Analiza performansi mobilnih aplikacija rađenih u programskim okvirima Flutter i React Native

Luka Starčević
student, Veleučilište u Šibeniku, luka.starcevic@vus.hr
mr. sc. Ivan Livaja
viši predavač, Veleučilište u Šibeniku, ivan.livaja@vus.hr
Marko Pavelić, mag. ing.
predavač, Veleučilište u Šibeniku, marko.pavelic@vus.hr

Programski okviri Flutter i React Native predstavljaju dva tehnološka rješenja za razvijanje višeplatformnih aplikacija. Programski okvir React Native se zasniva na principu tri dretvi: UI (engl. User Interface) tj. glavna dretva, dretva za JavaScript i Shadow dretva. UI dretva je zadužena za iscrtavanje sučelja sastavljenog od nativnih komponenti, JavaScript dretva je zadužena za svu poslovnu logiku aplikacije, te za slanje podataka o osvježavanju sučelja na UI dretvu, a Shadow dretva služi za računanje položaja i veličine pojedinih elemenata korisničkog sučelja. Arhitektura programskog okvira Flutter zasniva se na tri sloja: Framework, Engine i Emberdder. Zadaća Embeddera je pokretanje svih potrebnih dretvi, pokretanje Flutterovog virtualnog stroja te pružanje platna na kojem će Flutter crtat svoje widgete. Engine služi za rasteriziranje i crtanje komponenti na zaslon, obradu događaja koji se dogode na zaslonu. Framework sloj pruža skup knjižnica koje omogućuju pisanje korisničkog sučelja. U radu je dana kratka usporedba performansi oba rješenja.

Ključne riječi: *Flutter, React Native, Usporedba programskih okvira*

1. UVOD

Rastom popularnosti Interneta sve više i više kompanija se okretalo baš njemu kao glavnom mediju svog poslovanja. Tradicionalne metode prodaje robe i usluga polagano zamjenjuju one na Internetu. Razvitkom mobilne tehnologije otvorila se mogućnost razvitka mobilnih aplikacija koje bi korisnicima bile na dohvat ruke, a koje bi imale manje vrijeme učitavanja u usporedbi sa klasičnim web sjedištima, te imale bolje korisničko iskustvo na mobilnim uređajima od web aplikacija. Tu se pojavio novi problem, a to je, ako bi poduzeća htjela obuhvatiti oba svijeta korisnika mobilnih uređaja, Android i iOS, morali bi angažirati dva različita tima programskih inženjera koji bi morali razviti i održavati dvije različite aplikacije pisane u dva različita programska jezika. To je bio u većini slučajeva prevelik financijski zahvat, pa su se iz tog razloga pojavila dva različita programska rješenja s kojima bi mogli jednostavno riješiti ovaj problem. Prvenstveno *React Native* razvjen od strane Facebook-a 2015. godine, te *Flutter* koji je razvijen od strane Goolge-a. U ovom radu će se sagledati razlike u brzinama izvođenja na mobilnim platformama, analizirati načine na koje ove dvije tehnologije pristupaju rješavanju problema razvijanja aplikacija za više platformi te iznijeti zaključci koja je od njih bolja za korištenje u određenim slučajevima.

2. ARHITEKTURA TEHNOLOGIJA

2.1. React Native

React Native je programski okvir napisan u programskom jeziku JavaScript. On omogućava razvoj *cross-platform* aplikacija koristeći JavaScript. React Native mapira komponente React-a na native komponente platforme. To znači da koristi komponente specifične za pojedinu platformu. Sastoji se od 3 komponente: *Komponente specifične za platformu, JavaScript virtualni stroj i React Native most* (React Native Guide, 2020)

1.1.1. Komponente specifične za platformu

Komponente specifične za platformu predstavljaju API (engl. *Application Programming Interface*) značajke platforme kao što su API za pristup kameri, API za pristup lokalnoj pohrani itd. Te su komponente pisane ili u programskim jezicima Javi ili Objective C-u odnosno Swiftu (React Native Guide, 2020).

