

**КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
ІМЕНІ ТАРАСА ШЕВЧЕНКА**

ФАКУЛЬТЕТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

Кафедра програмних систем і технологій

«ЗАТВЕРДЖУЮ»

Заступник декана
з навчально-виховної роботи
Наталія ТМЄНОВА

«29» серпня 2026 року

РОБОЧА ПРОГРАМА НАВЧАЛЬНОЇ ДИСЦИПЛІНИ

«МУЛЬТИПЛАТФОРМНЕ ПРОГРАМУВАННЯ»

для здобувачів вищої освіти

галузь знань	12 Інформаційні технології
спеціальність	121 Інженерія програмного забезпечення
освітній рівень	бакалавр
освітня програма	Інженерія програмного забезпечення
вид дисципліни	вибіркова

Форма навчання	денна
Навчальний рік	2026/2027
Семестр	5
Кількість кредитів ECTS	4
Мова викладання, навчання та оцінювання	українська
Форма заключного контролю	залік

Викладачі: **Петрівський Володимир Ярославович, доктор філософії**

Пролонговано: на 2027/2028 н.р. _____ (підпис) _____ (ПІБ) «____» _____ 2027 р. (дата)

на 2028/2029 н.р. _____ (підпис) _____ (ПІБ) «____» _____ 2028 р. (дата)

на 2029/2030 н.р. _____ (підпис) _____ (ПІБ) «____» _____ 2029 р. (дата)

КИЇВ – 2026

Розробник: **Петрівський Володимир Ярославович**, доктор філософії,
асистент кафедри програмних систем і технологій

Завідувач кафедри
програмних систем і технологій


(підпис) проф. Олексій БИЧКОВ

Протокол № 18 від 17 червня 2026 року

Схвалено науково-методичною комісією факультету інформаційних
технологій

Протокол № 11 від 26 червня 2026 року

Голова науково-методичної комісії 
(підпис) Ганна КРАСОВСЬКА

© Петрівський В.Я., 2026 рік
© _____, 20__ рік
© _____, 20__ рік

1. Мета дисципліни – формування у студентів знань та вмінь з проектування програмного забезпечення, володіння технологіями та методологіями розробки та реалізації мультиплатформного програмного забезпечення. Практичні навички розробки, розгортання та супроводження мультиплатформного програмного забезпечення. Розуміння і застосування програмного багато-компонентного програмного забезпечення та організації взаємодії між складовими таких систем. Для досягнення даної мети необхідно опанувати систему знань, умінь, отримати практичні (спеціалізовані) навички, опанувати типові методи та техніки досягнення мультиплатформного дизайну. Крім того, відпрацювання умінь і навичок в області мультиплатформного програмування дозволяє реалізувати розвиваючу функцію освіти: креативну та логічну складові мислення.

2. Попередні вимоги до опанування або вибору навчальної дисципліни:

1. Знати – методи розробки програмного забезпечення; методи проектування логічної моделі бази даних; про якість програмного забезпечення та його тестування;
2. Вміти – використовувати дискретні структури та сучасні методи дискретної математики і аналізу; застосовувати методи об'єктно-орієнтованого конструювання програм, виділяючи інтерфейси і реалізації та забезпечуючи взаємодію між модулями, підсистемами і компонентами; створювати фізичну модель даних та проводити її реорганізацію, створювати та використовувати індекси;
3. Володіти – навичками роботи в групах, елементарними навичками обчислень; навичками абстрактного, алгоритмічного, логічного мислення, аналізу та синтезу, формулювання вимоги та завдання, знаходити методи і підходи до їх розв'язання.

3. Анотація навчальної дисципліни: Дисципліна «Кроссплатформне програмування» присвячена отриманню студентом компетенцій для того, щоб приймати участь у розробці мультиплатформного програмного забезпечення у стеку технологій Microsoft .Net. Розглядаються можливості мови програмування Microsoft .Net, основи технології Microsoft .Net, інструментальні засоби IDE для розробки та розгортання програмних продуктів, що відповідають сучасним вимогам. Дисципліна включає засоби Microsoft .Net для роботи із структурами даних, роботи із файловою системою, створення розвиненого GUI, розробки клієнт-серверних додатків, створення додатків для мобільних платформ та інші. Також в основі дисципліни лежить практичне застосування об'єктно-орієнтованого підходу при проектуванні складних програмних систем.

