Razvoj višeplatformskih aplikacija korištenjem iste kodne baze

Mijalić, Marin

Master's thesis / Diplomski rad

2021

Degree Grantor / Ustanova koja je dodijelila akademski / stručni stupanj: **University of Dubrovnik / Sveučilište u Dubrovniku**

Permanent link / Trajna poveznica: https://urn.nsk.hr/urn:nbn:hr:155:506855

Rights / Prava: In copyright / Zaštićeno autorskim pravom.

Download date / Datum preuzimanja: **2024-04-24**

Repository / Repozitorij:

Repository of the University of Dubrovnik

zir.nsk.hr

dabar
DIGITALNI AKADEMSKI ARHIVI I REPOZITORIJI

SVEUČILIŠTE U DUBROVNIKU
ODJEL ZA ELEKTROTEHNIKU I RAČUNARSTVO

Marin Mijalić

RAZVOJ VIŠEPLATFORMSKIH APLIKACIJA KORIŠTENJEM ISTE KODNE BAZE

DIPLOMSKI RAD

Dubrovnik, rujan, 2021.

SVEUČILIŠTE U DUBROVNIKU
ODJEL ZA ELEKTROTEHNIKU I RAČUNARSTVO

RAZVOJ VIŠEPLATFORMSKIH APLIKACIJA KORIŠTENJEM ISTE KODNE BAZE

DIPLOMSKI RAD

Studij: Primijenjeno / poslovno računarstvo
Mentor: doc. dr. sc. Krunoslav Žubrinić
Student: Marin Mijalić

Dubrovnik, rujan, 2021.

SAŽETAK

U ovom diplomskom radu kroz razvoj aplikacije *Domenitos* na web, mobilnoj i desktop platformi opisan je postupak izrade programskog rješenja i razlike između danas najčešće korištenih tehnologija za izradu višeplatformskih aplikacija. *Domenitos* aplikacija namijenjena je za evidenciju čišćenja apartmana. Kroz izrađenu aplikaciju mogu se dodavati, pregledavati i ažurirati podaci o vlasnicima, apartmanima, čistačima i zadacima.

U diplomskom radu opisan je proces modeliranja i izrade rješenja. Opisane su korištene tehnologije koje omogućuju izradu web, mobilnih i desktop aplikacije korištenjem iste baze izvornog koda, poput *Fluttera* i *Darta*. Napravljena je tržišna analiza programskog okvira *Flutter* te su opisane glavne prednosti i nedostaci tog okvira.

Ključne riječi: razvoj aplikacija, razvoj višeplatformskih aplikacija, *Flutter*, *Dart*

ABSTRACT

In this thesis, through the development of the application *Domenitos* on web, mobile and desktop platform process of software development and the differences between the most common technologies used today for the development of cross-platform applications. The *Domenitos* application was developed to keep records of apartment cleaning. The functionality of the application includes viewing, adding and updating data about owners, apartments, cleaners and tasks.

This thesis describes the process of modeling and solution development. The project includes a detailed description of the development technologies used to build web, mobile and desktop applications with the same source code, such as *Flutter* and *Dart*. A market analysis of the *Flutter* software framework was performed and the main advantages and disadvantages of this framework have been described.

Keywords: software development, development of cross-platform applications, *Flutter, Dart*

SADRŽAJ

Sažetak I
Abstract II
1 Uvod 1
1.1 Definicija rada 1
1.2 Svrha i ciljevi rada 1
1.3 Struktura rada 1
2 Razvoj aplikacija za više platformi korištenjem iste baze izvornog koda 2
2.1 Tehnologije za izradu aplikacija na više platformi 2
2.2 Flutter 2
2.3 Dart 4
2.4 FlutterDevTools 4
2.5 Android Studio IDE 4
3 Projektni zahtjevi aplikacije 6
3.1 Aplikacija *Domenitos* 6
3.2 Korisnici proizvoda 6
3.3 Funkcionalni zahtjevi 6
3.3.1 Pregled Zadataka 7
3.3.2 Dodjela zadataka 8
3.3.3 Pregled čistača 9
3.3.4 Dodavanje čistača 10
3.3.5 Pregled apartmana 11
3.3.6 Dodavanje apartmana 12
3.3.7 Izvršenje zadataka 13
3.3.8 Pregled vlasnika 14
3.3.9 Dodavanje vlasnika 15
3.3.10 Promjena jezika 16
3.3.11 Promjena lozinke 17
3.3.12 Ažuriranje apartmana 18
3.3.13 Ažuriranje vlasnika 19
3.3.14 Ažuriranje čistača 20

3.4 Ostali zahtjevi ... 21
4 Pregled komponenata i veza ... 22
4.1 Logički model baze podataka ... 22
4.2 Implementacija sustava ... 24
5 Opis različitih platformi ... 25
5.1 Mobilna aplikacija Domenitos ... 25
5.1.1 Administracijski podsustav ... 26
5.1.2 Podsustav Čistači ... 31
5.1.3 Zajednički ekran postavki ... 32
5.1.4 Analiza koda ... 33
5.2 Web aplikacija *Domenitos* ... 34
5.2.1 Tehnologije web *Fluttera* ... 34
5.2.2 Performanse *rendering engine* ... 35
5.2.3 Korištenje web *Fluttera* ... 36
5.2.4 Implementacija web aplikacije *Domenitos* ... 36
5.2.5 Problemi tijekom izrade web aplikacije ... 37
5.2.6 Osvrt na probleme u razvoju ... 39
5.2.7 Dizajn web aplikacije *Domenitos* ... 40
5.2.8 Distribucija web aplikacije Domenitos ... 42
5.3 Windows aplikacija Domenitos ... 42
5.3.1 Implementacija Windows aplikacije Domenitos ... 42
5.3.2 Kreiranje Windows aplikacije Domenitos ... 43
5.3.3 Distribucija Windows aplikacije Domenitos ... 44
5.3.4 Problemi s podrškom za razvoj Windows aplikacija ... 44
6 Analiza fluttera i konkurencije ... 46
6.1 Mobilne performanse ... 47
6.2 Web performanse ... 49
6.3 Developer utisak i želja za učenjem tehnologija ... 49
6.4 Popularnost ... 50
6.5 Tržišni trendovi ... 50
6.6 Krivulja učenja i produktivnost ... 51
6.7 Vrijeme izgradnje aplikacije i pronalazak programera ... 52

6.8 Primjeri aplikacija 53
6.9 Sažetak analize 53
7 Prednosti i nedostaci Fluttera 55
7.1 Isto korisničko sučelje i poslovna logika 55
7.2 Napravljeni i prilagođeni, widgeti i animacije 56
7.3 Jednostavna implementacija logike specifične za platformu 56
7.4 *Dart* prednosti 57
7.5 Googleovo jamstvo dugoročne podrške 58
7.6 Portabilnost 58
7.7 Internacionalnost i pristupačnost 58
7.8 DevTools 59
7.8.1 *Flutter* inspektor 59
7.8.2 Pregled performansi 60
7.8.3 Pregled radne memorije 62
7.8.4 Ispravljač pogrešaka 62
7.8.5 Pregled mreže 63
7.8.6 Alat za analizu veličine aplikacije 64
7.9 Nedostatci *Darta* 65
7.10 Veličina aplikacija 66
7.11 Manjak biblioteka 66
8 Zaključak 67
9 Prilozi 70
9.1 Popis slika 70
9.2 Popis tablica 72
9.3 Isporuka sustava 72
9.3.1 Izvorni kod programskog rješenja 72
9.3.2 Upute za instalaciju 72

1 UVOD

Svako poduzeće koje posluje u informacijskom dobu treba svoje poslovanje poboljšati i napraviti efikasnijim uz pomoć mobilnih, web i desktop aplikacija. Razvoj nativnih aplikacija za svaku platformu može uzrokovati probleme poduzeću i razvojnom timu. Problemi nastaju zbog rastućeg financijskog troška, razlike u korisničkom iskustvu i samoj brzini razvoja. Višeplatformno razvijanje aplikacija omogućava korištenje istog izvornog koda te s time rješava probleme u razvoju izvornih aplikacija za svaku platformu. Jedno od takvih tehnoloških rješenja je *Flutter* [1].

1.1 Definicija rada

Predmet istraživanja diplomskog rada je analiza razvoja višeplatformskih aplikacija korištenjem istog izvornog koda, *Flutter* okvira i njegovih mogućnosti. Koristeći *Flutter* izgradit će se mobilna, web i desktop aplikacija koja olakšava posao kompaniji za čišćenje.