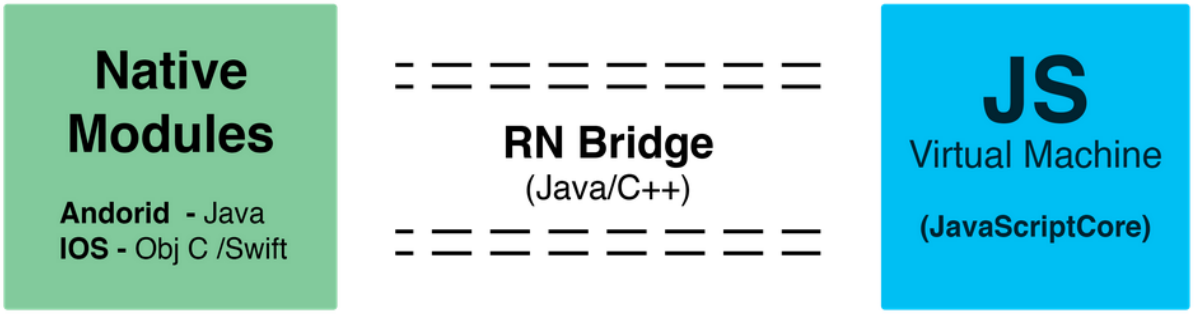
1.1.2. JavaScript virtualni stroj

JavaScript virtualni stroj izvršava sav JavaScript kod. Kao virtualni stroj React Native koristi JavaScriptCore. JavaScriptCore dolazi s iOS platformama ali ne i s Android platformama, te ga zato React Native pakira zajedno s aplikacijom (React Native Guide, 2020).

1.1.3. React Native most

React Native most napisan je u programskim jezicima C++ i Java koji služi za komunikaciju između Nativne dretve i JavaScript dretve. Koristi posebni protokol za prenošenje poruka koristeći JSON objekte (React Native Guide, 2020). Povezivost React mosta prikazana je na slici 1.

Slika 1: React Native most



Izvor: https://www.reactnative.guide/3-react-native-internals/3.1-react-native-internals.html

1.1.4. Raspored dretvi u React Native aplikacijama

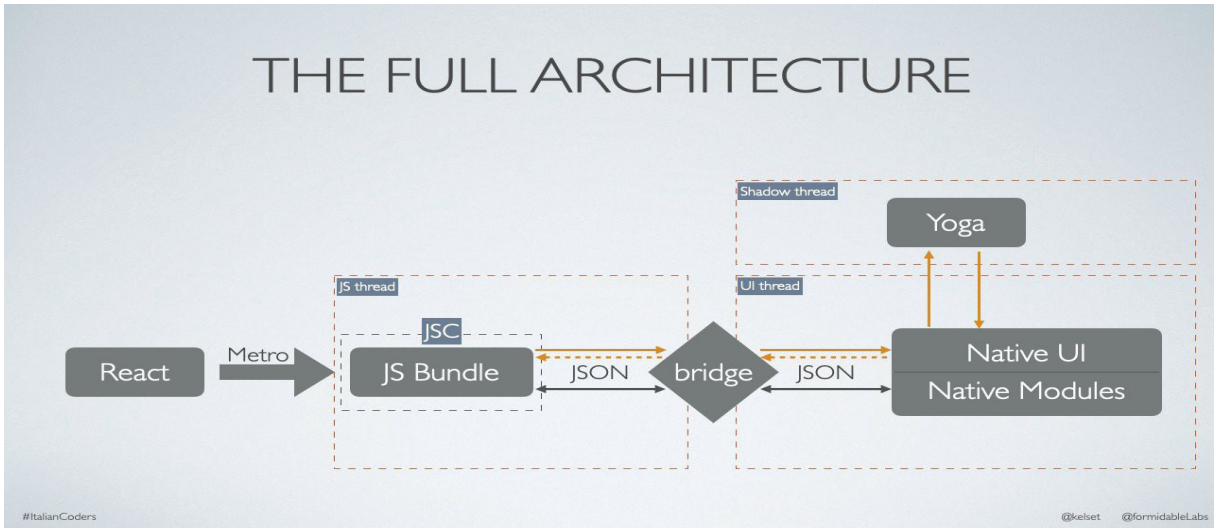
Izvođenje React Native aplikacije počinje od ulazne točke native sustava, kojeg pokreće *glavna dretva* ili *UI dretva* koji zauzvrat pokreće *JavaScript dretvu* koji izvršava Javascrip kod aplikacije. Arhitektura React Nativa aplikacije prikazana je na slici 2. *JavaScript dretva* pakira zajedno poruke o elementima sučelja koji se moraju naslikati te ih šalje na *glavnu dretvu*, ona zauzvrat traži podatke od Shadow dretve o položaju i veličini pojedinih komponenti. Shadow dretva koristi *Yoga layout engine* za izračun podataka o položaju i veličini komponenti na ekranu. Glavna dretva također ima zadaću slušanja korisničkih interakcija s ekranom kao što su dodiri, geste povlačenja po ekranu itd. koje zatim šalje na *JavaScript dretvama* obradu, te zauzvrat

prima upute kako će osvježiti korisničko sučelje (Konicek, 2015; Parashuram, 2018; React Native Guide, 2020; Flutter, 2023).

