УДК 004.438 DOI: 10.26565/2519-2310-2019-2-03

ДОСЛІДЖЕННЯ КРОСПЛАТФОРМНОГО ФРЕЙМВОРКУ FLUTTER. ЧИ ОЗНАЧАТИМЕ РОЗКВІТ ЦІЄЇ ТЕХНОЛОГІЇ ЗНИКНЕННЯ НАТИВНОЇ РОЗРОБКИ НА ANDROID ТА iOS?

Олександр Синельніков

Харківський національний університет імені В.Н. Каразіна, майдан Свободи 4, Харків, 61022, Україна
pride.sin.al@gmail.com,

Рецензент: Олександр Потій, д.т.н., проф., Харківський національний університет імені В.Н. Каразіна, майдан Свободи 6, Харків, 61022, Україна
potav@ua.fm

Поступила: жовтень 2019

Анотація. *Робота присвячена дослідженню кросплатформного фреймворку для розробки мобільних додатків Flutter від компанії Google. Проведено практичне випробування технології з ціллю встановити, чи вона достатньо зручна та надійна. У статті представлені висновки за результатами проведених досліджень. Наведено порівняння механізмів роботи Flutter та інших популярних мобільних кросплатформних фреймворків. Розглянуто основні особливості і відмінності технології. Надано рекомендації щодо її використання.*

Ключевые слова: *кросплатформні рішення; мобільні додатки; нативна розробка; мова програмування Dart.*

1 Введение

Станом на сьогодні статистика використання мобільних додатків (МД) говорить про те, що ринок зростає з геометричною прогресією, а особливо стрімка тенденція простежується саме серед кросплатформних програмних рішень, які набувають все більшої популярності. Мова йде про програмне забезпечення, яке може запускатися на різних платформах (*Android та iOS*), маючи при цьому одну кодову базу. Перевага таких МД полягає у зниженні витрат та прискоренні процесу їх розробки. Це робить крос платформні (К-П) програмні рішення дуже привабливими, особливо, у корпоративному секторі інформаційних технологій. Та чому ж попри всі переваги, нативна розробка (тобто розробка під одну конкретну платформу) МД і досі все ще затребувана? Справа в тому, що програмні рішення, на основі яких реалізовано К-П розробку, мають певній перелік недоліків, через які такі додатки поступаються нативним за тими чи іншими показниками (*наприклад, швидкість візуалізації елементів на екрані мобільного пристрою, продуктивність та інші*). У даної роботі представлений огляд К-П фреймворку Flutter, проведено його порівняння з іншими подібними технологіями, а також надано резюме досліду стосовно особливостей його практичного використання.

2 Принцип роботи Flutter та порівняння його з іншими рішеннями.

Flutter – це SDK (Software Development Kit) з відкритою кодовою базою для створення мобільних додатків від компанії Google. Він використовується для розробки додатків під Android та iOS, а також це основний спосіб розробки додатків під Google Fuchsia – оперативної системи від Google, яка зараз (станом на листопад 2019 року) знаходиться в стадії активної розробки; прогнозується, що вона дебютує через два роки і в кінцевому підсумку замінить Android, Wear OS і Chrome OS. Творці фреймворку стверджують, що він позбавлений більшості недоліків своїх попередників, а саме таких К-П фреймворків, як React Native, PhoneGap, Ionic, Xamarin та інших.

Ні для кого не секрет, що у сфері розробки К-П мобільних додатків є достатнє багатство вибору, що у різних технологій є свої переваги і недоліки. Що ж такого революційного може запропонувати нам Flutter? Щоб це наочно виявити, необхідно порівняти деталі архітектури Flutter та React Native. Справа в тому, що React використовує нативні віджети для відтворення UI і комунікацією з платформними компонентами. Він використовує для цього, так зва-

ний міст (див. Рис. 1), який і є тим «вузьким місцем», що значно уповільнює малювання (*візуалізації*) UI.

