

# ПОРІВНЯННЯ ІНСТРУМЕНТІВ SWIFTUI ТА UIKIT ДЛЯ РОЗРОБКИ ІНТЕРФЕЙСУ КОРИСТУВАЧА ДЛЯ IOS-ПРИСТРОЇВ НА МОВІ ПРОГРАМУВАННЯ SWIFT

Вінницький національний технічний університет

## Анотація

*У роботі розглянуто основні переваги та недоліки двох основних інструментів розробки графічного інтерфейсу для iOS на мові програмування Swift: SwiftUI та UIKit. Аналізуються можливості кожного з них, їх гнучкість, продуктивність, зручність використання, а також відповідність сучасним вимогам розробки мобільних застосунків.*

**Ключові слова:** SwiftUI, UIKit, iOS, інтерфейс користувача.

## Abstract

*This work examines the main advantages and disadvantages of two main tools for developing graphical interfaces for iOS in the Swift programming language: SwiftUI and UIKit. It analyzes the capabilities of each of them, their flexibility, performance, ease of use, and compliance with modern requirements for developing mobile applications.*

**Keywords:** SwiftUI, UIKit, iOS, UI.

## Вступ

SwiftUI та UIKit — це два основні фреймворки, які компанія Apple надає для створення інтерфейсів користувача в додатках для iOS, macOS, watchOS та tvOS з використанням мови програмування Swift. Обидва інструменти призначені для побудови сучасних, інтерактивних і візуально привабливих інтерфейсів, однак вони суттєво відрізняються за архітектурою, підходами до розробки та функціональними можливостями.

У роботі проаналізовано ключові особливості SwiftUI та UIKit, розглянуто їхні переваги й недоліки, а також визначено ситуації, в яких доцільніше використовувати той чи інший фреймворк.

## Основна частина

UIKit — це оригінальний фреймворк інтерфейсу користувача Apple, представлений у першій версії iOS. UIKit надає різноманітні функції для створення програм, включаючи компоненти, які можна використовувати для створення основної інфраструктури застосунків для iOS версії 2.0 і вище та для інших ранніх версій операційних систем Apple. Цей фреймворк забезпечує архітектуру вікон і переглядів для реалізації вашого інтерфейсу користувача, інфраструктуру обробки подій для доставки Multi-Touch та інших типів введення у вашу програму, а також основний цикл виконання для керування взаємодією між користувачем, системою та вашою програмою. Це імперативний фреймворк інтерфейсу користувача, заснований на архітектурі Model-View-Controller (MVC). UIKit надає повний набір класів і API для створення і керування інтерфейсами користувача. Протягом багатьох років UIKit розвивався і став стандартом для розробки програм для iOS, пропонуючи надійні функції та широкі можливості налаштування. [1]

SwiftUI — це сучасна структура інтерфейсу користувача Apple, представлена в 2019 році разом із Swift 5.1. Це декларативна структура інтерфейсу користувача, яка дозволяє розробникам створювати інтерфейс користувача за допомогою простого та інтуїтивно зрозумілого синтаксису. Завдяки SwiftUI розробники можуть писати менше коду та створювати більш потужний і гнучкий дизайн інтерфейсу користувача. SwiftUI розроблений для безперебійної роботи зі Swift і використовує всі переваги його функцій, таких як безпека типу та опції. Цей фреймворк є більш дружнім для розробників, які бажають розпочати свій шлях у розробці застосунків на пристрої Apple. [2]

Одна з ключових відмінностей між SwiftUI та UIKit полягає в архітектурі їхніх програмних парадигм. SwiftUI працює за декларативною програмною моделлю, що дозволяє розробникам просто описувати

