**МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ**
**УКРАЇНСЬКА ФЕДЕРАЦІЯ ІНФОРМАТИКИ**
**УКРАЇНСЬКА АСОЦІАЦІЯ ФАХІВЦІВ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ**
**VYTAUTAS MAGNUS UNIVERSITY (КАУНАС, ЛИТВА)**
**DANUBIUS UNIVERSITY (ГАЛАТІ, РУМУНІЯ)**
**КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ КУЛЬТУРИ І МИСТЕЦТВ**
**КИЇВСЬКИЙ УНІВЕРСИТЕТ КУЛЬТУРИ**
**ХАРКІВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ РАДІОЕЛЕКТРОНІКИ**
**ДЕРЖАВНИЙ УНІВЕРСИТЕТ ІНФРАСТРУКТУРИ ТА ТЕХНОЛОГІЙ**

# МАТЕРІАЛИ


**23-24 квітня 2025 р.**

**КИЇВ – 2025**

**МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ**
**УКРАЇНСЬКА ФЕДЕРАЦІЯ ІНФОРМАТИКИ**
**УКРАЇНСЬКА АСОЦІАЦІЯ ФАХІВЦІВ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ**
**VYTAUTAS MAGNUS UNIVERSITY (КАУНАС, ЛИТВА)**
**DANUBIUS UNIVERSITY (ГАЛАТІ, РУМУНІЯ)**
**КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ КУЛЬТУРИ І МИСТЕЦТВ**
**КИЇВСЬКИЙ УНІВЕРСИТЕТ КУЛЬТУРИ**
**ХАРКІВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ РАДІОЕЛЕКТРОНІКИ**
**ДЕРЖАВНИЙ УНІВЕРСИТЕТ ІНФРАСТРУКТУРИ ТА ТЕХНОЛОГІЙ**

# ІНФОРМАЦІЙНІ ТЕХНОЛОГІЇ В СОЦІОКУЛЬТУРНІЙ СФЕРІ, ОСВІТІ ТА ЕКОНОМІЦІ

**МІЖНАРОДНА**
**НАУКОВО-ПРАКТИЧНА КОНФЕРЕНЦІЯ**
**СТУДЕНТІВ І МОЛОДИХ УЧЕНИХ**
**23-24 квітня 2025 р.**

## МАТЕРІАЛИ КОНФЕРЕНЦІЇ

Київ – 2025

**ЗМІСТ**

*УДК 004.4*

***Вдовенко Д. О.***
*здобувач освітнього ступеня бакалавр,*
*спеціальність 122 «Комп'ютерні науки» кафедри ІТШК,*
*Національний університет харчових технологій, м. Київ, Україна*

***Сєдих О. Л.***
*старший викладач кафедри ІТШК,*
*Національний університет харчових технологій, м. Київ, Україна*

## ПОРІВНЯННЯ АРХІТЕКТУРНИХ ПАТЕРНІВ MVC, MVP І MVVM У РОЗРОБЦІ МОБІЛЬНИХ ЗАСТОСУНКІВ

Вибір архітектурного патерну є ключовим етапом у розробці мобільних застосунків, оскільки він визначає структуру коду, тестованість та масштабованість проєкту. Найбільш популярними архітектурами для мобільних застосунків є MVC, MVP та MVVM. Кожен з цих патернів має свої переваги та недоліки, які необхідно враховувати в залежності від складності проєкту.

Одним із найпоширеніших патернів є Model-View-Controller (MVC), який поділяє застосунок на три компоненти: Model, що відповідає за збереження даних і реалізацію бізнес-логіки, View, яка займається відображенням інформації для користувача, та Controller, що керує обміном даними між ними. Такий підхід добре працює у простих проєктах, але тісний зв'язок між компонентами ускладнює підтримку та тестування.

Альтернативним варіантом є Model-View-Presenter (MVP), який усуває обмеження MVC, розділяючи логіку керування та відображення. Model у цій архітектурі, як і раніше, зберігає дані, а View лише відображає їх, проте основна взаємодія між ними відбувається через Presenter. На відміну від Controller у MVC, Presenter не просто обробляє запити, а повністю керує логікою представлення, що підвищує модульність і тестованість коду.

Ще одним кроком у розвитку архітектурних патернів є Model-View-ViewModel (MVVM). У цьому підході зв'язок між Model і View забезпечує ViewModel, який відповідає за підготовку даних і їхню трансформацію для відображення. Завдяки використанню механізму data binding View автоматично оновлюється при зміні даних у Model, що зменшує розмір коду для керування станом UI. Це робить MVVM особливо ефективним у великих застосунках із складними інтерфейсами.

*Таблиця 1. Порівняння архітектурних патернів MVC, MVP та MVVM*

| Характеристика | MVC | MVP | MVVM |
|---|---|---|---|
| **Компоненти** | Model, View, Controller | Model, View, Presenter | Model, View, ViewModel |
| **Роль компонента** | Model: зберігає дані та бізнес-логіку. View: відображає дані та взаємодіє з користувачем. Controller: обробляє ввід користувача та взаємодіє з Model і View. | Model: зберігає дані та бізнес-логіку. View: відповідає за відображення інтерфейсу. Presenter: обробляє введення, взаємодіє з Model і View, виконує логіку. | Model: зберігає дані та бізнес-логіку. View: відображає інтерфейс та отримує введення від користувача. ViewModel: надає дані для View та керує їх відображенням через data binding. |
| **Зв'язок між компонентами** | Тісний зв'язок між компонентами, Controller керує View і Model. | Presenter виступає посередником між View і Model. | View і ViewModel зв'язані через data binding, що забезпечує легку взаємодію без безпосереднього зв'язку. |
| **Масштабованість** | Підходить для малих проєктів через високу залежність між компонентами. | Гнучкіший за MVC, підходить для застосунків середньої та високої складності. | Найкраще підходить для масштабованих застосунків завдяки розділенню логіки та гнучкості. |
| **Тестованість** | Обмежена тестованість через в наслідок тісного зв'язку між компонентами. | Вища тестованість, хоча зв'язок між View і Presenter може ускладнювати процес. | Найвища тестованість завдяки розділенню View і ViewModel та використанню data binding. |
| **Оптимальне використання** | Для невеликих застосунків з простою логікою. | Для застосунків середньої та високої складності. | Для великих і складних проєктів із необхідністю чіткої структури та масштабованості. |
| **Модифікація та зміни** | Важко змінювати через тісний зв'язок компонентів. | Простішим за MVC, але зміни в Presenter можуть ускладнити процес. | Найгнучкіший завдяки чіткому розділенню компонентів і data binding. |

Вибір архітектурного патерну визначається складністю та масштабом застосунку. MVC є доцільним для простих проєктів, проте його

тестованість і масштабованість обмежені. MVP забезпечує вищу модульність, однак може ускладнити розробку великих застосунків. MVVM є найгнучкішим підходом завдяки data binding, що спрощує роботу з інтерфейсом, хоча цей патерн потребує ретельного налагодження.

**СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ**

1. Difference Between MVC, MVP and MVVM Architecture Pattern in Android. *GeeksforGeeks*. URL: https://www.geeksforgeeks.org/difference-between-mvc-mvp-and-mvvm-architecture-pattern-in-android (date of access: 03.04.2025).