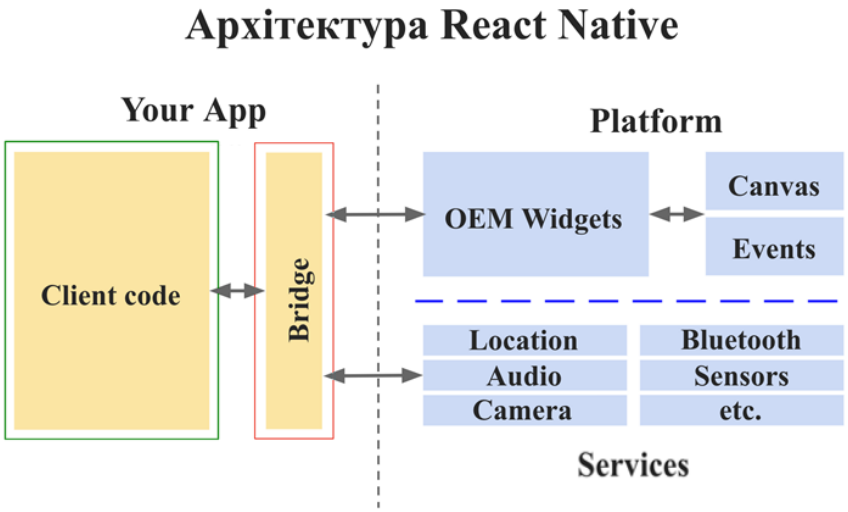

Рис. 1 – Схема архітектури React Native

В архітектурі Flutter (*флаттер*) такий міст відсутній. І хоча інтерфейс між Dart і платформним кодом все ще існує (Рис. 2), йому для роботи потрібно значно менше часу.

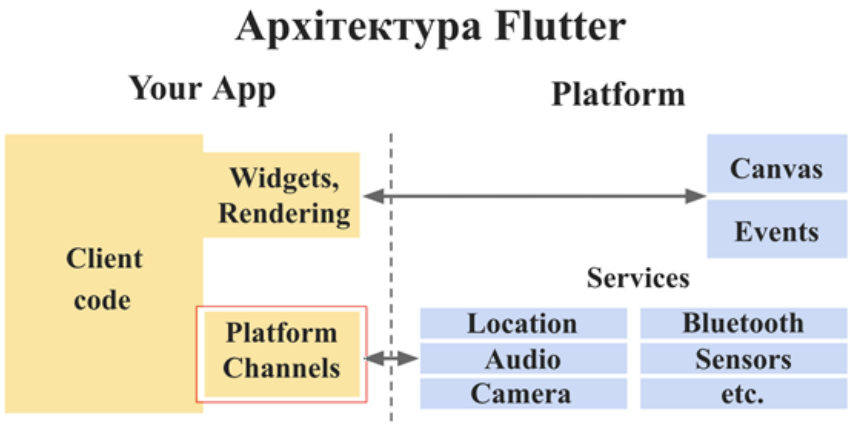

Рис. 2 – Схема архітектури Flutter

Варто звернути увагу на малювання елементів. У кросплатформних рішеннях, які використовують нативні компоненти UI, для взаємодії з ними використовується віртуальне дерево віджетів (Рис. 3).

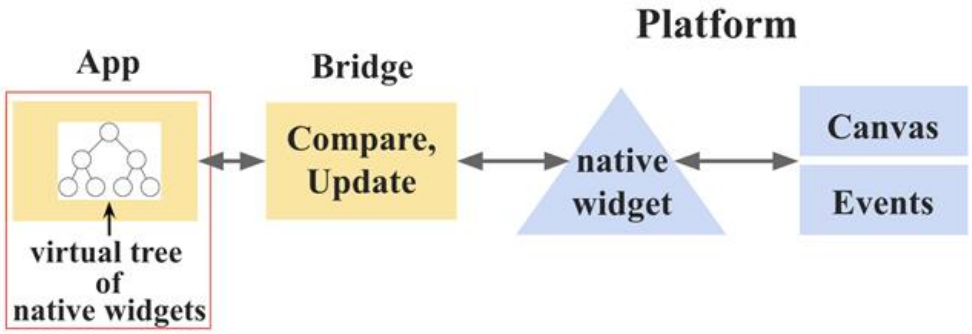

Рис. 3 – Схема малювання елементів у React Native

Віртуальне дерево – це абстрактна версія зображення з екрану, створена з використанням об'єктів, тобто програмне уявлення того, що бачить користувач. Віртуальне дерево є незмінним і перебудовується з нуля кожен раз, коли що-небудь треба поміняти на екрані. Віртуальне дерево порівнюється з положенням і станом нативних віджетів для генерації набору мінімальних змін, які потім виконуються, щоб оновити стан. Нарешті, платформа повторно відображає реальне дерево і малює його на екрані. Все це необхідно лише тому, що оновлення всього екрану відразу – занадто дорога в сенсі продуктивності операція.