1.2 Svrha i ciljevi rada

Cilj rada je ustanovljenje lakoće, efikasnosti i mogućnosti izrade mobilne, web i desktop aplikacije korištenjem iste baze izvornog koda u alatu *Fluttera*. Nakon izrade analizirati će se *Flutterovo* mjesto na tržištu te objasniti njegove prednosti i nedostatci.

Svrha rada je utvrđenje i iskorištenje teorijskog i praktičnog znanja stečenoga tijekom studiranja na sveučilištu.

1.3 Struktura rada

Diplomski rad se sastoji od osam poglavlja. Prvo poglavlje je uvodni dio diplomskog rada s opisom svrhe i cilja. Drugo poglavlje ukratko opisuje korištene tehnologije pri izradi mobilne, web i desktop aplikacije korištenjem zajedničke baze izvornog koda. Treće poglavlje opisuje funkcionalne i ostale zahtjeve aplikacije koja je korištena kao primjer u ovom radu. Četvrto poglavlje uz pomoć dijagrama opisuje komponente aplikacije i veze među njima. Peto poglavlje opisuje *Domenitos* aplikaciju uz pomoć opisanih ekrana mobilne aplikacije te opisuje specifičnosti izrade Windows i web verzije aplikacije korištenjem iste baze koda. Šesto poglavlje analizira *Flutter* i njegovu konkurenciju na tržištu. Sedmo poglavlje opisuje prednosti i nedostatke *Fluttera*. Osmo poglavlje donosi zaključak.

2 RAZVOJ APLIKACIJA ZA VIŠE PLATFORMI KORIŠTENJEM ISTE BAZE IZVORNOG KODA

2.1 Tehnologije za izradu aplikacija na više platformi

Razvoj aplikacije na više platformi sastoji se od razvoja jedne aplikacije koja se može izvoditi na različitim operacijskim sustavima umjesto razvijanja različitih verzija aplikacije za svaku traženu platformu. Glavni razlog ovakvog razvoja aplikacije je dobra funkcionalnost na više operacijskih sustava npr. Android i iOS, sa svrhom njegove prodaje ili distribucije većem broju korisnika. Prednosti višeplatformskog razvoja su:

Brži razvoj – Zbog korištenja jednog izvornog koda kod svih traženih platformi vrijeme se znatno smanjuje. Time se smanjuje razvojno vrijeme i vrijeme plasiranja na tržište što je dobro za razvojni i marketinški tim.

Veći doseg korisnika – Kroz razvijanje jedne aplikacije moguće je ciljati na različite platforme i s time maksimizirati doseg aplikacije na veći broj korisnika.

Ujednačenost dizajna – Korisnici lagano prepoznaju elemente korisničkog sučelja i njihove interakcije na različitim platformama, stoga je korisničko iskustvo bolje. Kod izrade aplikacija za svaku pojedinu platformu teže je sinkronizirati dizajn radi različitih osnovnih *widgeta*.

Isplativost – Pošto se razvoj odvija brže i zahtjeva programere koje znaju samo jednu odabranu tehnologiju, potrebni uloženi novac je znatno manji.

Glavni nedostatak višeplatformskog razvoja su performanse. Performansa je jedna od najvažnijih čimbenika aplikacije. Aplikacije razvijene s izvornim tehnologijama će imati bolje performanse, a što su one složenije ili kompliciranije, ta se razlika između performansi povećava u korist izvornih tehnologija.

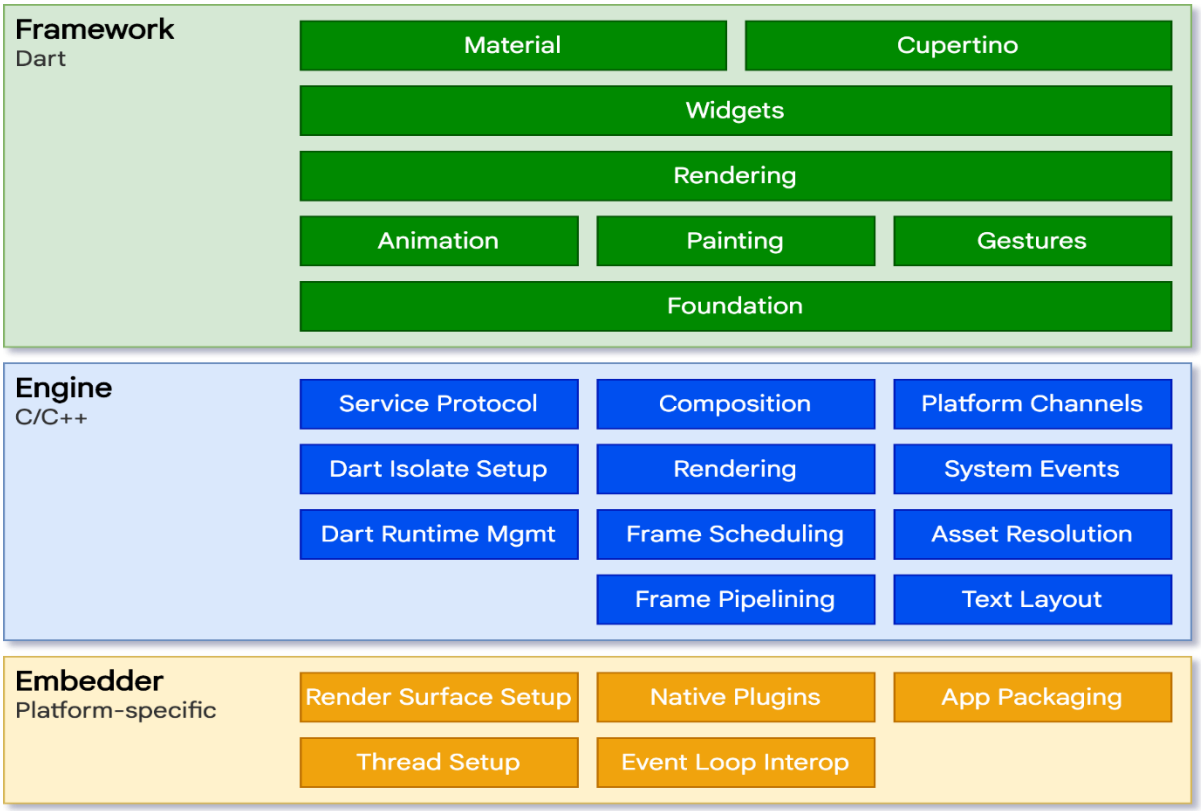
2.2 Flutter

Flutter je Googleov otvoreni alat za korisničko sučelje s kojim je moguće izrada mobilnih (iOS i Android), web i desktop aplikacija iz jedne baze koda. Razlog zbog čega je *Flutter* drugačiji od većine ostalih opcija za izradu mobilnih aplikacija je u tome što ne koristi web tehnologije niti *widgete* već dolazi sa svojim prikazivačkim strojem (engl. *rendering engine*) koji pravi svoje *widgete*. On je napisan u C/C++ kodu. Za implementaciju koristi objektno orijentirani programerski jezik *Dart* [2]. Razvojni software sadrži:

- Optimizirani 2D *rendering engine*,
- Bogat set *widgeta*,
- API za jedinične i integracijske testove,
- Dart DevTools za testiranje, otklanjanje pogrešaka aplikacije,
- Sučelje naredbenih linija za kreiranje, testiranje, izradu i kompiliranje aplikacija.

Pri razvoju je moguće koristiti različite editore za izradu aplikacije poput Android Studija, VS Codea ili IntelliJ IDEA-e i raznorazne dodatke koji su napravljene specifično za njih [3].

Flutter omogućuje specifične funkcije poput *hot-reload* koji omogućuje trenutačno osvježenje cijelog koda bez potrebe ponovnoga izvršenja koda. Moguća je izrada izvršnog koda na više emulatora ili uređaja u isto vrijeme. Na slici 1. vidi se arhitektura *Fluttera*.



Slika 1. Arhitektura *Fluttera* [4]

Programeri vrše interakciju s *Flutterom* kroz okvir *Flutter* koji je napisan u jeziku *Dart*. Uključuje bogat skup biblioteka, izgleda, osnovnih biblioteka i sastoji se od više slojeva. Slojevi su:

Osnovne klase i funkcije poput animacija, slikanja i gesta koje predstavljaju uobičajenu apstrakciju nad temeljima.

Sloj za prikazivanje pruža potpunu kontrolu nad izgledom. Pomoću njega može se izgraditi stablo objekata za ispisivanje kojim se može dinamički manipulirati.

Widgets sloj je apstrakcija kompozicije. Svaki objekt se iscrtava u sloju za iscrtkavanje te ima odgovarajuću klasu u ovom sloju. Omogućuje definiranje kombinacija klasa koje se mogu koristiti više puta.

