Хіміч В. О., магістрант,
Чижмотря О. В., ст. викладач
Державний університет «Житомирська політехніка»

МИНУЛЕ, ТЕПЕРІШНЄ ТА МАЙБУТНЄ СПОСОБІВ РОЗРОБКИ КОРИСТУВАЦЬКИХ ІНТЕРФЕЙСІВ ДЛЯ ОС ANDROID

Проаналізовано шлях розвитку способів розробки користувацьких інтерфейсів для ОС Android. Описано переваги нових підходів та способів розробки та зроблено висновки щодо майбутнього цих підходів.

Користувацькі інтерфейси пройшли довгу історію розвитку з самого початку створення ЕОМ. Спочатку взаємодія користувача відбувалася за допомогою строкових інтерфейсів, які керували комп'ютерами, що займали чи не цілу кімнату. Потім комп'ютери стали набувати більшого поширення та набувати популярності серед звичайних користувачів. Однак при цьому інтерфейси залишалися застарілими. Вирішенням цього стала розробка графічних інтерфейсів, з якими ми взаємодіємо і донині.

Хоч інтерфейси для користувача не змінили своєї суті, засоби їх розробки постійно змінюються і покращуються для кожної технології чи операційної системи. Android порівняно молода ОС, але при цьому постійно розвивається, і для розробників теж.

З самого початку користувацькі інтерфейси для застосунку під цю операційну систему мали досить типовий спосіб визначення - опис елементів інтерфейсу через теги в файлах з розширенням .xml, котрі були окремими ресурсами для модулю застосунку. Далі ці файли розмітки мали поєднуватися з програмним кодом через Activity/Fragment в окремих Java-файлах, де писалася логіка для інтерфейсу та обробка подій..

Google розуміли цю проблему і в 2016 році представили новий спосіб розробки інтерфейсів – databinding. Його суть полягала в тому, що для кожного .xml файлу застосунку, де корінним тегом є layout, автоматично генерувався код з елементами інтерфейсу, котрі мали атрибут id, тому тепер доступ до всіх потрібних елементів інтерфейсу в Activity/Fragment здійснювався лише через одну змінну з Binding необхідного екрану. Ще однією перевагою цього способу є те, що можна встановлювати значення для атрибутів елементів До того ж з часом розвитку ОС Android та смартфонів користувацькі інтерфейси стали значно більш складні та навантажені, а головна концепція залишилася на рівні 2008 року.

Однак у 2019 році на GoogleI/O [1] було перевинайдено концепцію та інструменти створення інтерфейсу для Android-застосунків, що отримало назву JetpackCompose. Це декларативний спосіб побудови інтерфейсу, на ідеї модульної розробки інтерфейсів з використанням можливостей головної мови розробки для Android – Kotlin.

З його застосуванням всі елементи інтерфейсу описуються у звичайних функціях з анотацією Composable, що дозволяє ділити інтерфейс на групи елементів, розмір та суть яких обирає сам розробник. Таким чином при роботі з JetpackCompose розробник по суті створює бібліотеку елементів інтерфейсу, необхідних саме його застосунку. Ще однією перевагою JetpackCompose є ідея використання принципу UnidirectionalDataFlow для оновлення стану UI в самій основі технології, що дозволяє значно зменшити кількість багів та проблем, пов'язаних з оновленням даних. Значним покращенням в порівнянні з .xml є те, що при зміні стану UI відбувається рендеринг лише тих Composable функцій, стан яких було змінено, і таким чином, якщо це зміна стану лише певного, навіть малого, елементу, то вона буде стосуватися лише його, зменшуючи навантаження на процесор мобільного пристрою. Разом зі стислістю коду, потрібного для опису інтерфейсу, та акцентом компанії Google, цей інструмент поступово набирає популярність.

Згідно досліджень з цієї статті [2] Compose має більшу продуктивність, коли елементи JetpackCompose підзавантажуються завчасно, наприклад, при переході з одного екрану написаного таким способом на інший. В порівнянні з .xml такий комплексний підхід матиме меншу кількість "заморожених" кадрів, але повільнішу швидкість завантаження екрану.

Отже, Android мав типову для свого часу концепцію побудови інтерфейсу, що давала достатню продуктивність, але з розвитком смартфонів та користувацьких інтерфейсів стала надто громіздкою та надлишковою для розробки. Цю проблему інженери Google вирішили гнучким, модульним та продуктивним інструментом.

Список використаних джерел

1. GoogleI/O Compose presentation [Електронний ресурс] – Режим доступу до ресурсу: https://www.youtube.com/watch?v=VsStyq4Lzxo.

2. Comparing Jetpack Compose performance with XML. [Електронний ресурс] – Режим доступу до ресурсу: https://medium.com/okcredit/comparing-jetpack-compose-performance-with-xml-9462a1282