4. Завдання вивчення дисципліни:

- ознайомити студентів з основами технології Microsoft .Net Core, можливостями мови програмування та інструментальних засобів Visual Studio Code та Microsoft Visual Studio;

- закріпити здатність студентів до об'єктно-орієнтованого мислення, а також уміння застосовувати об'єктно-орієнтований підхід при проектуванні складних програмних систем;
- закріпити здатність студентів до використання сучасних операційних систем, взаємодії з системами вводу-виводу, файловими системами та мережевими ресурсами;
- закріпити знання про якість програмного забезпечення та його тестування: планування, розробка тестів, методів проведення.
- навчити студентів сучасним технологіям та інструментальним засобам розробки мультиплатформенних програмних систем на платформі Microsoft .Net Core, а також навчити застосовувати їх на всіх етапах життєвого циклу програмного продукту.

Для цього дисципліна передбачає формування у студентів знань про принципи Microsoft .Net Core технології та їх використання при розробленні прикладного програмного забезпечення інформаційних систем і технологій; принципи ООП та об'єктну модель, яка використовується на платформі Microsoft .Net Core; синтаксис мови програмування Microsoft .Net Core, її основні конструкції; інструментальні засоби розробки програмних систем на платформі Microsoft .Net Core; технології ефективного використання структур даних; технології створення графічного інтерфейсу користувача (GUI); технологій багатопотокового програмування; технології розгортання та верифікації готового програмного забезпечення за допомогою пакетних менеджерів Nuget, контейнеризації Docker та віртуалізації.

У результаті вивчення навчальної дисципліни студент повинен виробити вміння застосовувати мову програмування Microsoft C# та інструментальні засоби при проектуванні та створенні інформаційних систем, продуктів і сервісів інформаційних технологій; застосовувати об'єктно-орієнтований підхід при проектуванні складних програмних систем; реалізовувати та тестувати компоненти програмного забезпечення, технології створення GUI, що відповідають сучасним вимогам до інтерфейсів користувача; створювати багатопотокові додатки.

Набуття загальних компетентностей (ЗК):

ЗК1. Здатність до абстрактного мислення, аналізу та синтезу.

ЗК2. Здатність застосовувати знання у практичних ситуаціях.

Набуття фахових компетентностей (ФК):

ФК1. Здатність ідентифікувати, класифікувати та формулювати вимоги до програмного забезпечення.

ФК2. Здатність брати участь у проектуванні програмного забезпечення, включаючи проведення моделювання (формальний опис) його структури, поведінки та процесів функціонування.

ФК4. Здатність формулювати та забезпечувати вимоги щодо якості програмного забезпечення у відповідності з вимогами замовника, технічним

завданням та стандартами.

ФК6. Здатність аналізувати, вибирати і застосовувати методи і засоби для забезпечення інформаційної безпеки.

ФК8. Здатність застосовувати фундаментальні і міждисциплінарні знання для успішного розв'язання завдань інженерії програмного забезпечення.

ФК12. Здатність здійснювати процес інтеграції системи, застосовувати стандарти і процедури управління змінами для підтримки цілісності загальної функціональності і надійності програмного забезпечення.

ФК13. Здатність обґрунтовано обирати та освоювати інструментарій з розробки та супроводження програмного забезпечення.

ФК15. Здатність розуміти процес побудови архітектури програмної системи та архітектурні стилі та застосовувати інструменти проєктування архітектури системи, прототипувати та оцінювати архітектурні рішення.

ФК17. Здатність створювати та супроводжувати захищене програмне забезпечення відповідно до вимог сучасних стандартів кібербезпеки та критеріїв безпеки захищених програм.