Однак Flutter не використовує нативні елементи екрану або навіть WebViews, він надає свої власні віджети. Це збільшує їх гнучкість та швидкість рендеринга. Flutter малює (відображує) свої віджети прямо на canvas. Таким чином, те, що було віртуальним деревом віджетів в попередньому прикладі, тепер є безпосередньо деревом віджетів (Рис. 4), яке відображується на екрані.

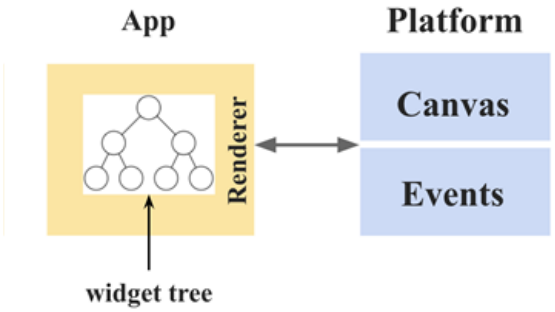

Рис. 4 – Схема відображення елементів у Flutter

Рендерер Flutter перемальовує тільки ті віджети, які необхідно оновити на екрані. Незмінені віджети, навіть ті, які були просто переміщені, частково відновлюються з кешу (дуже швидко).

3 Мова програмування Dart

Dart – це мова програмування з повністю відкритим вихідним кодом, саме на ній працює Flutter. «Всередині» Google ця мова розвивається дуже швидко, вона використовується Adwords, Flutter, Fuchsia та іншими. Нижче представлений невеликий перелік деяких принципових особливостей Dart, які значно спрощують роботу з Flutter:

- Dart може бути AOT (Ahead Of Time) компільованим в швидкий, передбачуваний, машинний ARM код. Це робить Flutter дуже швидким в плані роботи (для збірки release).
- Dart може бути JIT (Just In Time) компільованим для виключно швидких циклів розробки (*включаючи популярну Hot reload Flutter, про яку мова піде трохи пізніше*).
- Код в Дарт є однопоточним, що позбавляє розробників від певних проблем синхронізації даних. При цьому, якщо все ж виникає необхідність у багатопоточності, то є відповідні ізоляти (*isolates*), які виконують роль потоків, і працюють таким чином, що у них немає області спільної пам'яті, а спілкування між ними відбувається через відправку повідомлень по спеціальним каналам.
- Схема «збору сміття» і розподілу пам'яті в Dart особливо швидка для розподілу пам'яті для безлічі недовговічних об'єктів (*ідеально підходить для Flutter, який знищує та повторно перебудовує незмінне дерево віджетів для кожного кадру*).
- Хоча програмістів, які знають цю мову, поки ще, не так вже й багато, ті, хто знають мови Java, JavaScript, Kotlin, C# або Swift, вже можуть почати програмування на Dart практично відразу.

4 Особливості «Hot reload»

Варто звернути особливу увагу на одну з особливостей Flutter – Hot reload. Вона дозволяє розробникам вносити зміни в додаток в той час, коли він уже запущений, при цьому перезавантажиться тільки змінений код, а роботу з додатком можна продовжити з того ж місця, де

розробник зупинився до зміни коду. Весь описаний процес займає менше секунди, а все це можливо завдяки згаданому вище JIT-компілятору. Завдяки цьому функціоналу зникає необхідність в попередньому перегляді зовнішнього вигляду екрану, як це було, наприклад, з xml файлами в Java/Kotlin проектах. В цьому разі можна буквально створювати свій додаток прямо на екрані пристрою без необхідності перезапускати сам додаток.

