури, поведінки та процесів функціонування;

К15. Здатність розробляти архітектури, модулі та компоненти програмних систем;

К17. Здатність дотримуватися специфікацій, стандартів, правил і рекомендацій в професійній галузі при реалізації процесів життєвого циклу;

К23. Здатність реалізовувати фази та ітерації життєвого циклу програмних систем та інформаційних технологій на основі відповідних моделей і підходів розробки програмного забезпечення.

Програмні результати навчання:

ПР-03. Знати основні процеси, фази та ітерації життєвого циклу програмного забезпечення;

ПР-04. Знати і застосовувати професійні стандарти і інші нормативно правові документи в галузі інженерії програмного забезпечення;

ПР-05. Знати і застосовувати відповідні математичні поняття, методи доменного, системного і об'єктно-орієнтованого аналізу та математичного моделювання для розробки програмного забезпечення;

П-10. Проводити передпроектне обстеження предметної області, системний аналіз об'єкта проектування;

ПР-11. Вибирати вихідні дані для проектування, керуючись формальними методами опису вимог та моделювання.

ПР-12. Застосовувати на практиці ефективні підходи щодо проектування програмного забезпечення.

ПР-14. Застосовувати на практиці інструментальні програмні засоби доменного аналізу, проектування, тестування, візуалізації, вимірювань та документування програмного забезпечення.

САМОСТІЙНА РОБОТА СТУДЕНТІВ

Необхідним елементом успішного засвоєння матеріалу освітньої компоненти є самостійна робота студентів з вітчизняною та закордонною спеціальною літературою, періодичними виданнями тощо. Основні види самостійної роботи, які запропоновані студентам:

1. Вивчення лекційного матеріалу.

2. Опрацювання та вивчення рекомендованої літератури.

3. Вивчення основних термінів та понять за темами дисципліни.

4. Підготовка до лабораторний занять, модульних контрольних робіт.

5.Контрольна перевірка кожним студентом особистих знань за запитаннями для самоконтролю.

Для оцінки ступеня опрацювання матеріалу, який запропоновано для самостійної роботи застосовується усна бесіда з викладачем за питаннями для самостійної роботи, тестування.

Перелік питань для самостійного опрацювання

1. Основи SOLID-принципів у проектуванні програмного забезпечення. 6 год (денна), 13 год (заочна).

Завдання:

Вивчити основні принципи SOLID:

S (Single Responsibility Principle)

O (Open/Closed Principle)

L (Liskov Substitution Principle)

I (Interface Segregation Principle)

D (Dependency Inversion Principle)

Навести приклади реалізації кожного принципу.

Проаналізувати код із порушенням SOLID та запропонувати виправлення.

2. Порівняльний аналіз монолітної та мікросервісної архітектур. 6 год (денна), 12 год (заочна)

Завдання:

Визначити переваги та недоліки монолітної та мікросервісної архітектур.

Дослідити випадки, коли доцільно використовувати кожен тип архітектури.

Провести SWOT-аналіз для монолітного та мікросервісного підходів.

3. Практичне використання UML-діаграм у реальних проєктах. 6 год (денна), 16 год (заочна)

Завдання:

Побудувати Class Diagram для простої системи управління користувачами.

Створити Sequence Diagram для реєстрації користувача на веб-платформі.

Використати Activity Diagram для представлення бізнес-логіки.

Використати інструменти (Lucidchart, Draw.io, Enterprise Architect).

4. Основи модульного тестування компонентів у програмному забезпеченні. 7 год (денна), 10 год (заочна)

Завдання:

Ознайомитися з принципами модульного тестування.

Дослідити основні фреймворки для тестування (JUnit, PyTest, Mocha).

Написати тести для простого модуля.

5. Вплив вибору методології проектування на подальший розвиток ПЗ. 7 год (денна), 10 год (заочна)

Завдання:

Порівняти методології Waterfall, Agile, Scrum, Kanban.

Визначити їхній вплив на підтримку та розширюваність ПЗ.

Проаналізувати кейси використання у відомих IT-компаніях.