5. Результати навчання за дисципліною:

Результат навчання (1. знати; 2. вміти; 3. комунікація; 4. автономність та відповідальність)		Форми (та / або методи і технології) викладання і навчання	Методи оцінювання та пороговий критерій оцінювання	Відсоток у підсумковій оцінці з дисципліни
Ко д	Результат навчання			
1.1	Знати предмет та завдання дисципліни. Знати класифікацію мультиплатформеного програмування.	Лекція, словесний метод (дискусія, бесіда), наочний метод (метод ілюстрацій і метод демонстрацій), проблемне навчання, дослідницький метод, парне програмування (вирішення практичних завдань під час інтерактивних сесій), самостійна робота (розв'язання	Виконання лабораторних робіт, модульний контроль, комплексна контрольна робота, усний контроль (усне опитування); письмовий контроль (модульні контрольні роботи), виконання практичних робіт	2%
1.2	Знати елементи та структуру .Net Core проєкту.			1%
1.3	Знати способи застосування пакетного менеджера Nuget. Знати способи керування залежностями проєкту .Net Core.			2%
1.4	Знати способи тестування та розгортання мультиплатформних застосунків. Знати особливості взаємодії з файловою системою мультиплатформних застосунків.			2%
1.5	Знати способи захисту даних мультиплатформних застосунків з веб інтерфейсом. Протокол OAuth2, Open ID, CORS.			2%
1.6	Знати особливості взаємодії з Базами Даних. Знати способи тестування та розгортання мультиплатформних API застосунків. Знати REST принципи.			2%

1.7	Знати основні прийоми побудови графічного інтерфейсу для API застосунків. Знати способи версіонування API інтерфейсів. Знати принципи побудови маршрутизації в REST сумісних застосунках. Знати HTTP протокол. Знати засоби побудови WCF сервісів та концепцію ABC. Знати способи версіонування API інтерфейсів.	програмних завдань)		2%
1.8	Знати методи застосування контейнерів при розробці мультиплатформенного програмного забезпечення.			2%
1.9	Знати способи трасування та відладки. Способи моніторингу то збору телеметричних даних застосунків.			2%
1.1 0	Знати методи створення мультиплатформенних інсталяційних пакетів.			2%
1.1 1	Знати методи побудови мультиплатформних консольних, веб, настільних та мобільних застосунків.			2%
1.1 2	Знати методи комбінування .Net Core з іншими програмними мовами при розробці мультиплатформенних веб-додатків.			2%
2.1	Вміти розробляти Microsoft .Net Core консольні, настільні, веб та мобільні застосунки, мультиплатформні інсталяційні пакети. Вміти програмувати на мові C#.			25%
2.2	Вміти розробляти компоненти, бібліотеки, керувати станами системи, працювати з середовищем та файловою системою.			10%
2.3	Вміти застосовувати Nuget пакетного менеджера. Застосування віртуальних машин. Знати контейнеризацію мультиплатформених веб-застосунків, Налаштовувати відладку та конфігурувати телеметричних сервісів для мультиплатформених додатків. OpenTrace стандарт.			10%