Material i Cupertino biblioteke omogućuju *Flutteru* korištenje „*Material design*“ ili iOS dizajniranih jezika.

Središte *Fluttera* je *Flutter engine,* koji funkcionira tako da svaki okvir prikazuje na platno kako dolaze. On je u dodiru s okvirom kada se koriste klase, metode i funkcije iz temeljne biblioteke dart:ui.

Ugrađivač (engl. *embedder*) je programski jezik specifičan za platformu na kojoj se izvodi. Za mobilne aplikacije to su Java i C++ za Android i Objective C za iOS i MacOS. Za web aplikacija to su HTML i JS a za Windows i Linux aplikacije to je C++ [5].

2.3 Dart

Dart je klijentsko optimizirani objektno orijentirani programski jezik za izradu aplikacija na različitim platformama. Cilj je ponuditi najproduktivniji programski jezik za izradu aplikacija na više platformi s istom kodnom bazom. Njegov prevoditelj omogućuje pokretanje koda na izvornim platformama (mobilna, desktop) uz pomoć stvaranja strojnog koda a kod web platformi prevođenje u JavaScript. Jezik je *type-safe,* koristi statičke tipove za provjeru da li se varijabla podudara s deklariranim tipom. Vrijednosti ne mogu biti null jedino ako se eksplicitno varijabli deklarira null svojstvo. Dart koristi programske različite biblioteke poput dart:math za funkcije, generiranje brojeva i konstanta, dart:io za HTTP protokol i datoteke i sl. [4].

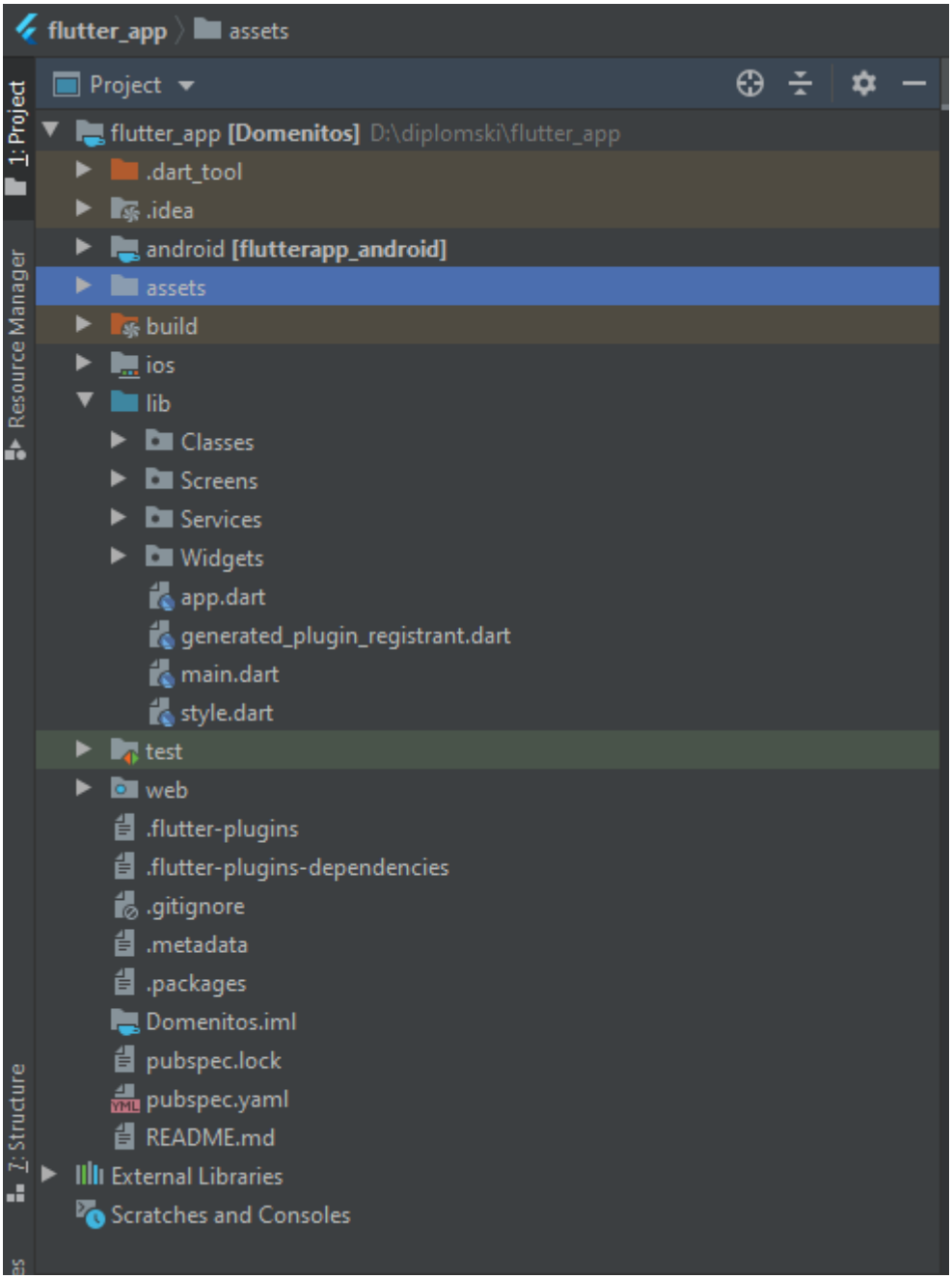
2.4 FlutterDevTools

DevTools [6] je alat koji olakšava uvid u pogreške i njihovo popravljanje, performanse te uvid u detalje korisničkog sučelja. Svaki element se može izabrati i vidjeti detaljno stablo u kojem se jasno vide sva njegova obilježja (boje, širina, dužina, oblik i sl.), što je korisno ako nešto treba popraviti u korisničkom sučelju, jer je na takav način jednostavno locirati i izolirati problem. Više o DevToolsu u poglavlju 7.8.

2.5 Android Studio IDE

Android Studio je službeno razvojno okruženje za izradu Android aplikacija temeljen na IntelliJ IDEA razvojnom okruženju. Razvojno okruženje sadrži uređivač koda, Gradle alat za automatizaciju razvoja, omogućuje pristup emulatorima i različitim programskim alatima poput

alata za testiranje, profiliranje aplikacije i sl. [7]. Na slici 2. vidi se struktura projekta u Android Studiju.



Slika 2. Struktura projekta u Android Studiju

3 PROJEKTNI ZAHTJEVI APLIKACIJE

3.1 Aplikacija *Domenitos*

Aplikacija *Domentios* namijenjena je poduzeću kojemu je potrebna evidencija čistača i njihovih zadataka. Karakteristike i mogućnosti višeplatformskog razvoja će se demonstrirati na njoj. Ona treba biti sigurna, jednostavna za korištenje, pouzdana i efikasna. Glavni zahtjevi koje treba ispuniti su:

- Evidencija čišćenja apartmana,
- Evidencija vlasnika i njihovih apartmana,
- Evidencija čistača i njihovih djelovanja

3.2 Korisnici proizvoda

Administrator – Osoba koja može pristupiti svim dijelovima sustava. Ima mogućnost pregleda, dodavanja čistača, zadataka, vlasnika i apartmana. Administrator također ima uvid u detalje apartmana i njegovu lokaciju preko Google mapa te otkazivanje zadataka.

Čistač – Osoba koja čisti objekte i ima uvid u svoje zadataka i mogućnost njihovog dovršenja, te lokaciju zadataka putem Google mapa.

3.3 Funkcionalni zahtjevi

U tablici 1. pobrojani su svi funkcionalni zahtjevi. Svaki zahtjev označen je prioritetom pri čemu je 1 označava najveći, a 3 najmanji prioritet.