6. Використання Kubernetes для автоматизованого розгортання мікросервісів. 7 год (денна), 14 год (заочна)

Завдання:

Ознайомитися з основними концепціями Docker та Kubernetes.

Розібратися з поняттями Pod, Deployment, Service, Ingress.

Розгорнути простий застосунок у Kubernetes-кластері.

7. Використання патернів проектування у розробці масштабованих веб-додатків. 7 год (денна), 13 год (заочна)

Завдання:

Ознайомитися з GoF-патернами (Factory, Singleton, Adapter, Observer).

Дослідити, які патерни найчастіше використовуються у веб-розробці.

Реалізувати один із патернів у коді.

8. Перехід від монолітної до мікросервісної архітектури: кроки та виклики. 7 год (денна), 10 год (заочна)

Завдання:

Визначити ключові етапи переходу від монолітної до мікросервісної архітектури.

Проаналізувати ризики та проблеми під час міграції.

Ознайомитися з реальними кейсами переходу (Netflix, Uber).

9. Використання автоматизованих інструментів аналізу якості коду. 7 год (денна), 10 год (заочна)

Завдання:

Ознайомитися з основними метриками якості коду (Cyclomatic Complexity, Code Smells, Coupling & Cohesion).

Вивчити SonarQube або аналогічний інструмент для аналізу коду.

Проаналізувати код за допомогою автоматизованого аналізатора.

КОНТРОЛЬНІ ЗАПИТАННЯ ДЛЯ САМОДІАГНОСТИКИ

1. Що таке проектування програмного забезпечення, і яка його роль у життєвому циклі ПЗ?

2. Які основні принципи проектування програмного забезпечення?

3. У чому відмінність між високорівневим (архітектурним) і деталізованим проектуванням?

4. Які існують основні архітектурні стилі програмного забезпечення?

5. У чому різниця між монолітною та мікросервісною архітектурою?

6. Які характеристики архітектури програмного забезпечення є найважливішими?

7. Що таке UML, і які типи UML-діаграм використовуються у проектуванні?

8. У чому полягає роль UML-діаграм у комунікації між командами розробників?

9. Які основні види структурних діаграм UML і їх застосування?

10. Які основні види діаграм поведінки UML і їх застосування?

11. Що таке модуль у контексті проектування програмного забезпечення?

12. Які основні принципи проектування компонентів у програмному забезпеченні?

13. Як забезпечується інтеграція компонентів у єдину систему?

14. Яку роль відіграє документування компонентів і модулів?

15. Які методи та стратегії проектування існують?

16. У чому полягає різниця між функціонально-орієнтованим і об'єктно-орієнтованим проектуванням?

17. Що таке проектування, орієнтоване на компоненти (CBD)?

18. У чому особливості подійно-орієнтованого проектування?

19. Які особливості проектування розподілених систем?

20. Які основні принципи клієнт-серверної взаємодії?

21. Що таке middleware, і яку роль воно відіграє у розподілених системах?

22. Які моделі хмарних обчислень існують (IaaS, PaaS, SaaS)?

23. У чому особливості serverless-архітектури?

24. Як контейнеризація (Docker) впливає на архітектуру ПЗ?

25. Які переваги використання Kubernetes у розробці розподілених систем?

26. Що таке шаблони проектування, і навіщо вони використовуються?

27. Які основні категорії шаблонів проектування існують?

28. Які характеристики мають створювальні шаблони проектування? Наведіть приклади.

29. Які характеристики мають структурні шаблони проектування? Наведіть приклади.

30. Які характеристики мають поведінкові шаблони проектування? Наведіть приклади.

31. Які основні принципи рефакторингу архітектури програмного забезпечення?

32. Як забезпечується масштабування архітектури програмного забезпечення?

33. Які основні підходи до переходу від монолітної до мікросервісної архітектури?

34. Як документується архітектура програмного забезпечення?

35. Які автоматизовані інструменти використовуються для документування архітектури?

36. Які критерії оцінки якості проектування програмного забезпечення?