5 Власний досвід розробки

Найпростіший спосіб знайти сильні та слабкі сторони будь-якого інструменту – спробувати щось зробити з його допомогою. Так, щоб з'ясувати чи є Flutter достатньо стабільний та зручний, та чи можна за його допомогою комфортно створювати додатки, було вирішено створити відповідний додаток для цілей бронювання кімнат для ділових зустрічей.

Творці флаттер подбали про міграцію розробників з інших платформ і пыдготовили велику кількість статей та відповідної документації. На офіційному сайті [1] можна знайти багато корисної інформації по розробці, а також матеріали щодо переходу з Android/iOS/Web розробки на флаттер. Так, первинне ознайомлення займає від пари днів до тижня. В результаті вже можна вільно орієнтуватися в простих прикладах коду фреймворку і розуміти Dart 2 код.

Одним з першочергових завдань був пошук сторонніх компонентів. Розробники Flutter подбали про зручність підключення бібліотек (у Flutter - плагінів). Список можливих залежностей можна знайти на спеціалізованому сайті [2], на якому викладені як офіційні плагіни, так і розроблені сторонніми розробниками (*в даному випадку можна провести аналогію з maven репозиторієм*). Плагіни можуть мати нативний код платформи, який дозволяє їм звертатися до компонентів системи. Такі плагіни під Android, створюються на Java або Kotlin, а під iOS – на Objective-C або Swift. Їх принцип роботи заснований на використанні каналів, де канали – це механізм зв'язку між Dart кодом та кодом конкретної платформи хост-додатку. Однак ці канали можуть передавати тільки бінарну і текстову інформацію. Таким чином, коли деякої нативної функціональності у флаттер немає, то розробник завжди має змогу надати її самостійно.

Враховуючи все вище зазначене, та спираючись на перші особисті враження від розробки тестового додатку, можна зробити наступні висновки стосовно плагінів:

1. Для стабільної роботи додатків необхідно ретельно підходити до вибору плагінів. В першу чергу варто дивитися на офіційні плагіни від розробників флаттер. Їх список є на github [3] і вони знаходяться прямо в репозиторії Flutter.
2. Якщо в запропонованому списку немає потрібного плагіна, то слід переміститися на сайт з плагінами від сторонніх розробників [2], на якому є безліч відповідних бібліотек. Але тут потрібно бути «обережним». Для стабільної роботи додатка варто вибирати тільки ті, для яких є не тільки приклади від розробників, а також і згадки на сторонніх ресурсах (*наприклад, питання з відповідями на StackOverflow*) [4].
3. Велика частина бібліотек, робоча. Однак, багато плагінів ще не мають стабільної версії. Будь-який з них може мати застарілу документацію або приклад використання. Також, хорошою порадою є перевірка працездатності плагіна на обох платформах.

6 Чи потрібен Flutter проекту?

Як зрозуміти що певний проект потрібно або, взагалі, можна стартувати на Flutter? В цілому, цей фреймворк варто використовувати коли завчасно відомий загальний обсяг роботи та відомо, що її можна реалізувати за допомогою стандартних засобів Flutter та плагінів. При цьому не будь-який додаток варто починати писати на Flutter. Так, прикладами поганих додатків на флатер слід вважати:

- додаток, який інтенсивно працює з нативними бібліотеками. На жаль, для того щоб викликати С-код, який нерідко необхідний в Android розробці, потрібно використовувати 2 моста. (Канали та JNI);

- якщо принципово важливе питання ефективного використання ресурсів. Наприклад, простий таймер буде використовувати більше пам'яті або потужності центрального процесору ніж нативний додаток;
- коли в додатку багато особливостей для яких доведеться створювати платформенний код. Наявність каналів робить можливим реалізувати на Flutter все, що можна зробити в нативному додатку, але писати одночасно 2 реалізації може виявитися занадто дорогим у сенсі накладних витрат. В цьому разі мова йде про такі особливості, як контент провайдер, CallKit і т.і.

У свою чергу, позитивними прикладами додатків на Flutter є:

- додаток, який має дизайн, що не залежить від платформи;
- додатки з складним інтерфейсом користувача. В цьому разі Flutter спрощує налаштування UI елементів;
- коли потрібна висока швидкість розробки. Адже необхідна лише одна команда та один клієнт-додаток.