ID zahtjeva	Naziv funkcionalnog zahtjeva	Prioritet
1001	**Pregled Zadataka** – administratori i čistači pregledavaju zadatke.	**1**
1002	**Dodjela Zadataka -** administratori dodjeljuju zadatke čistačima.	**1**
1003	**Pregled Čistača** – administrator ima uvid u čistače.	**1**
1004	**Dodavanje Čistača** – administrator unosi nove čistače.	**1**
1005	**Pregled Apartmana** – administrator ima uvid u apartmane.	**1**
1006	**Dodavanje Apartmana** – administrator unosi nove apartmane.	**1**
1007	**Izvršenje Zadatka** – administrator prekine zadatak ili čistač obavi zadatak.	**1**
1008	**Pregled Vlasnika** – administrator ima uvid u vlasnike.	**2**
1009	**Dodavanje Vlasnika** – administrator ima unosi nove vlasnike.	**2**
1010	**Promjena Jezika** – administrator ili čistač mijenja jezik.	**3**

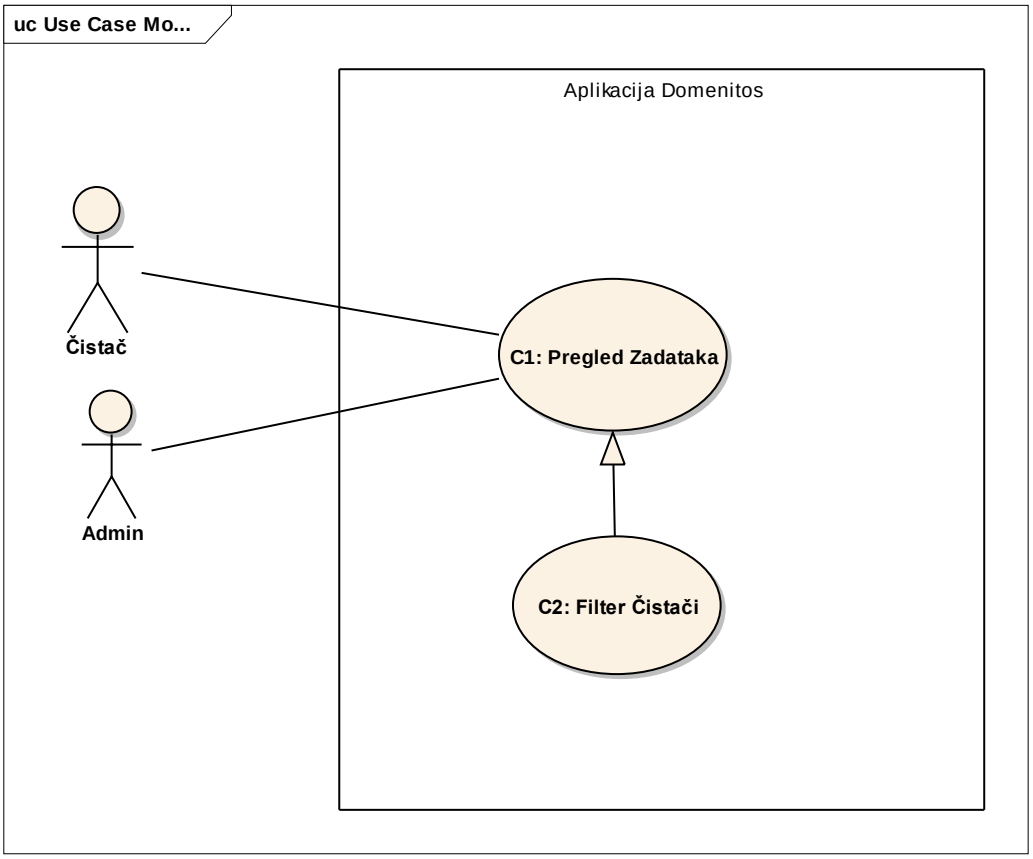
1011	**Promjena Lozinke** – administrator ili čistač mijenja lozinku.	**3**
1012	**Ažuriranje Apartmana** – administrator ažurira apartman	**3**
1013	**Ažuriranje Vlasnika –** administrator ažurira vlasnika	**3**
1014	**Ažuriranje Čistača –** administrator ažurira čistača	**3**

Tablica 1. Funkcionalni zahtjevi

U nastavku ovog potpoglavlja *use case* dijagramima su detaljnije opisani funkcionalni zahtjevi.

3.3.1 Pregled Zadataka

Na slici 3. vidljiv je zahtjev pregled zadataka.



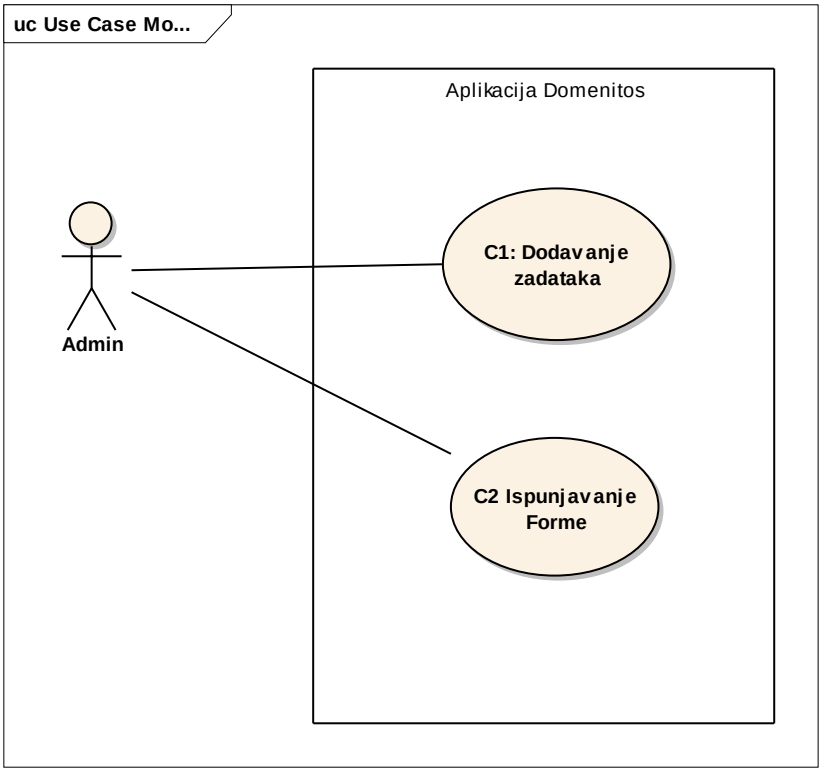
Slika 3. Zahtjev pregled zadataka

ID obrasca	2001	**Oznaka prioriteta**	1	**ID zahtjeva**	1001
Opis oblikovnog obrasca	Služi za pregled zadataka u aplikaciju.				
Glavni sudionik	Administrator, Čistač				
Ostali sudionici	--				
Preduvjeti za početak	--				

Rezultat po završetku	Zadaci ispisani
Glavni scenarij	
i. Administrator ili Čistač otvara aplikaciju. ii. Administrator ili Čistač navigacijom dolazi do ekrana zadaci. iii. Sustav ispisuje zadatak s filterom po čistaču.	
Alternativni scenariji i proširenja	
iii: a) Ako je u sustav ušao čistač nema filtera nego ima u uvidu samo svoje zadatke. iii: b) Ako u sustavu nema aktivnih zadataka ispisuje se filter s čistačima. iii: c) Ako u sustavu nema aktivnih čistača sustav ispisuje poruku o tom stanju.	

3.3.2 Dodjela zadataka

Na slici 4. vidljiv je zahtjev dodavanje zadataka.



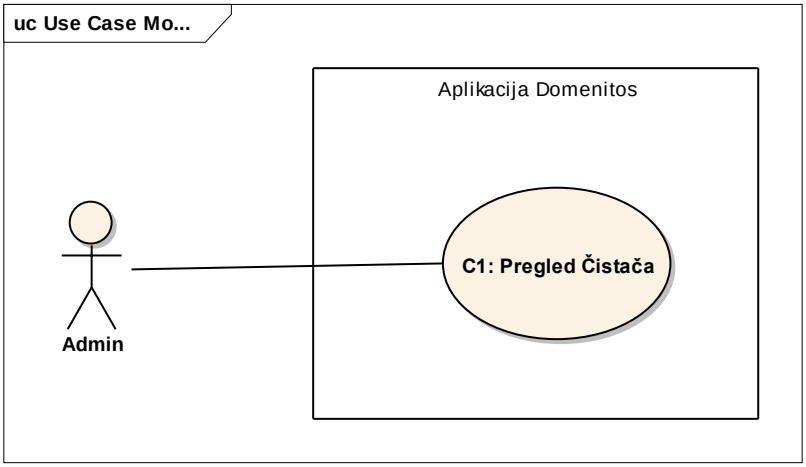
Slika 4. Zahtjev dodavanje zadataka

ID obrasca	2002	**Oznaka prioriteta**	1	**ID zahtjeva**	1002
Opis oblikovnog obrasca	Služi za dodavanje zadataka u aplikaciju.				
Glavni sudionik	Administrator				
Ostali sudionici	--				
Preduvjeti za početak	--				
Rezultat po završetku	Zadatak dodan				
Glavni scenarij					
i. Administrator otvara aplikaciju. ii. Administrator navigacijom dolazi do ekrana dodavanje zadataka.					

iii. Administrator popunjava formu, vidljivu na slici 32. iv. Administrator naređuje spremanje podataka zadataka. v. Sustav dodaje novi zadatak.
Alternativni scenariji i proširenja
v. a) Ako je došlo do pogreške sustav ukazuje gdje je pogreška s odgovarajućom porukom.