37. Які метрики використовуються для оцінки якості архітектури програмного забезпечення?

38. Як виявляти та усувати дефекти на етапі проектування?

39. Які стратегії забезпечення якості проектування існують?

ПОРЯДОК ПОТОЧНОГО І ПІДСУМКОВОГО ОЦІНЮВАННЯ ЗНАНЬ З ДИСЦИПЛІНИ

Порядок поточного оцінювання знань

Завданням поточного контролю є перевірка розуміння та засвоєння певного матеріалу, вироблених навичок проектування та створення прикладних програм.

Оцінювання знань студентів проводиться у трьох напрямах:

1. Контроль систематичності та активності роботи на лабораторних заняттях.

2. Контроль виконання завдань для самостійного опрацювання.

3. Контроль виконання завдань модульних контрольних робіт.

Поточний контроль – це оцінювання роботи здобувачів освіти на лабораторних заняттях, за результатами самостійної, індивідуальної робіт, що передбачені навчальним планом (оцінюється в балах, максимально – 100 балів). Об’єкт поточного контролю – процес встановлення рівня досягнень програмних результатів в оволодінні змістом предмету, уміннями та навичками. Ліквідація заборгованостей щодо поточних контрольних заходів може здійснюватися впродовж усього періоду вивчення навчальної дисципліни у семестрі.

Оцінка з поточного контролю визначається як середня арифметична оцінка з усіх навчальних занять та розраховується при оцінюванні після проведення останнього у семестрі навчального заняття.

Бали оцінювання лабораторних робіт нараховуються за наступним співвідношенням:

90–100 – студент в повному обсязі володіє навчальним матеріалом, має повне розуміння розглянутої теми, надає правильні відповіді на запитання по темі, код програми функціонує відповідно до завдання;

75–89 – студент достатньо розуміє розглянутий матеріал та принципи написаного ним коду програми, присутні неточності та незначні помилки у відповідях на запитання по темі, код програми функціонує відповідно до завдання (або з несуттєвими недоліками);

65-74 – студент не досить добре розуміє розглянутий матеріал та написаний ним код програми, вагається та надає неточні/не конкретні відповіді на запитання по темі, код програми функціонує неточно, або з помірними недоліками;

60-64 – студент погано розуміє розглянутий матеріал та написаний ним код програми, студент в більшості надає помилкові відповіді на питання по темі, код програми функціонує з суттєвими недоліками;

35-59 – студент погано розуміє розглянутий матеріал та написаний ним код програми, код програми не функціонує належним чином;

0-34 – студент зовсім не засвоїв розглянутий матеріал, написаний ним код програми не відповідає темі/не функціонує взагалі.

Контроль за виконанням модульних завдань

При виконанні завдань модульних контрольних робіт оцінюванню підлягають теоретичні знання та практичні навички, яких набули студенти після опанування матеріалу дисципліни.

Протягом семестру проводиться дві модульні контрольні роботи щодо перевірки рівня засвоєння знань студентами.

Модульні контрольні завдання містять теоретичні питання. Контрольні завдання складені з урахуванням вимоги однакової складності для всіх студентів.

Модульний контроль містить 30 тестових завдань. Відповідь на кожне питання модульного контролю оцінюється в 1(вірна) або 0(не вірна) балів. Загальна сума балів складає 100 балів.

Тестові завдання охоплюють теоретичний матеріал теми, який вивчаються в межах навчальної дисципліни "Системний аналіз" та згруповані за двома модулями, кожен з яких складається з тестових завдань різного рівня складності.

Тестові завдання розрізняються за принципом побудови відповіді.

Контроль за виконанням завдань для самостійного опрацювання

При контролі виконання завдань для самостійного опрацювання оцінці підлягає самостійне опрацювання окремих питань.

Підсумковий контроль проводиться у формі екзамену в період екзаменаційної сесії згідно сформованого розкладу іспитів.

Результати підсумкового контролю оцінюються за 100-бальною шкалою і включаються в підсумкову оцінку з навчальної дисципліни (освітньої компоненти) як окремий заліковий модуль з відповідним ваговим коефіцієнтом.