2.2. Flutter

Programski okvir Flutter je Googleov razvojni alat za razvoj korisničkih sučelja, a njegova arhitektura prikazana je na slici 3. Podržava *ahed-of-time* kompiliranje kao i *just-in-time* kompiliranje. Koristi *Dart* programski jezik koji je razvijen s Flutterom na umu. Za razliku od React Native-a kompilira se direktno u native kod platforme što eliminira potrebu za skupom komunikacijom između dretvi ili za korištenjem WebView-a kao što je slučaj s tehnologijama Ionic i Cord.

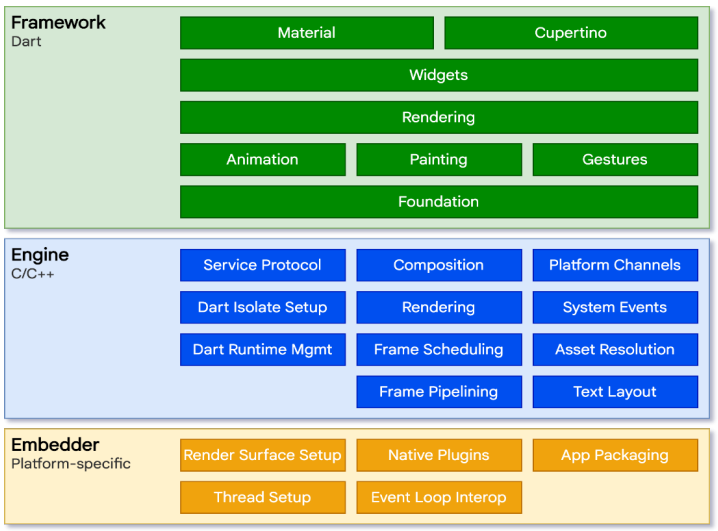
Slika 2: Prikaz arhitekture React Native-a.



Izvor: https://tkssharma.gitbook.io/react-training/react-native/react-native-architecture

Arhitektura mu se sastoji od 3 sloja: programskog okvira (eng. *Framework*), Stroj za renderiranje (eng. *Engine*) te Ugrađivač (eng. *Emberdder*) kao što je prikazano na Slici 3 (Konicek, 2015).

Slika 3: Prikaz Flutter arhitekture



Izvor: https://docs.flutter.dev/resources/architectural-overview#rendering-and-layout

2.2.1. Ugrađivač

Za razliku od React Native-a Flutter ne koristi *native* komponente platforme već stvara svoj skup *widget*-a koje zatim obrađuje i iscrtava na ekranu. Ugrađivač je najniži sloj arhitekture te predstavlja svojevrsno rješenje pokretanja Flutter aplikacija na više platformi.

Zadatak Ugrađivača je:

- Služiti kao ulazna točka na *native* platformama
- Pokretanje dretvi zaduženih za renderiranje widgeta
- Pokretanje Flutter virtualnog stroja na kojem se izvršavaju aplikacije
- Stvaranje podloge na kojem će se iscrtati *widget*-i

Prednost pristupa korištenjem Ugrađivača jest ta da svatko u teoriji može napisati vlastiti Ugrađivač za bilo koju platformu (Konicek, 2015).

2.2.2. Stroj za renderiranje

Stroj za renderiranje je zadužen za rasteriziranje i crtanje grafika na ekran. Pisan je u programskom jeziku C++ te implementira Skia stroj za obradu grafika.

Uz to je zadužen i za (Konicek, 2015):

- Obradu grafika
- Raspored teksta na ekranu
- Mrežni i datotečni I/O
- Postavljanje arhitekture dodataka
- Skup alata za Dart runtime i compile-time

2.2.3. Programski okvir

Prilikom razvoja aplikacije razvojni programer pristupa Flutteru kroz programski okvir (*framework)* koji sadrži skup platformskih, rasporednih i temeljnih knjižnica sastavljenih od serije slojeva (Konicek, 2015):

- *Osnovne temeljne klase* su sastavni dio elemenata kao što su: animacije, slikanje elemenata i geste.
- *Sloj za renderiranje* ima zadaću pružiti sloj apstrakcije kod slaganja elemenata na ekranu, te gradnju stabla objekata koji se mogu renderirati.
- *Sloj widget-a* je apstrakcija u sastavljanju korisničkog sučelja. Svaki objekt u sloju za renderiranje ima odgovarajući objekt u sloju *widget*-a.
- *Material i Cupertino knjižnice* sadrže opsežan skup kontrola koje koriste primitive iz sloja *widget*-a kako bi omogućili stvaranje korisničkog sučelja prema Android ili iOS stilu.