3.3.3 Pregled čistača

Na slici 5. vidljiv je zahtjev pregled čistača.

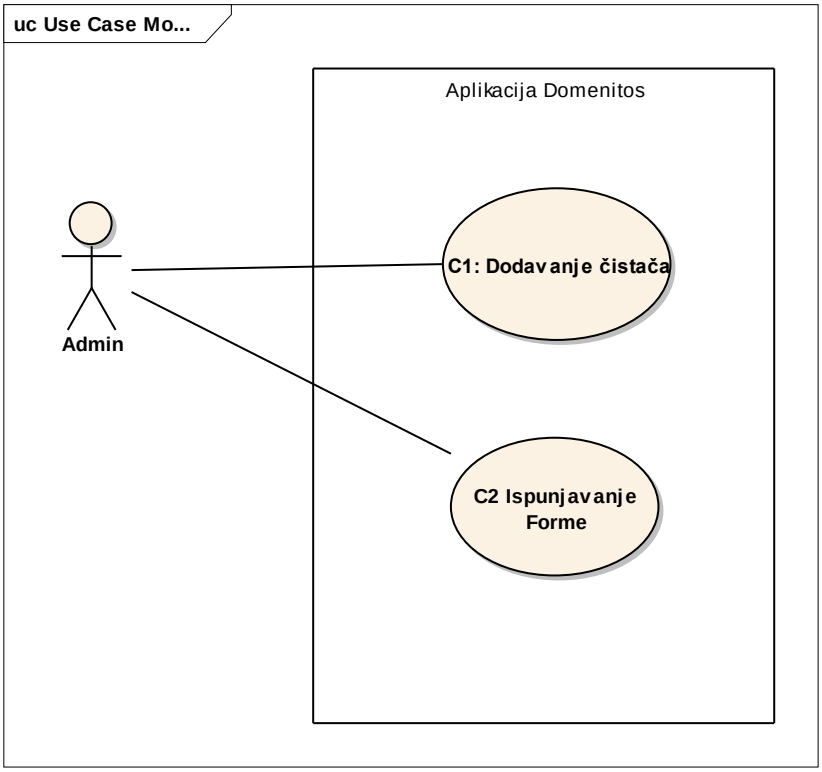


Slika 5. Zahtjev pregled čistača

ID obrasca	2003	**Oznaka prioriteta**	1	**ID zahtjeva**	1003
Opis oblikovnog obrasca	Služi za pregled čistača u aplikaciju.				
Glavni sudionik	Administrator				
Ostali sudionici	--				
Preduvjeti za početak	--				
Rezultat po završetku	Čistači ispisani				
Glavni scenarij					
i. Administrator otvara aplikaciju. ii. Administrator navigacijom dolazi do ekrana čistači. iii. Sustav ispisuje listu čistača.					
Alternativni scenariji i proširenja					
iii. a) Ako u sustavu nema nijedan čistač ispisuje se poruka da nema čistača.					

3.3.4 Dodavanje čistača

Na slici 6. vidljiv je zahtjev pregled čistača.

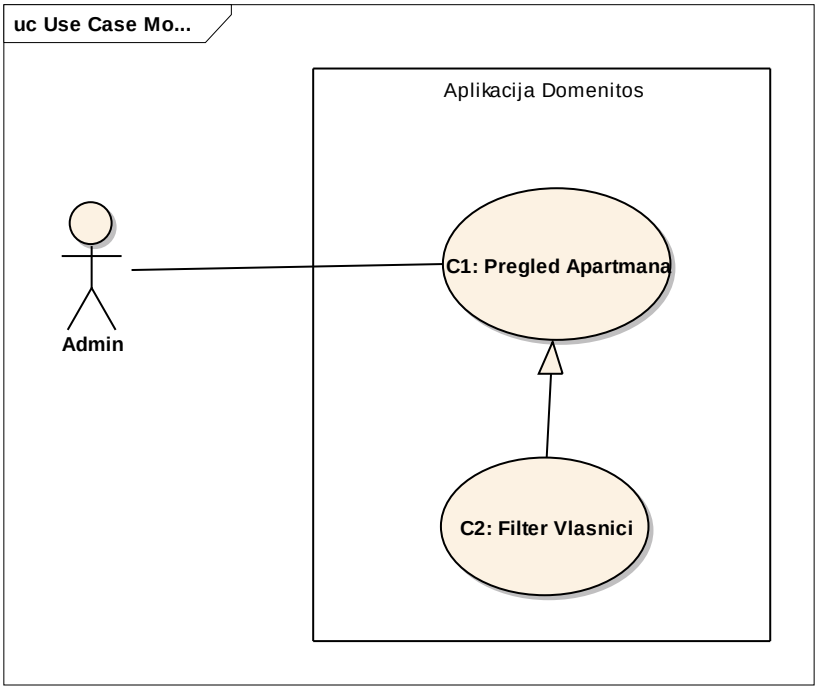


Slika 6. Zahtjev dodavanje čistača

ID obrasca	2004	**Oznaka prioriteta**	1	**ID zahtjeva**	1004
Opis oblikovnog obrasca	Služi za dodavanje čistača u aplikaciju.				
Glavni sudionik	Administrator				
Ostali sudionici	--				
Preduvjeti za početak	--				
Rezultat po završetku	Čistač dodan				
Glavni scenarij					
i. Administrator otvara aplikaciju. ii. Administrator navigacijom dolazi do ekrana dodavanje čistača. iii. Administrator popunjava formu, vidljivu na slici 29. iv. Administrator naređuje spremanje podataka čistača. v. Sustav uspješno dodaje čistača i omogućava mu funkcionalnosti korisnika aplikacije.					
Alternativni scenariji i proširenja					
v. a) Ako je došlo do pogreške sustav ukazuje gdje je pogreška s odgovarajućom porukom.					

3.3.5 Pregled apartmana

Na slici 7. vidljiv je zahtjev pregled apartmana.

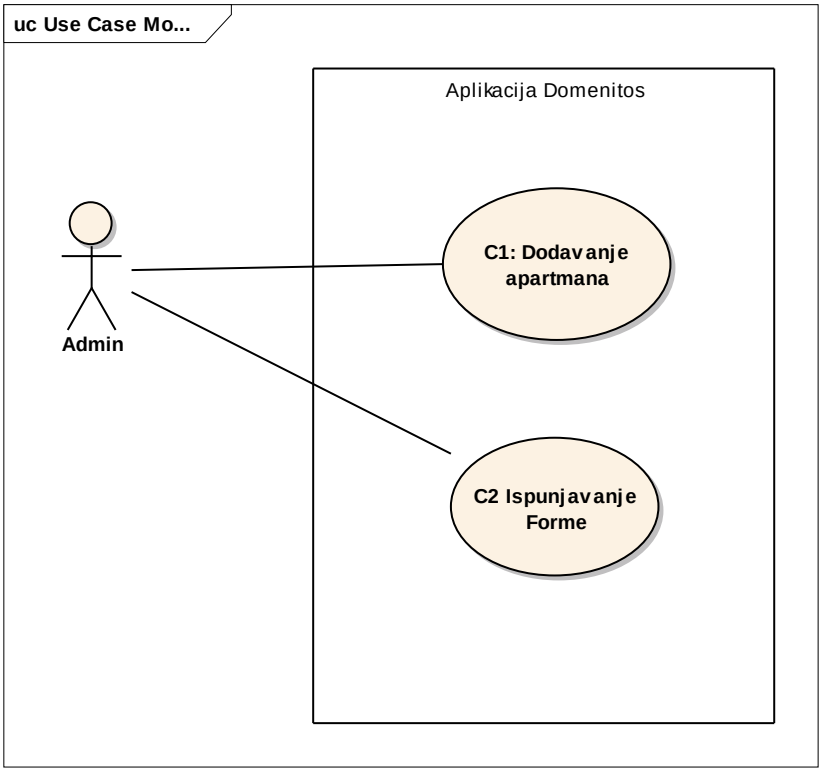


Slika 7. Zahtjev pregled apartmana

ID obrasca	2005	**Oznaka prioriteta**	1	**ID zahtjeva**	1005
Opis oblikovnog obrasca	Služi za pregled apartmana u aplikaciju.				
Glavni sudionik	Administrator				
Ostali sudionici	--				
Preduvjeti za početak	--				
Rezultat po završetku	Apartmani ispisani				
Glavni scenarij					
i. Administrator otvara aplikaciju. ii. Administrator navigacijom dolazi do ekrana apartmani. iii. Sustav ispisuje listu apartmana po filteru vlasnik.					
Alternativni scenariji i proširenja					
iii. a) Ako u sustavu nema niti jednog apartmana sustav ispisuje samo filter po vlasniku. iii. b) Ako u sustavu nema vlasnika sustav ispisuje tu poruku.					