Критерії підсумкового оцінювання:

90–100 балів отримує здобувач освіти, який вільно володіє програмним обсягом матеріалу, виявляє і демонструє особисті творчі здібності, вміє самостійно здобувати нові знання, демонструє ґрунтовні знання, вміння та практичні навички; без допомоги викладача знаходить та опрацьовує необхідну інформацію, використовує набуті знання і вміння для прийняття рішень у

нестандартних ситуаціях, переконливо аргументує відповіді, вміє використовувати методи наукового обґрунтування власних рішень, самостійно розкриває власні обдарування й нахили.

85–89 балів отримує здобувач освіти, який вільно володіє програмним обсягом матеріалу, застосовує його на практиці, вільно розв’язує вправи і задачі у стандартних ситуаціях, самостійно виправляє допущені помилки, кількість яких незначна, вміє обґрунтувати на аргументувати свою думку.

75–84 балів отримує здобувач освіти, який вміє зіставляти, узагальнювати, систематизувати інформацію під керівництвом викладача, в цілому, самостійно застосовувати її на практиці; контролювати власну діяльність; виправляти помилки, серед яких є суттєві, добирати окремі аргументи для підтвердження своїх думок.

65–74 балів отримує здобувач освіти, який відтворює значну частину теоретичного матеріалу, демонструє знання і розуміння основних положень з допомогою викладача, поверхнево відтворює і аналізує навчальний матеріал, виправляє помилки, серед яких є значна кількість суттєвих.

60–64 балів отримує здобувач освіти, який володіє навчальним матеріалом на рівні, вищому за початковий, значну частину його відтворює на репродуктивному рівні або володіє матеріалом на рівні окремих фрагментів, що становлять незначну частину навчального матеріалу.

35–59 балів отримує здобувач освіти, який володіє матеріалом на рівні окремих фрагментів, що становлять незначну частину навчального матеріалу.

0–34 балів отримує здобувач освіти, який володіє матеріалом на рівні елементарного розпізнавання і відтворення окремих фактів, елементів, об’єктів.

Підсумковий бал (за 100-бальною шкалою) з дисципліни «Архітектура та проектування програмного забезпечення» визначається як середньозважена величина, залежно від питомої ваги кожної складової залікового кредиту:

	Поточний контроль	Модульний контроль		Підсумковий контроль	Підсумкова оцінка
	Заліковий модуль 1	Заліковий модуль 2	Заліковий модуль 3	Екзамен	
Вагові коефіцієнти	40%	10%	10%	40%	100%
Максимальна кількість балів (за 100 бальною шкалою)	100	100	100	100	100

Оцінювання знань здобувачів освіти здійснюється відповідно до загальних критеріїв паралельно за:

– 4-бальною національною шкалою (позитивні оцінки – «відмінно», «добре», «задовільно» або «зараховано», негативні оцінки – «незадовільно» або «не зараховано»);

– 100-бальною накопичувальною шкалою ЄКТС.

Шкала оцінювання:

За шкалою ЛНТУ	**За національною шкалою**	**За шкалою ECTS**
90–100	відмінно	A (відмінно)
85–89	добре	B (дуже добре)
75-84		C (добре)
65-74	задовільно	D (задовільно)
60-64		E (достатньо)
35-59	незадовільно	FX (незадовільно з можливістю повторного складання)
0-34		F (незадовільно з обов’язковим повторним курсом)

ОРІЄНТОВНІ ТЕСТОВІ ЗАВДАННЯ

1. Що є головною метою проектування програмного забезпечення?

A) Розробка коду без документування

B) Вибір мови програмування

C) Визначення структури та компонентів системи

D) Використання максимальної кількості бібліотек

E) Підвищення складності коду

2. Який з наведених принципів не належить до SOLID?

A) Принцип єдиної відповідальності

B) Принцип відкритості/закритості

C) Принцип централізованої обробки

D) Принцип заміни Барбари Лісков

E) Принцип поділу інтерфейсів

3. Що таке архітектура програмного забезпечення?