5 Висновки

За результатами аналізу інформації профільних Інтернет ресурсів, узагальнення думки багатьох фахівців та особистого досвіду можна стверджувати, що Flutter є потенційно перспективною технологією, яка швидко набирає популярність. Кілька сміливих концепцій, що лежать в основі роботи даного фреймворку, привносять в розробку нові ідеї та можливості, завдяки яким створювати нові додатки стає значно простіше. Наприклад, таке поширене завдання, як вибір потрібного зображення зі сховища на мобільному пристрої у Flutter робиться одним рядком коду.

Flutter певно не зможе повністю замінити нативну розробку, але він точно задасть новий тренд, в якому в найближчі роки буде рухатися вагома частина індустрії мобільної розробки.

Стосовно перспектив розвитку фреймворку, команда Flutter оголосила про хід активної розробки Hummingbird – Flutter для Web – та розповіла, як це, приблизно, буде працювати.

В планах команди Flutter змінити лише самий низькорівневий шар таким чином, щоб код запускався в браузері. Як очікується, таким чином збережеться основна особливість Flutter – переносимість єдиної кодової бази на різні платформи, в даному випадку – на Web.

Більш детально результати даної роботи, було розглянуто в межах роботи конференції NixMulticonf, відео з якої можна знайти в переліку посилань [5].

Посилання

[1] Flutter dev. URL: https://flutter.dev/ (дата звернення 05.10.2019).
[2] Flutter packages. URL: https://pub.dev/flutter / (дата звернення 05.10.2019).
[3] Github flutter repository. URL: https://github.com/flutter/flutter / (дата звернення 05.10.2019).
[4] Stackoverflow. URL: https://ru.stackoverflow.com / (дата звернення 05.10.2019).
[5] Синельников А. Как приручить Flutter? URL: https://www.youtube.com/watch?v=fiwQy1vLvrM/ (дата звернення 05.10.2019).

Reviewer: Alexandr Potii, Dr. of Sciences (Engineering), Full Prof., V. N. Karazin Kharkiv National University, Kharkiv, 61022, Ukraine. E-mail: potav@ua.fm

Received: October 2019.

Authors:
Oleksandr Synelnikov, computer science student,, V. N. Karazin Kharkiv National University, Kharkiv, Ukraine.
E-mail: pride.sin.al@gmail.com

Investigation of the cross-platform Flutter framework. Will the heyday of this technology mean the disappearance of the native development on Android and iOS?

Abstract. The work is devoted to research of the cross-platform framework for developing mobile applications Flutter from Google. A practical test of the technology was carried out to establish whether it is convenient enough and reliable. The article presents the findings of the testing. The comparison of the mechanisms of Flutter and other popular mobile cross-platform frameworks is given. The main features and differences of the technology are considered. Recommendations on its use are formulated.

Keywords: Cross-Platform Solutions; Mobile Applications; Native Development; Programming Language Dart.

Рецензент: Александр Потий, д.т.н., проф., Харьковский национальный университет имени В. Н. Каразина, Харьков, 61022, Украина..
E-mail: potav@ua.fm

Поступила: Октябрь 2019.

Авторы:
Александр Синельников, студент факультета компьютерных наук, Харьковский национальный университет имени В.Н. Каразина, площадь Свободы, 4, Харьков, 61022, Украина.
E-mail: pride.sin.al@gmail.com

Исследование кросплатформенного фреймворка Flutter. Будет ли означать расцвет этой технологии исчезновение нативной разработки на Android и iOS?

Аннотация. Работа посвящена исследованию кросплатформенного фреймворка для разработки мобильных приложений Flutter от Google. Проведено практическое испытание технологии с целью установить, достаточно ли она удобная и надежная. В статье представлены выводы по результатам проведенных исследований. Приведено сравнение механизмов работы Flutter и других популярных мобильных кроссплатформенных фреймворков. Рассмотрены основные особенности и отличия технологии. Приведены рекомендации по ее использованию.

Ключевые слова: кроссплатформенные решения; мобильные приложения; нативная разработка; язык программирования Dart.