3.3.6 Dodavanje apartmana

Na slici 8. vidljiv je zahtjev dodavanja apartmana.

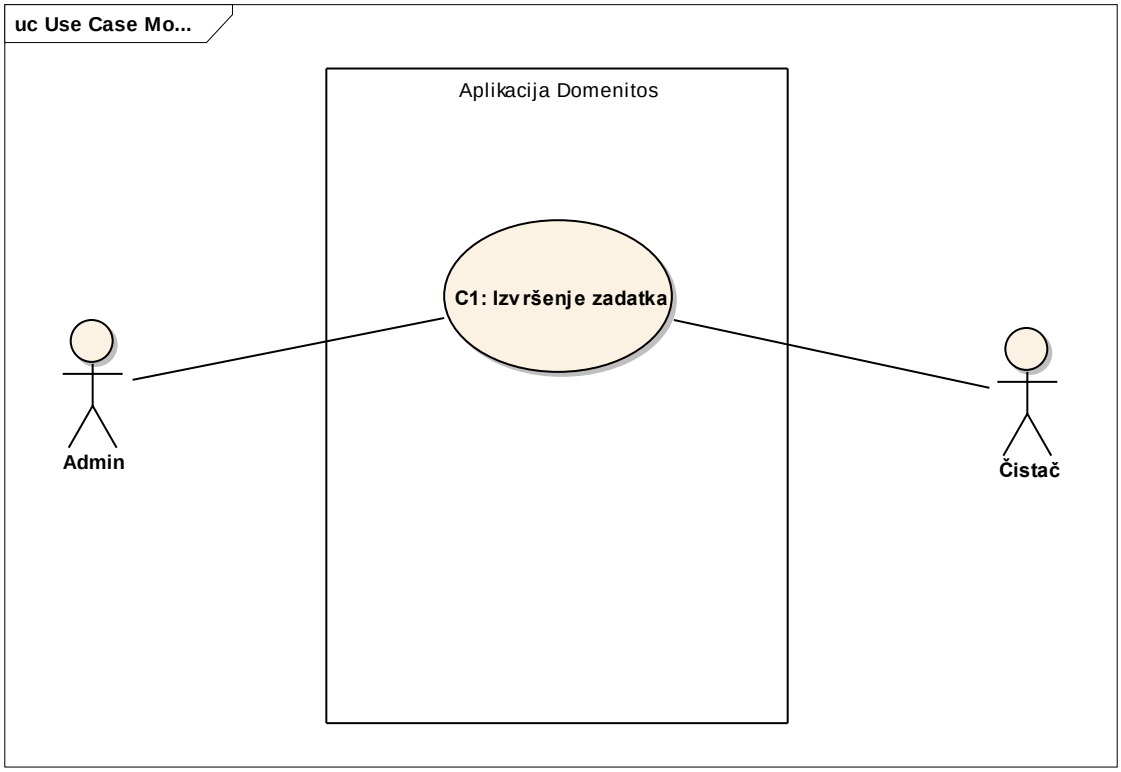


Slika 8. Zahtjev dodavanje apartmana

ID obrasca	2006	**Oznaka prioriteta**	1	**ID zahtjeva**	1006
Opis oblikovnog obrasca	Služi za dodavanje apartmana u aplikaciju.				
Glavni sudionik	Administrator				
Ostali sudionici	--				
Preduvjeti za početak	--				
Rezultat po završetku	Apartman dodan				
Glavni scenarij					
i. Administrator otvara aplikaciju. ii. Administrator navigacijom dolazi do ekrana dodavanje apartmana. iii. Administrator popunjava formu, vidljivu na slici 26. iv. Administrator naređuje spremanje podataka apartmana. v. Sustav dodaje novi apartman.					
Alternativni scenariji i proširenja					
v. a) Ako je došlo do pogreške sustav ukazuje gdje je pogreška s odgovarajućom porukom.					

3.3.7 Izvršenje zadataka

Na slici 9. vidljiv je zahtjev izvršenja zadatka.

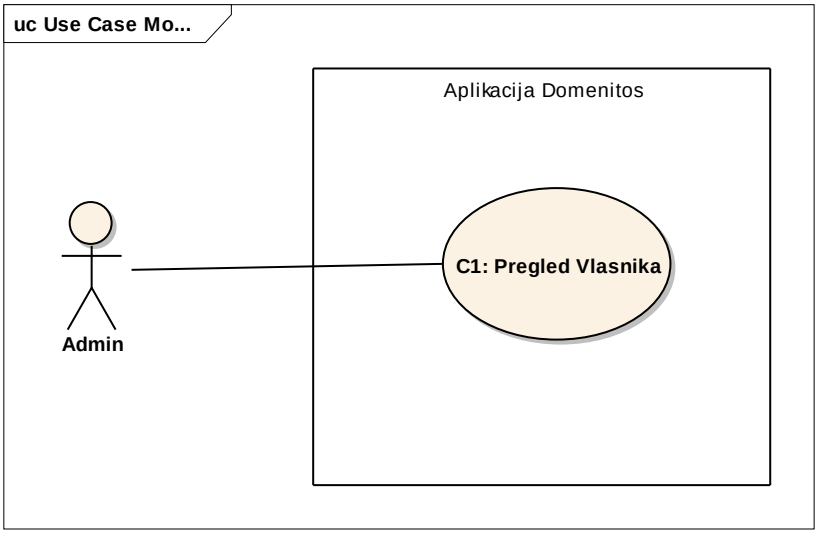


Slika 9. Zahtjev izvršenje zadatka

ID obrasca	2007	**Oznaka prioriteta**	1	**ID zahtjeva**	1007
Opis oblikovnog obrasca	Služi za dodavanje apartmana u aplikaciju.				
Glavni sudionik	Administrator, Čistač				
Ostali sudionici	--				
Preduvjeti za početak	--				
Rezultat po završetku	Zadatak izvršen				
Glavni scenarij					
i. Administrator ili Čistač otvara aplikaciju. ii. Administrator ili Čistač navigacijom dolazi do ekrana zadaci. iii. Sustav ispisuje listu zadataka s filterom po čistaču. iv. Administrator ili Čistač naređuje izvršenje zadataka. v. Sustav izvršava zadatak.					
Alternativni scenariji i proširenja					
iii: a) Ako je u sustav ušao čistač nema filtera nego ima u uvidu samo svoje zadatke.					

3.3.8 Pregled vlasnika

Na slici 10. vidljiv je zahtjev pregleda vlasnika.

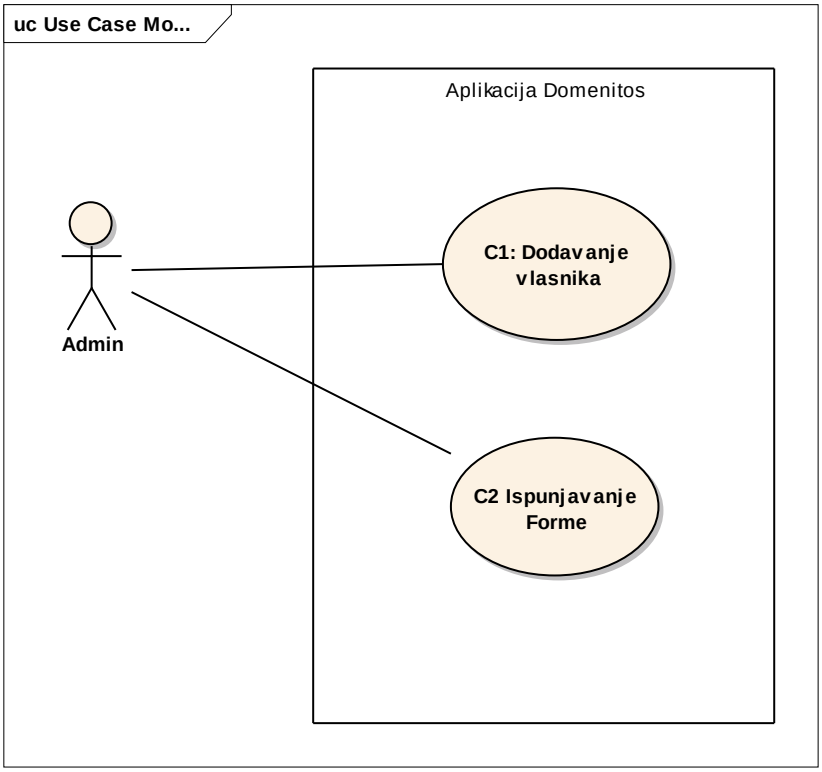


Slika 10. Zahtjev pregled vlasnika

ID obrasca	2008	**Oznaka prioriteta**	2	**ID zahtjeva**	1009
Opis oblikovnog obrasca	Služi za pregled vlasnika u aplikaciju.				
Glavni sudionik	Administrator				
Ostali sudionici	--				
Preduvjeti za početak	--				
Rezultat po završetku	Čistači ispisani				
Glavni scenarij					
i. Administrator otvara aplikaciju. ii. Administrator navigacijom dolazi do ekrana vlasnici. iii. Sustav ispisuje listu vlasnika.					
Alternativni scenariji i proširenja					
iii. a) Ako u sustavu nema nijedan vlasnik ispisuje se poruka da nema vlasnika.					