те, як має виглядати користувацький інтерфейс та як він має поводитися. Цей підхід дозволяє створювати стислий і читабельний код, полегшуючи розуміння та підтримку компонентів інтерфейсу користувача. З іншого боку, UIKit дотримується моделі імперативного програмування, де розробники вказують покрокові інструкції для створення та оновлення компонентів інтерфейсу користувача. Хоча UIKit забезпечує більше контролю та гнучкості над інтерфейсом користувача, він часто вимагає написання додаткового коду та обробки змін стану вручну. Інколи різниця між кількістю програмного коду на UIKit та SwiftUI стає дуже суттєвою. Наприклад, на рисунку 1 зображений код для реалізації функціоналу кнопки на UIKit, яка займає близько 29 стрічок з відступами.

```
1 import UIKit
2
3 class ViewController: UIViewController {
4     override func viewDidLoad() {
5         super.viewDidLoad()
6
7         let button = UIButton(type: .system)
8         button.setTitle("Натисни мене", for: .normal)
9         button.setTitleColor(.white, for: .normal)
10        button.backgroundColor = .blue
11        button.layer.cornerRadius = 10
12        button.titleLabel?.font = UIFont.systemFont(ofSize: 18, weight: .medium)
13        button.contentEdgeInsets = UIEdgeInsets(top: 10, left: 20, bottom: 10, right: 20)
14
15        button.addTarget(self, action: #selector(buttonTapped), for: .touchUpInside)
16
17        button.translatesAutoresizingMaskIntoConstraints = false
18        view.addSubview(button)
19
20        NSLayoutConstraint.activate([
21            button.centerXAnchor.constraint(equalTo: view.centerXAnchor),
22            button.centerYAnchor.constraint(equalTo: view.centerYAnchor)
23        ])
24    }
25
26    @objc func buttonTapped() {
27        print("Кнопка натиснута")
28    }
29 }
```

Рис.1. Реалізація кнопки на UIKit

Якщо порівнювати з кодом кнопки, який був написаний з використанням SwiftUI, можна відчутти значну різницю як в читабельності, так і в кількості коду, а саме 17 стрічок з відступами, який було використано. (рис. 2)

```
1 import SwiftUI
2
3 struct ContentView: View {
4     var body: some View {
5         Button(action: {
6             print("Кнопка натиснута")
7         }) {
8             Text("Натисни мене")
9                 .font(.system(size: 18, weight: .medium))
10                .foregroundColor(.white)
11                .padding(.vertical, 10)
12                .padding(.horizontal, 20)
13                .background(Color.blue)
14                .cornerRadius(10)
15        }
16    }
17 }
```

Рис. 2. Реалізація кнопки на SwiftUI

Загалом, можна виокремити наступні переваги SwiftUI над UIKit:

1. Простий декларативний синтаксис, адже SwiftUI дозволяє описувати UI в коротшому та зрозумілішому коді, замість імперативного підходу UIKit.
2. Спрощене управління станом, де оболонки властивостей (@State, @Binding, @ObservedObject) дають можливість автоматично оновлювати інтерфейс при зміні даних.
3. Автоматична підтримка темного режиму, доступності та локалізації.
4. Інтерактивний попередній перегляд у Xcode Canvas, який дає змогу бачити оновлення інтерфейсу в реальному часі без потреби запускати застосунок.
5. Просте створення адаптивного інтерфейсу, що легко масштабується під різні розміри екранів без Auto Layout.
6. Тісна інтеграція з Combine, що робить більш зручною роботу з асинхронними потоками даних та реактивним програмуванням.

Також, розглянемо переваги UIKit над SwiftUI:

1. Зрілість і стабільність, адже UIKit використовується з перших версій iOS і має широкую документацію та підтримку спільноти.
2. Повна зворотна сумісність починаючи з iOS 2.0, тоді як SwiftUI працює тільки з iOS 13 і новішими версіями.
3. Глибока кастомізація інтерфейсу, що дає більше можливостей для створення складних анімацій, спеціалізованих компонентів і нестандартного UI.
4. Вища продуктивність у складних проєктах тому, що UIKit краще працює з великими наборами даних і складними макетами.
5. Широкий вибір бібліотек і ресурсів завдяки тривалому існуванню UIKit.
6. Точний контроль над рендерингом та поведінкою елементів, що важливо для складних застосунків та ігор.