A) Конкретна мова програмування, що використовується для розробки

B) Візуалізація UI

C) Сукупність принципів, структур і процесів, що визначають програмну систему

D) Виключно UML-діаграми

E) Вибір середовища розробки

4. Яка архітектура підходить для розподілених систем?

A) Монолітна

B) Мікросервісна

C) Локальна

D) Інкапсульована

E) Одношарова

5. Який з інструментів використовується для моделювання архітектури ПЗ?

A) Docker

B) UML

C) Kubernetes

D) SQL

E) JSON

6. Яке твердження щодо модулів є правильним?

A) Модулі не взаємодіють між собою

B) Кожен модуль повинен бути незалежним від інших

C) Модуль – це логічно завершений компонент системи

D) Модулі мають містити тільки одну функцію

E) Модулі є частиною фізичного серверного обладнання

7. Яка з UML-діаграм використовується для моделювання поведінки системи?

A) Class Diagram

B) Sequence Diagram

C) Deployment Diagram

D) Component Diagram

E) Package Diagram

8. Яке з понять характеризує слабкий зв’язок між модулями?

A) Когезія

B) Зчеплення

C) Інкапсуляція

D) Поліморфізм

E) Наслідування

9. Який принцип проектування підвищує повторне використання коду?

A) Інкапсуляція

B) Наслідування

C) Модульність

D) Поліморфізм

E) Рефакторинг

10. Яка з архітектур є найбільш масштабованою?

A) Монолітна

B) Мікросервісна

C) Клієнт-серверна

D) Функціональна

E) Подієво-орієнтована

11. Що таке middleware у розподілених системах?

A) Набір бібліотек для обробки JSON

B) Програмний компонент для взаємодії між сервісами

C) Мова програмування для архітектурного проектування

D) Тестовий фреймворк

E) Методологія розробки

12. Який архітектурний стиль застосовується у хмарних сервісах?

A) Монолітний

B) Мікросервісний

C) Компонентний

D) ООП

E) Ієрархічний

13. Що таке Docker?

A) Мова програмування

B) Фреймворк для моделювання UML

C) Інструмент для контейнеризації

D) Інструмент для написання SQL-запитів

E) Система контролю версій

14. Який з патернів належить до створювальних?

A) Singleton

B) Adapter

C) Observer

D) Proxy

E) Strategy

15. Який принцип використовується для масштабування системи?

A) Ієрархічна організація коду

B) Вертикальне та горизонтальне масштабування

C) Поліморфізм

D) Інкапсуляція

E) Наслідування

16. Що таке рефакторинг?

A) Заміна всіх компонентів системи

B) Покращення коду без зміни функціональності

C) Додавання нових функцій у програму

D) Перехід на нову мову програмування

E) Створення нової архітектури з нуля

17. Що визначає SLA (Service Level Agreement)?

A) Дизайн користувацького інтерфейсу

B) Рівень послуг, які надає система

C) Внутрішню структуру бази даних

D) Методи проектування API

E) Спосіб організації комунікації у команді

18. Яка перевага використання Kubernetes?

A) Автоматизація розгортання контейнерів

B) Зменшення розміру вихідного коду

C) Вбудована підтримка монолітної архітектури

D) Забезпечення єдиного кодування для всіх сервісів

E) Спрощене тестування інтерфейсів

19. Що таке API Gateway у мікросервісній архітектурі?

A) Сервер баз даних

B) Компонент, що керує запитами між клієнтом і сервісами

C) Модуль для обробки великих даних

D) Механізм збереження стану додатку

E) Інструмент аналізу продуктивності

20. Яка головна мета патерну Singleton?

A) Забезпечити унікальність об'єкта у програмі

B) Створювати новий об'єкт при кожному виклику

C) Автоматично тестувати код

D) Оптимізувати швидкість виконання SQL-запитів

E) Підтримувати зв’язок між різними модулями

21. Який з підходів найкраще підходить для проектування складних програмних систем?