3.3.9 Dodavanje vlasnika

Na slici 11. vidljiv je zahtjev dodavanja vlasnika.

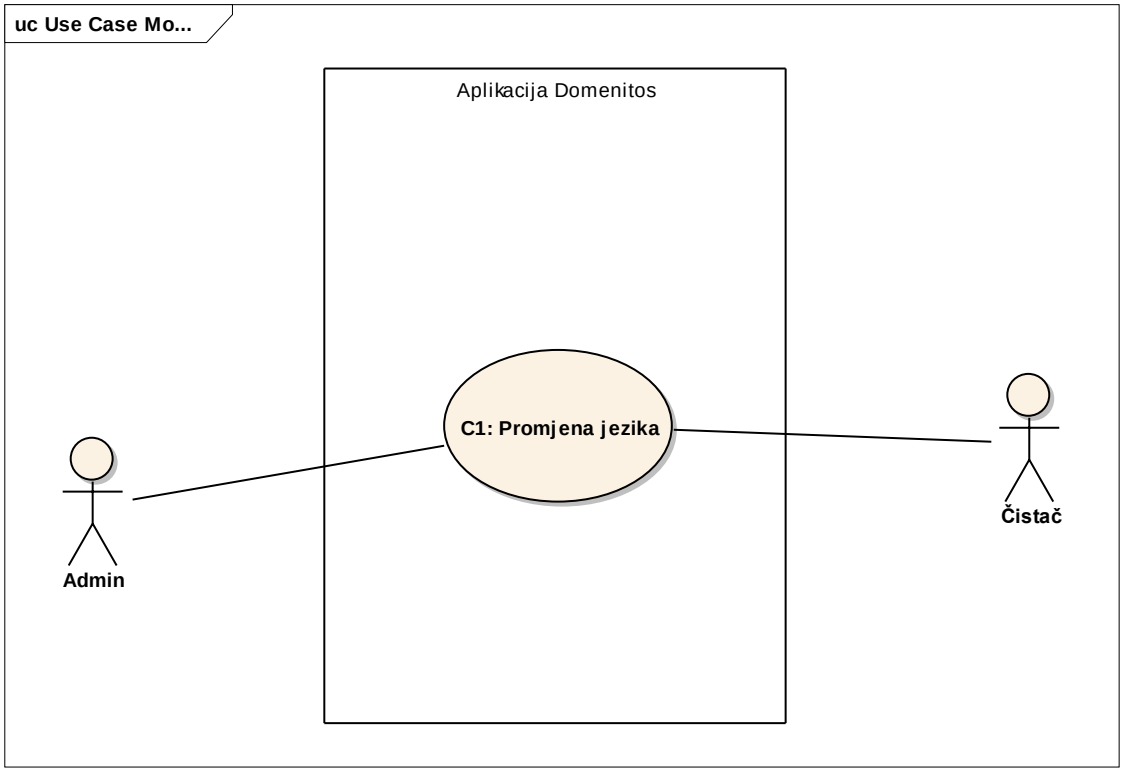


Slika 11. Zahtjev dodavanje vlasnika

ID obrasca	2009	**Oznaka prioriteta**	1	**ID zahtjeva**	1009
Opis oblikovnog obrasca	Služi za dodavanje vlasnika u aplikaciju.				
Glavni sudionik	Administrator				
Ostali sudionici	--				
Preduvjeti za početak	--				
Rezultat po završetku	Vlasnik dodan				
Glavni scenarij					
i. Administrator otvara aplikaciju. ii. Administrator navigacijom dolazi do ekrana dodavanje vlasnika. iii. Administrator popunjava formu, vidljivu na slici 24. iv. Administrator naređuje spremanje podataka vlasnika. v. Sustav dodaje novog vlasnika.					
Alternativni scenariji i proširenja					
v. a) Ako je došlo do pogreške sustav ukazuje gdje je pogreška s odgovarajućom porukom.					

3.3.10 Promjena jezika

Na slici 12. vidljiv je zahtjev promjena jezika.

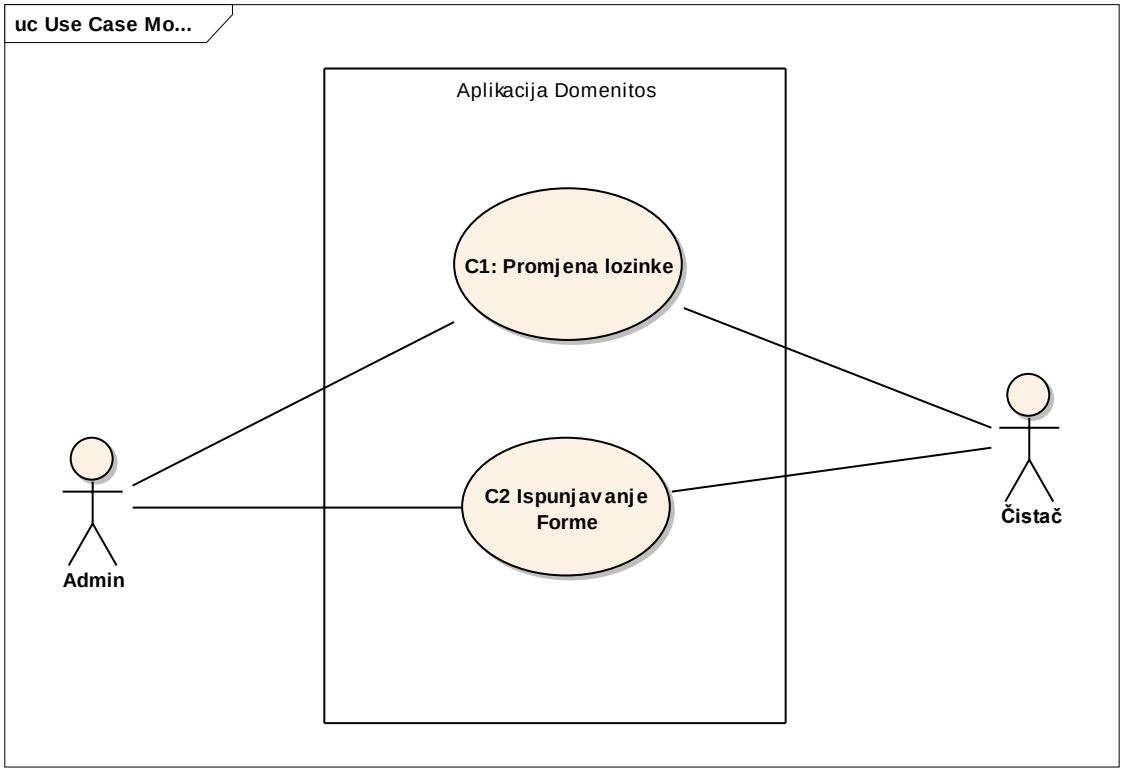


Slika 12. Zahtjev promjena jezika

ID obrasca	2010	**Oznaka prioriteta**	3	**ID zahtjeva**	1010
Opis oblikovnog obrasca	Služi za promjenu jezika.				
Glavni sudionik	Administrator, Čistač				
Ostali sudionici	--				
Preduvjeti za početak	--				
Rezultat po završetku	Lozinka izmijenjena				
Glavni scenarij					
i. Administrator ili Čistač otvara aplikaciju. ii. Administrator ili Čistač navigacijom dolazi do ekrana postavke. iii. Administrator ili Čistač odabire opciju za promjenu jezika. iv. Sustav mijenja jezik aplikacije za administratora ili čistača.					
Alternativni scenariji i proširenja					

3.3.11 Promjena lozinke

Na slici 13. vidljiv je zahtjev promjena lozinke.