Код	Критерій	Вага	Об'єктивне правило нарахування рівня (у %, 0–100)
А. Процес і дисципліна (індивід.)			
A1	Дотримання проміжних дедлайнів (3 чекпоінти)	6	Рівень за дискретною шкалою (не частка): усі 3 чекпоінти вчасно з повним переліком артефактів → 1.0; 1 етап із затримкою ≤24 год → 0.8; 2 з 3 вчасно → 0.6; 1 з 3 → 0.4; жодного або постфактум → 0.
A2	Ведення репозиторію / логбука	5	+0.2 за кожен виконаний пункт (перевірка Git-історії): (1) ≥1 змістовний коміт/тиждень; (2) коміти прив'язані до issue; (3) README з інструкцією запуску; (4) лог рішень ≥5 записів «проблема→рішення»; (5) гілкова модель (не все в main).
A3	Опрацювання зворотного зв'язку	4	Рівень = частка зауважень із чекпоінтів, оформлених як issue і закритих до наступного етапу. 1.0 ≥90%; 0.8 = 70–89%; 0.6 = 50–69%; 0.4 = 30–49%; 0 <30%.
В. Внесок і надійність (індивід.)			
B1	Виконання індивідуальної зони відповідальності	12	Зона фіксується на Зрізі 0. Рівень оцінюється за сукупністю артефактів (ланцюг issue → branch → PR → коміти → review → merge; код, архітектурні рішення, тести, документація, debugging); story points — лише один із доказів, не самоціль. Повний обсяг якісно → 1.0; переважно (незначні прогалини) → 0.8; частково → 0.6; фрагментарно → 0.4; мінімально → 0.2; не виконано → 0. Авторство визначається за виконаною роботою, а не git blame.
B2	Виконання зобов'язань і комунікація ризиків	8	Рівень = частка взятих задач, які АБО виконані в узгоджений строк, АБО за якими ризик/затримку завчасно (до дедлайну) комуніковано з оновленням issue та новою датою. 1.0 ≥95%; 0.8 = 80–94%; 0.6 = 60–79%; 0.4 = 40–59%; 0.2 = 20–39%; 0 <20%. Мовчазний зрив дедлайну як виконане не зараховується.
B3	Результативність code-review	5	Зараховуються лише результативні рев'ю (провів рев'ю чужого PR → знайшов суттєву проблему або запропонував обґрунтоване покращення → автор відреагував → перевірено виправлення). ≥3 результативних → 1.0; 2 → 0.8; 1 → 0.5; 0 → 0. Кількість дрібних косметичних коментарів не враховується.
С. Технічна якість продукту (КОМАНДНЕ — однакове всім)			
C1	Функціональна коректність (приймальні кейси)	12	Рівень = пройдені приймальні тест-кейси / усього узгоджених кейсів (фіксований список кейсів). Прямий розрахунок частки.

C2	Обробка крайніх випадків і помилок	5	+0.2 за кожен: (1) валідація введення; (2) обробка порожніх/некоректних даних; (3) обробка винятків без аварійного завершення; (4) межові значення; (5) відсутність необроблених винятків у лог-файлі прогону.
C3	Автотести і покриття	6	1.0 = автотести + покриття $\geq 80\%$ (звіт інструмента) + CI гаяє тести; 0.8 = автотести + покриття $\geq 60\%$; 0.6 = автотести + покриття $\geq 40\%$; 0.4 = автотести є, покриття $< 40\%$; 0 = автотестів немає.
C4	Якість коду за статичним аналізом	4	Critical/major зауважень лінера-аналізатора на 1000 рядків: 0 $\rightarrow$ 1.0; $\leq 2 \rightarrow 0.8$; $\leq 5 \rightarrow 0.6$; $\leq 10 \rightarrow 0.4$; $> 10 \rightarrow 0$. Інструментальна перевірка (pylint/ESLint/SonarQube).
C5	Цілісність і безпека роботи з даними / схеми БД	3	+0.2 за кожен застосовний пункт: (1) нормалізація $\geq 3NF$ або обґрунтована денормалізація; (2) РК/ФК та обмеження цілісності; (3) індекси на полях пошуку/JOIN; (4) параметризовані запити (захист від ін'єкцій); (5) транзакції де потрібно. Для проєктів без БД — еквівалент для файлових даних (валідація, транзакційність, узгодженість).
D. Складність зони відповідальності (індивід.)			
D1	Складність і повнота індивідуальної зони відповідальності	10	Зона фіксується на Зрізі 0 (для індивідуальних проєктів — весь проєкт). Рівень = складність зони $\times$ повнота якісного виконання: експертна зона повністю виконана (event-driven backend, складна авторизація, розподілені воркери тощо) $\rightarrow$ 1.0; просунута $\rightarrow$ 0.9; базова, повністю та якісно виконана $\rightarrow$ 0.8; зона виконана частково/спрощено $\rightarrow$ 0.4–0.5; тривіальний внесок (напр. README + шаблонний CRUD) $\rightarrow$ 0.2; не виконано $\rightarrow$ 0. Добре виконаний базовий рівень НЕ карається.
Е. Захист і розуміння (індивід.)			
E1	Відповіді за відкритим банком питань	12	Рівень = сума за відповіді / максимум. Студент отримує К питань з оприлюдненого банку; кожна відповідь: повна = 1, часткова = 0.5, немає = 0.
E2	Модифікація умови «на льоту»	5	2 заздалегідь заготовлені мікрозміни умови; кожна коректно пояснена/реалізована = 0.5. Рівень = 0 / 0.5 / 1.0.
E3	Демонстрація запуску за README	3	Запускається з нуля за README без втручання автора у визначений ліміт часу $\rightarrow$ 1.0; із дрібним втручанням $\rightarrow$ 0.5; не запускається $\rightarrow$ 0.
РАЗОМ (сума ваг)		100	Підсумок = $\text{MIN}(100; M \times \Sigma(\text{вага} \times \text{рівень}) + \text{Бонус})$