3. ANALIZA PERFORMANSI

Kompanija inVerita je provela analizu performansi Flutter i React Native aplikacija u usporedbi s native aplikacijama na Androidu i iOS-u. *Benchmark* testovi su provedeni na slijedećim uređajima: Xiaomi Redmi Note 5, iPad Mini 3 i Iphone 6s-u. Testovi su provedeni u 3 testna slučaja: Pregled popisa slika, Vektorske animacije i Animacije (inVerita, 2020).

3.1. Pregled popisa slika

inVerita je započela analizu implementacijom istog korisničkog sučelja na 3 navedene platforme. Definirali su fiksno vrijeme za koje su sve platforme trebale doći do dna popisa te su slike spremili u lokalnu predmemoriju. Test je proveden s popisom od 1000 slika.

Vanjske knjižnice koje su korištene za učitavanje i spremanje slika u predmemoriju su:

- iOS-Nuke
- Android-Glide
- React Native React-native-fast-images

3.1.1. Android rezultati

Prema rezultatima testiranja, vidljivima na slici 4, *native* platforma se pokazala najboljom u svim aspektima, slijedi ju Flutter pa na posljetku React Native koji ima najlošije performanse. Povećano korištenje procesne snage se može prepisati korištenju RN mosta koji uzima više procesne moći za serijalizaciju i deserijalizaciju JSON objekata (inVerita, 2020).

Slika 4: Prikaz rezultata testa listanja kroz slike, Android

Android	FPS	CPU %	Max Memory Mb	Battery mAh
Native (Android)	60	2.4	58	49.7 mAh
RN	58	11.7	139	79.01 mAh
Flutter	60	5.4	114	65.28 mAh

Izvor: https://inveritasoft.com/blog/flutter-vs-react-native-vs-native-deep-performance-comparison

3.1.2. iOS rezultati

Rezultati izvođenja testa na iOS platformi pokazuju da je Flutter odmah iza native-a što se tiče performansi, kao što je vidljivo na slici 5. Koriste sličnu količinu memorije, dok native ima daleko manje opterećenje procesora, a Flutter zauzvrat ima manje opterećenje na GPU (inVerita, 2020).

Slika 5: Prikaz rezultata testa Listanja kroz popis slika, iOS

iOS iPhone 6s	FPS	CPU %	GPU %	Memory Mb
Native (iOS)	60	12.72	21.24	154
RN	59	113.13	19.56	220
Flutter	60	33.3	10.75	159

Izvor: https://inveritasoft.com/blog/flutter-vs-react-native-vs-native-deep-performance-comparison

3.2. Vektorske animacije

Tokom testa vektorskih animacija inVerita je koristio slijedeće pakete: Lottie za Android, iOS i React Native te Flare za Flutter.

3.2.1. Android rezultati

Možemo primijetiti sa slike 6 da kod vektorskih animacija Flutter zaostaje za React Native-om te Android *native*-om, iako inVerita navodi kako uklanjanjem jedne specifične animacije iz mreže dolazi do povećanja od 40% u performansama Fluttera.

Slika 6: Prikaz rezultata Vektorske animacije na Androidu

Android	FPS	CPU	Memory Mb	Battery mAh
Native (Android)	30	18.9	205	15.97 mAh
RN	29	15.6	280	14.80 mAh
Flutter	9	12.8	266	14.11 mWh

Izvor: https://inveritasoft.com/blog/flutter-vs-react-native-vs-native-deep-performance-comparison

3.2.1. iOS rezultati

Rezultati iOS testa, prikazanih na slici 7, su slični onom provedenom na androidu u kojem vidimo da Flutter s Flareom zaostaje dramatično za React Native-om i iOS *native*-om.Uzrok tog zaostatkaFlare-ova zahtjevnost na resurse platforme.

Slika 7: Prikaz rezultata testa vektorskih animacija na iOS-u

iOS iPhone 6s	FPS	CPU % per core in total	GPU	Memory Mb	Battery mAh
Native (iOS)	25	151	62.9	48	16.00 mAh
RN	23	72.1	65.1	134.9	18.00 mAh
Flutter	8	123	57.71	117	17.00 mAh

Izvor: https://inveritasoft.com/blog/flutter-vs-react-native-vs-native-deep-performance-comparison

4.3. Animacije

Ovaj test inVerita provodi animirajući 200 slika, koristeći animacije skaliranja, okretanja i nestajanja.