A) Написання коду без попереднього планування

B) Використання компонентного та об’єктно-орієнтованого проектування

C) Виключно тестування без створення модулів

D) Використання тільки структурного програмування

E) Створення складних монолітних додатків

22. Яка особливість архітектурного стилю REST?

A) Використання HTTP-запитів для комунікації між сервісами

B) Вимога запуску сервісів тільки на локальних серверах

C) Відсутність взаємодії між клієнтом та сервером

D) Обов’язкове використання XML для передачі даних

E) Використання монолітного підходу до проектування

23. Яка з діаграм UML використовується для представлення потоків управління та бізнес-логіки?

A) Activity Diagram

B) Class Diagram

C) Deployment Diagram

D) Component Diagram

E) Package Diagram

24. Який рівень зчеплення між модулями є найкращим для підтримуваності коду?

A) Слабке зчеплення (Low Coupling)

B) Жорстке зчеплення (High Coupling)

C) Центральне зчеплення

D) Повне зчеплення між всіма модулями

E) Відсутність зчеплення

25. Яка головна мета архітектурного проектування?

A) Забезпечити структуровану основу для розробки програмного забезпечення

B) Вибрати найсучасніший фреймворк

C) Написати якомога більше коду

D) Мінімізувати використання баз даних

E) Використати найбільшу кількість серверів

СПИСОК РЕКОМЕНДОВАНОЇ ЛІТЕРАТУРИ

1. Бородкіна І. Л. Архітектура та проектування програмного забезпечення. Інженерія програмного забезпечення: Навч. посібник , НУБіП. Київ: Центр учбової літ., 2020. 204 с.
2. Manshreck T., Wright H. Software Engineering at Google: Lessons Learned from Programming Over Time Titus Winters. "O'Reilly Media, Inc.", 2020. 602 p.
3. Lecture notes on software engineering. Course By Dr. H.S.Behera Asst. Prof K.K.Sahu Asst. Prof Gargi Bhattacharjee. URL: https://www.vssut.ac.in/lecture_notes/lecture1428551142.pdf.
4. Software Engineering Tutorial. Simple Easy Learning At Your Fingertips. URL: https://www.tutorialspoint.com/software_engineering/index.htm.
5. Ерік Фрімен, Елізабет Робсон. Патерни проєктування. Фабула, 2020. 672 с.
6. Umesh Kumar Tiwari, Santosh Kumar. Component-Based Software Engineering. Methods and Metrics. Taylor & Francis Group, LLC, 2021. 226 p.
7. Роберт Мартін Чиста архітектура. Мистецтво розробки програмного забезпечення, 2019. 368с.
8. Guide to Software Engineering Base of Knowledge (SWEBOK). URL: https://surl.li/mtfwtz.
9. Len Bass Software Architecture in Practice Addison-Wesley Professional, 4th Edition, 2022, 442 p.
10. Як навчати архітектурі ПЗ. URL: https://dou.ua/forums/topic/3398/.
11. ДСТУ ISO/IEC/IEEE 12207:2018 Інженерія систем і програмних засобів. Процеси життєвого циклу програмних засобів. (ISO/IEC/IEEE 12207:2017, IDT) UTL: https://online.budstandart.com/ua/catalog/doc-page.html?id_doc=77957 .

Архітектура та проектування програмного забезпечення: методичні вказівки до виконання самостійної роботи для здобувачів першого (бакалаврського) рівня освітньо-професійної програми «Інженерія програмного забезпечення» галузі знань 12 Інформаційні технології спеціальності 121 Інженерія програмного забезпечення денної та заочної форм навчання / уклад. В. О. Ліщина, М. В. Хвищун. Луцьк: ЛНТУ. 2026. 22 с.

Комп'ютерний набір *В. Ліщина*

Редактор *В. Ліщина*

Підп. до друку «__» 2025 р. Папір офс.
Гарнітура Таймс. Ум. друк. арк. . Обл.-вид. арк.
Тираж прим.

Відділ іміджу та промоції
Луцького національного технічного університету
43018 м. Луцьк, вул. Львівська, 75
Друк – ВІП ЛНТУ