7.1.1 Академічна доброчесність і політика щодо ШІ

Доброчесність застосовується як зовнішній множник M до підсумкового бала – Оцінка = $M \times \text{MIN}(100; \Sigma + \text{Бонус})$:
 $M = 1.0$ – порушень не виявлено.

$M = 0.5$ – встановлене порушення: плагіат, запозичення коду/викладок без атрибуції, недеклароване використання згенерованого коду. Застосовується за рішенням згідно з політикою кафедри.

$M = 0.0$ – груба фальсифікація результатів або видавання чужої роботи за власну; підсумок = 0 (навіть із бонусом), робота передається на розгляд комісії.

Використання інструментів метапрограмування в широкому сенсі, зокрема ІІІ-інструментів генерації коду та інших інструментів, пов'язаних з генерацією і автоматичною обробкою текстів формальними і неформальними мовами в контексті процесу розробки ПЗ у процесі виконання проєктної роботи є дозволеним (і, залежно від завдання роботи, може бути рекомендованим) за таких обов'язкових умов:

(1) у файлі META-USAGE студент декларує застосовані принципи (мета)програмування, їх переваги і обмеження у контексті проєктної роботи, а також конкретні застосовані інструменти, ключові запити до них, згенеровані фрагменти коду / інші артефакти;

(2) на захисті студент демонструє володіння матеріалом файлу META-USAGE, зокрема, здатний відповісти на додаткові питання щодо обраної стратегії застосування згаданих інструментів та її порівняння з альтернативними стратегіями, що впливають з раніше пройденого матеріалу даної дисципліни та з попередніх дисциплін;

(3) на захисті студент демонструє розуміння змісту згенерованих фрагментів коду / артефактів, способів перевірки їх якостей (наприклад, коректності, ефективності і т.д.), можливих шляхів їх подальшого використання.

Недекларований згенерований результат кваліфікується як порушення доброчесності (див. вище). Нездатність пояснити власний результат безпосередньо знижує бали блоку Е (захист) за відповідними правилами.

7.2 Індивідуальна складова оцінки

Індивідуальна оцінка обчислюється як сума балів за індивідуальні лабораторні роботи та балів за модульні контрольні роботи.

Розподіл балів за видами індивідуальної діяльності:

Вид діяльності	Кількість балів
Лабораторні роботи	10
Модульна контрольна робота 1	5
Модульна контрольна робота 2	5
Разом	20

Теми лабораторних робіт:

Тема лабораторної роботи	Кількість балів
ЛР1. Entity Framework Core та незалежність від СУБД.	2
ЛР2. ASP.NET Core REST API.	2
ЛР3. Authentication, Authorization та безпека API.	2
ЛР4. Мультиплатформний клієнт на .NET MAUI.	2
ЛР5. Контейнеризація та мультиплатформне розгортання.	2
Разом	10

7.3 Шкала відповідності оцінок

Максимальна кількість балів, яку може отримати здобувач за роботу в семестрі не перевищує 100 балів. Форма підсумкового оцінювання – залік. Залік проводиться шляхом виставлення результуючої оцінки, що визначається як сума балів за всіма успішно оціненими результатами навчання. За умови отримання результуючої підсумкової кількості балів від 60 і вище здобувачу виставляється зараховано. Для здобувачів, які набрали сумарно меншу кількість балів, ніж 60 балів, для одержання заліку необхідно виконати та успішно захистити невиконані компоненти оцінки.

