4. Антонов Ю. С. Оцінка повноти відповідей в автоматизованих системах контролю знань. *Наукові праці Донецького національного технічного університету. Сер.: Інформатика, кібернетика та обчислювальна техніка.* 2012. № 15. С. 113–117.

5. Антонов Ю. С., Космінська О. М. Методика аналізу тестових завдань на основі отриманих результатів тестування. *Інформаційні технології і засоби навчання.* 2009. № 4. DOI: 10.33407/itlt.v12i4.81. URL: https://journal.iitta.gov.ua/index.php/itlt/article/view/81/

УДК 004.42:004.43

Чернишенко Я. А., *здобувач вищої освіти 4 курсу спеціальності 122 Комп'ютерні науки,*
Зелінська О. В., *канд. техн. наук, доцент,*
в. о. завідувача кафедри інформаційних технологій

ЗАСТОСУВАННЯ ШАБЛОНУ MVVM У РЕАЛІЗАЦІЇ UI-ЛОГІКИ ДЕСКТОПНИХ ДОДАТКІВ

Донецький національний університет імені Василя Стуса, м. Вінниця

У сучасній розробці десктопних додатків шаблон Model-View-ViewModel (MVVM) став одним із ключових підходів до організації логіки користувацького інтерфейсу (UI). Він спрямований на чітке розділення графічного інтерфейсу (View) та прикладної логіки, що дає змогу уникнути надмірної залежності між ними. Завдяки MVVM інтерфейс користувача визначається окремо від коду, який обробляє дії користувача та оперує даними додатка. Такий підхід підвищує зрозумілість структури програми та закладає основу для більш керованої і масштабованої розробки [1].

Сьогодні MVVM фактично став стандартом проєктування інтерфейсної логіки у багатьох сучасних фреймворках для десктопних застосунків. Зокрема, у технологіях на кшталт Windows Presentation Foundation (WPF) та сучасних кросплатформних фреймворках – Avalonia, .NET MAUI [1]. Цей шаблон використовується для побудови чіткого розмежування між відображенням даних і бізнес-логікою. Популярність MVVM пояснюється тим, що він розв'язує типові проблеми розробки UI:

- ускладнення підтримки коду під час росту додатка;
- труднощі тестування;
- повторного використання компонентів.

Завдяки цьому MVVM залишається актуальним і вважається ефективним підходом для створення складних інтерфейсів користувача у сучасних десктопних застосунках.

Шаблон MVVM передбачає чітке розподілення відповідальностей між трьома складниками:

- Model;
- View;
- ViewModel.

Модель (Model) відповідає за управління даними і бізнес-логікою застосунку, тоді як вигляд (View) містить тільки опис графічного інтерфейсу – вікон, кно-

пок, полів та інших елементів UI. Модель вигляду (ViewModel) виступає посередником між ними: вона отримує дані від Моделі, адаптує їх до потреб інтерфейсу та надає вигляду через властивості, а також містить команди для обробки дій користувача. Така архітектура гарантує, що логіка роботи інтерфейсу ізольована у ViewModel і може змінюватися незалежно від реалізації самого відображення на екрані.

Практичне застосування MVVM демонструє підвищення ефективності розробки та підтримки програмного забезпечення. Завдяки відокремленню інтерфейсу від логіки зміни у користувацькому інтерфейсі можна вносити без ризику порушити роботу бізнес-логіки, і навпаки, що скорочує витрати часу на доопрацювання масштабного застосунку. Модель вигляду, яка не залежна від графічного фреймворка, легко піддається модульному тестуванню, тож ключова логіка інтерфейсу перевіряється автономно від візуальної частини [1]. До того ж використання прив'язки даних автоматизує синхронізацію стану між ViewModel та View, що зменшує обсяг шаблонного коду і кількість потенційних помилок під час взаємодії з елементами інтерфейсу.

У межах технології WPF шаблон MVVM став найпоширенішим та є загальноприйнятою практикою. WPF має розвинену систему двостороннього зв'язування даних і команд (ICommand), що природно підтримує принципи MVVM, де елементи інтерфейсу прив'язуються до властивостей та команд ViewModel замість прямого виклику логіки [2]. Це дає змогу майже повністю відмовитися від розміщення бізнес-логіки у коді форм (code-behind), що забезпечує чистий поділ XAML-розмітки інтерфейсу та програмної логіки. Тому застосунки з використанням WPF, побудовані на MVVM, вирізняються гнучкістю у розвитку, де можна змінювати або розширювати інтерфейс без значних змін у внутрішній логіці, і навпаки [2].

Схожі архітектурні підходи реалізовані і в аналогічних кросплатформових фреймворках – Avalonia та .NET MAUI, які успадковують ключові принципи WPF, зокрема підтримку шаблону MVVM. У Avalonia використовується XAML-подібна розмітка та механізми прив'язування даних і команд, що дає змогу застосовувати знайомі розробникам WPF-підходи під час створення інтерфейсів для різних ОС [3]. Завдяки цьому можна використовувати одну й ту саму ViewModel-логіку для додатків під Windows, Linux і macOS без істотних змін. Натомість .NET MAUI, як наступник Xamarin.Forms, також підтримує MVVM і дає змогу будувати інтерфейс під мобільні й десктопні платформи з єдиною базою коду. До того ж .NET MAUI пропонує альтернативу у вигляді Blazor-компонентів, які реалізують розподілення логіки та UI як у MVVM [3]. Тож попри різні цільові платформи, MVVM зберігає свою роль як уніфікований архітектурний підхід в екосистемі .NET.

Отже, шаблон MVVM затвердився як ефективний спосіб організації логіки інтерфейсу в десктопних додатках. Його застосування дає змогу створювати програмні системи, які легше тестувати, масштабувати та підтримувати у довгостроковій перспективі. MVVM успішно інтегрується з різними сучасними технологіями розробки UI, від класичних платформ Windows до кросплатформних рішень. Збереження чіткого поділу між виглядом і логікою продовжує бути запорукою надійної та гнучкої архітектури інтерфейсу в сучасному програмному забезпеченні.

Список використаних джерел

1. Fuksa M., Speth S., Becker S. MVVM Revisited: Exploring Design Variants of the Model-View-View Model Pattern / *Enterprise Design, Operations, and Computing.* EDOC 2024. Lecture Notes in Computer Science. Vol. 15409. Springer, Cham. DOI: /10.1007/978-3-031-78338-8_9.

2. MvvmCross: Introduction to Model/View/ViewModel pattern for building WPF apps (2023). URL: https://www.mvvmcross.com/documentation/fundamentals/viewmodel-lifecycle

3. Classon I. .NET MAUI: Features, Alternatives, and Future. *Migrating from Xamarin. Forms to .NET MAUI.* Apress, Berkeley, CA. 2025. DOI: /10.1007/979-8-8688-1215-6_2.