Незважаючи на значні відмінності між цими інструментами, SwiftUI та UIKit можуть працювати разом у межах одного застосунку. Apple передбачила інструменти для взаємодії та комбінації роботи цих фреймворків. Наприклад, UIKit може містити SwiftUI-компоненти за допомогою UIHostingController, а SwiftUI може використовувати елементи UIKit через UIViewControllerRepresentable. [3]

## Висновки

SwiftUI та UIKit — це потужні фреймворки для створення інтерфейсів користувача в iOS-додатках, кожен з яких має власні переваги й обмеження. SwiftUI пропонує сучасний декларативний підхід до побудови UI, що значно спрощує розробку, особливо на ранніх етапах. Натомість UIKit забезпечує перевірену часом імперативну архітектуру з широкими можливостями налаштування та високим рівнем контролю над інтерфейсом.

Вибір між SwiftUI та UIKit залежить від низки чинників: специфіки проєкту, цільових платформ, вимог до зворотної сумісності, рівня складності інтерфейсу, а також досвіду й уподобань розробника.

SwiftUI доцільно обирати для нових проєктів, орієнтованих на iOS 13 і вище, для швидкого створення прототипів, кросплатформної розробки (iOS, iPadOS, watchOS, macOS) та застосунків із сучасним, адаптивним UI.

UIKit залишається незамінним у випадках, коли потрібна підтримка старіших версій iOS, реалізація складної логіки взаємодії, високопродуктивних анімацій або глибока кастомізація компонентів.

У підсумку, обидва фреймворки є потужними інструментами, які дозволяють створювати привабливі та функціональні інтерфейси користувача для екосистеми Apple. За потреби їх можна ефективно поєднувати в межах одного проєкту, використовуючи сильні сторони кожного.

## СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. Apple. UIKit Documentation. [Електронний ресурс] – Режим доступу: <https://developer.apple.com/documentation/uikit> (дата звернення: 15.03.2025).
2. Apple. SwiftUI Documentation. [Електронний ресурс] – Режим доступу: <https://developer.apple.com/documentation/swiftui> (дата звернення: 16.03.2025).
3. Apple. Interfacing with UIKit. [Електронний ресурс] – Режим доступу: <https://developer.apple.com/tutorials/swiftui/interfacing-with-uikit>. (дата звернення: 16.03.2025).

**Чумак Олексій Віталійович** – студент групи ІІСТ-21б, кафедра автоматизації та інтелектуальних інформаційних технологій, факультет інтелектуальних інформаційних технологій та автоматизації, Вінницький національний технічний університет, м.Вінниця, e-mail: [oleksiychumak2004@gmail.com](mailto:oleksiychumak2004@gmail.com).

**Богач Ілона Віталіївна** – к.т.н., доцент кафедри автоматизації та інтелектуальних інформаційних технологій, факультет інтелектуальних інформаційних технологій та автоматизації, Вінницький національний технічний університет, м.Вінниця, e-mail: [ilona.bogach@gmail.com](mailto:ilona.bogach@gmail.com).

**Chumak Oleksii Vitaliyovich** – student of ІІСТ-21b group, Department of Automation and Intelligent Information Technologies, Faculty of Intelligent Information Technology and Automation, Vinnytsia National Technical University, Vinnytsia, e-mail: [oleksiychumak2004@gmail.com](mailto:oleksiychumak2004@gmail.com).

**Bogach Ilona Vitaliivna** – Associate Professor of Automation and Intelligent Information Technologies Department, Faculty of Intelligent Information Technology and Automation, Vinnytsia National Technical University, Vinnytsia, e-mail: [ilona.bogach@gmail.com](mailto:ilona.bogach@gmail.com).