4.3.1. Android rezultati

Android *native* u ovom testu, slika 8, pokazuje najbolje performanse. Flutter i u ovom slučaju pokazuje primjetno lošije rezultate, dok je React Native ovaj test završio s najlošijim FPS-om no, opterećenje na procesor je manje nego kod Flutter platforme.

Slika 8: Prikaz rezultata testa Animacije na Androidu

Android	FPS	CPU	Memory Mb
Native (Android)	58	6.53	80
RN	7	8.5	424
Flutter	19	10.28	168

Izvor: https://inveritasoft.com/blog/flutter-vs-react-native-vs-native-deep-performance-comparison

4.3.2. iOS rezultati

Sa slike 9 možemo vidjeti da je iPhone 6s dovoljno snažan da se provede test animacije bez padanja *frame*-ova. Native je opet najbolji, dok je Flutter blaži na procesor i memoriju od React Native-a, ali je zato zahtjevniji na GPU (inVerita, 2020).

Slika 9: Prikaz rezultata testa animacije na iOS-u

iOS iPhone 6s	FPS	CPU % per core in total	GPU %	Memory Mb
Native (iOS)	59	61	48.28	158
RN	59	118.6	19.8	220
Flutter	59	69	81.91	191

Izvor: https://inveritasoft.com/blog/flutter-vs-react-native-vs-native-deep-performance-comparison

3. ZAKLJUČAK

Flutter i React Native predstavljaju jako zanimljive tehnologije s izuzetno svijetlom budućnosti. U ovom radu su opisane arhitekture jednog i drugog programskog okvira, prikazani su rezultati testova performansi čiji rezultati pokazuju da je korištenje *native* programskih okvira najbolji pristup razvoju mobilnih aplikacija ako se traže što bolje performanse. No, u većini slučajeva performansa između Fluttera, React Native-a i *native* platforme je slična te na taj način gotovo neprimjetna korisniku. Flutter se definitivno pokazao boljim rješenjem *cross-platform* problema zato što eliminira potrebu za korištenjem mosta između dviju dretvi, koje je zahtjevno na procesor. React tim priprema novu arhitekturu bez mosta, Fabric koja bi mogla prestići Flutter u aspektu performansi. Još jedna značajna prednost React Native-a nad Flutterom je korištenje JavaScript-a koji je popularniji za korištenje od Dart-a. Kad se sve uzme u obzir React Native ima potencijala da prestigne Flutter u većini aspekata, od performansi do vremena prilagodbe programera na kod.

LITERATURA:

1. *Flutter*. (2023). Flutter architectural overview. Preuzeto sa https://docs.flutter.dev/resources/architectural-overview (pristupljeno 24.01.2023.)
2. *inVerita*. (2020). Flutter vs React Native vs Native: Deep performance comparison. Preuzeto sa https://inveritasoft.com/blog/flutter-vs-react-native-vs-native-deep-performance-comparison (pristupljeno 29.01.2023.)
3. Konicek, M. (2015). *Under the hood of React Native – Reactive 2015*. Video prezentacije: https://www.youtube.com/watch?v=8N4f4h6SThc
4. Parashuram, N. (2018). *React Native's new Architecture – React Conf 2018*. Video prezentacije: https://www.youtube.com/watch?v=UcqRXTriUVI
5. *React Native Guide*. (2020). React Native Internals. https://reactnative.dev/docs/performance (pristupljeno 23.01.2023.)

ABSTRACT

PERFORMANCE ANALYSIS OF MOBILE APPLICATIONS MADE IN FLUTTER AND REACT NATIVE FRAMEWORKS

Flutter and React Native programming frameworks represent two technological solutions for developing cross-platform applications. The React Native programming framework is based on the principle of three elements: UI (User Interface), i.e. main tree, JavaScript tree and Shadow tree. The UI tree is responsible for drawing the interface composed of native components, the JavaScript tree is responsible for all the business logic of the application, and for sending data about refreshing the interface to the UI tree, and the Shadow tree is used for calculating the position and size of individual elements of the user interface. The architecture of the Flutter framework is based on three layers: Framework, Engine and Embedder. Embedder's job is to run all the necessary drivers, start Flutter's virtual machine, and provide a canvas on which Flutter will draw its widgets. Engine is used for rasterizing and drawing components on the screen, processing events that happen on the screen. The Framework layer provides a set of libraries that allow writing user interfaces. The paper provides a brief comparison of the performance of both solutions.

Keywords: ReactNative, Flutter, Framework Comparrison