За 100-бальною шкалою	За національною шкалою
60-100	Зараховано / Passed
0-59	Не зараховано / Fail

8. Структура навчальної дисципліни

№ п/п	Назва теми	Кількість годин		
		лекції	лабораторні	СР
1.	Мультиплатформне програмування та сучасна платформа .NET. CLR, SDK, Runtime, CLI, RID, LTS/STS.	2	0	0
2.	Сучасне програмування мовою C#: generics, LINQ, records, nullable types, asynchronous programming	2	2	2
3.	Архітектура .NET solution. Assemblies, NuGet, Dependency Injection, configuration та dependency management	2	2	6
4.	Файлова система, JSON serialization, configuration, secrets та мультиплатформна робота з ресурсами	2	2	6

5	ASP.NET Core. Kestrel. HTTP. Routing. Controllers та Minimal API. Побудова REST API	4	4	6
6	Робота з даними. LINQ, Entity Framework Core, migrations, DB providers, asynchronous data access	4	4	6
7	Безпека .NET-застосунків. Authentication, Authorization, JWT, OAuth 2.0, OpenID Connect, CORS	2	2	6
	Модульна контрольна робота 1		2	
8	Проектування HTTP API. DTO, validation, OpenAPI, API versioning, REST і огляд gRPC	2	2	6
9	Контейнеризація та мультиплатформне розгортання .NET-застосунків. Docker, Compose, WSL, OCI.	2	2	6
10	Якість та observability. Unit/Integration Testing, Logging, Metrics, Tracing, OpenTelemetry	2	2	6
11	Мультиплатформний GUI у .NET MAUI. XAML, MVVM, Binding, Commands, Navigation, Adaptive UI	4	2	6
12	Інтеграція компонентів, publishing, packaging, CI/CD та сучасні тенденції .NET-розробки	2	2	4
	Модульна контрольна робота 2		2	
	ВСЬОГО	30	30	60

Загальний обсяг 120 год., в тому числі:

Лекції – 30 год.;

Лабораторні заняття – 30 год.;

Самостійна робота – 60 год.

9. Рекомендовані джерела:

1. C# - творчість програмування. Том 1. Об'єктно-орієнтоване програмування: підручник \ \ Під ред. Бичкова О.С. Авторський колектив: Іванов Є.В., Жабська Є.О, Петрівський В.Я., Меркулова К.В., Шатирко А.В., Порєв Г.В.: Волин. Обереги, 2024.-292 с. ISBN 978-617-8161-30-9.

2. Albahari, J. C# 12 in a Nutshell: The Definitive Reference. O'Reilly Media. 2023.

3. Price, M. J. C# 12 and .NET 8 – Modern Cross-Platform Development Fundamentals. Packt Publishing. 2024.

4. Price, M. J. C# 14 and .NET 10 – Modern Cross-Platform Development Fundamentals: Build modern websites and services with ASP.NET Core, Blazor, and EF Core using Visual Studio 2026. Packt Publishing. 2025.

5. Joseph Albahari. C# 12 in a Nutshell: The Definitive Reference. O'Reilly Media. 2023 ISBN-1098147448.

6. Griffiths, I. Programming C# 10: Build Cloud, Web, and Desktop Applications. O'Reilly Media. 2022.

7. Gabriel Baptista, Francesco Abbruzzese. Software Architecture with C# 14 and .NET 10: Build enterprise applications using microservices, DevSecOps, EF Core, and design patterns for Azure. Packt Publishing. 2026. ISBN-1806663554.

8. Liberty, J., Juarez, R., Montaquila, M. NET MAUI for C# Developers: Build cross-platform mobile and desktop applications. Packt Publishing. 2023.

9. Byrne, G. Target C#: Simple Hands-On Programming with Visual Studio 2022. Apress. 2022.

10. Kajetan Duszyński. .NET 10 Revealed Master the Latest Advancements in .NET. Apress. 235.

10. Додаткові ресурси

1. MSDN, онлайн документація (C#):
<https://docs.microsoft.com/uk-ua/welcome-to-docs>

2. VirtualBox, онлайн документація:
<https://www.virtualbox.org/wiki/Documentation>

3. Vagrant, онлайн документація: <https://www.vagrantup.com/docs>