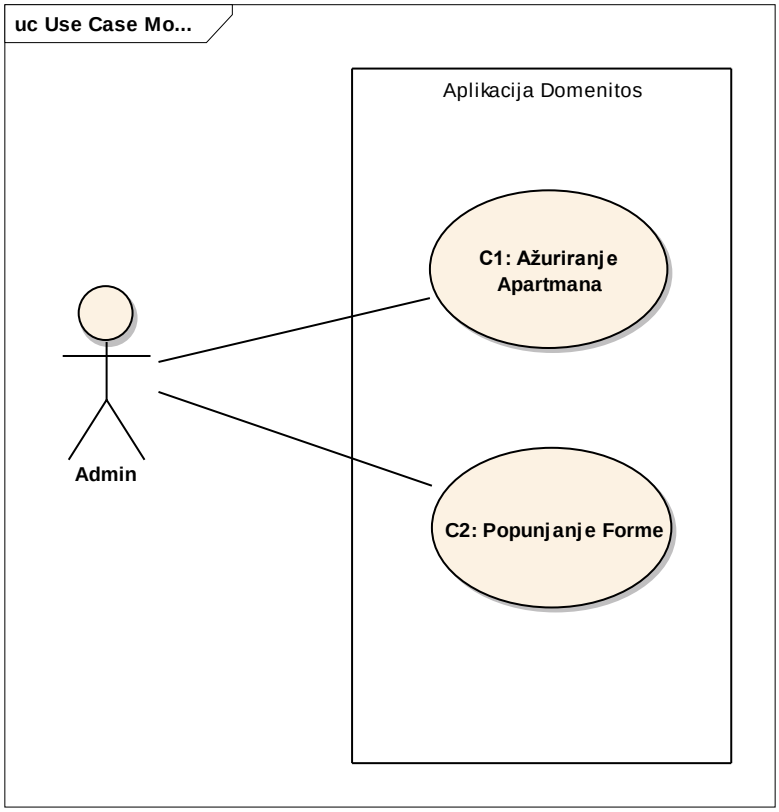


Slika 13. Zahtjev promjena lozinke

ID obrasca	2011	**Oznaka prioriteta**	3	**ID zahtjeva**	1011
Opis oblikovnog obrasca	Služi za promjenu lozinke.				
Glavni sudionik	Administrator, Čistač				
Ostali sudionici	--				
Preduvjeti za početak	--				
Rezultat po završetku	Lozinka izmijenjena				
Glavni scenarij					
i. Administrator ili Čistač otvara aplikaciju. ii. Administrator ili Čistač navigacijom dolazi do ekrana postavke. iii. Administrator ili Čistač odabire opciju za promjenu lozinke. iv. Administrator ili Čistač popunjava formu u kojoj upisuje trenutačnu lozinku i buduću. v. Sustav izmjenjuje lozinku administratora ili čistača.					
Alternativni scenariji i proširenja					
v. a) Ako je došlo do pogreške sustav ukazuje gdje je pogreška s odgovarajućom porukom.					

3.3.12 Ažuriranje apartmana

Na slici 14. vidljiv je zahtjev ažuriranje apartmana.

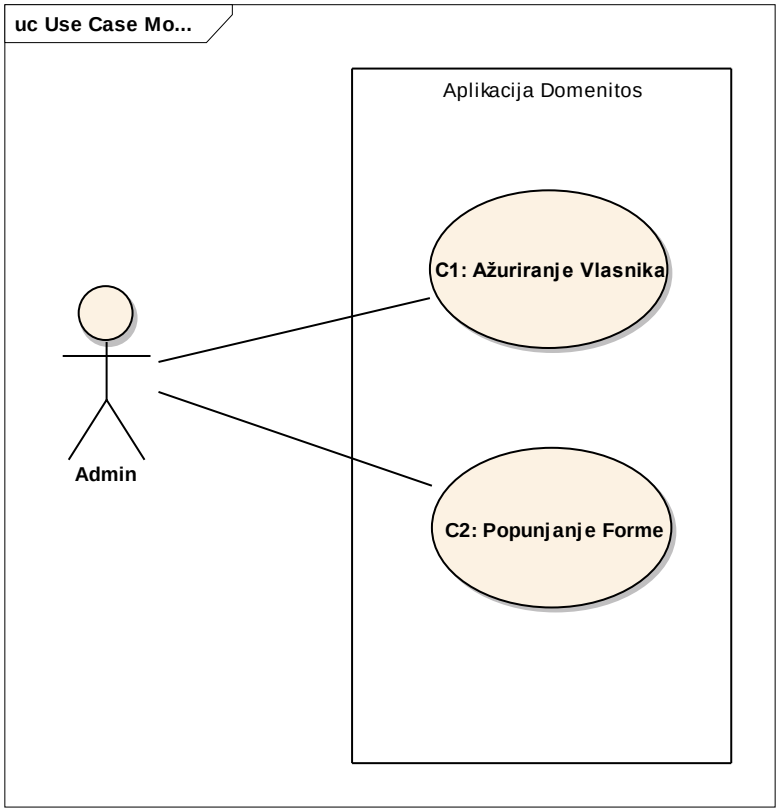


Slika 14. Zahtjev ažuriranje apartmana

ID obrasca	2012	**Oznaka prioriteta**	3	**ID zahtjeva**	1012
Opis oblikovnog obrasca	Služi za ažuriranje apartmana.				
Glavni sudionik	Administrator				
Ostali sudionici	--				
Preduvjeti za početak	--				
Rezultat po završetku	Ažuriranje apartmana				
Glavni scenarij					
i. Administrator otvara aplikaciju. ii. Administrator navigacijom dolazi do ekrana ažuriranje apartmana. iii. Administrator ažurira formu, vidljivu na slici 26. (forma za dodavanje i ažuriranje je ista). iv. Sustav izmjenjuje ažurirane podatke za apartman.					
Alternativni scenariji i proširenja					
iv. a) Ako je došlo do pogreške sustav ukazuje gdje je pogreška s odgovarajućom porukom.					

3.3.13 Ažuriranje vlasnika

Na slici 15. vidljiv je zahtjev ažuriranje vlasnika.

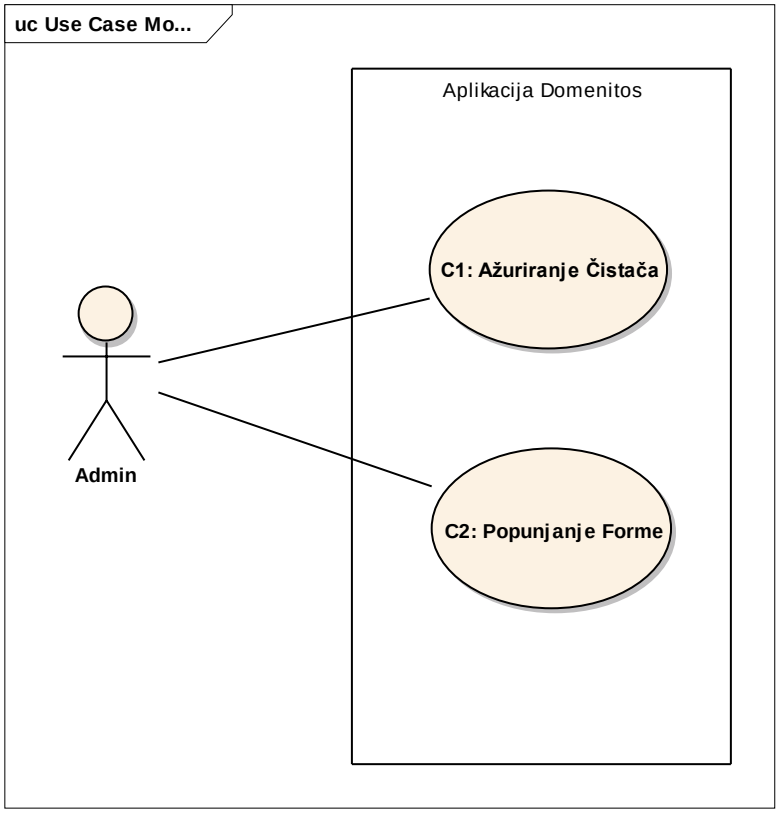


Slika 15. Zahtjev ažuriranje vlasnika

ID obrasca	2013	**Oznaka prioriteta**	3	**ID zahtjeva**	1013
Opis oblikovnog obrasca	Služi za ažuriranje vlasnika.				
Glavni sudionik	Administrator				
Ostali sudionici	--				
Preduvjeti za početak	--				
Rezultat po završetku	Ažuriranje vlasnika				
Glavni scenarij					
i. Administrator otvara aplikaciju. ii. Administrator navigacijom dolazi do ekrana ažuriranje vlasnika. iii. Administrator ažurira formu, vidljivu na slici 24. (forma za dodavanje i ažuriranje je ista). iv. Sustav izmjenjuje ažurirane podatke za vlasnika.					
Alternativni scenariji i proširenja					
iv. a) Ako je došlo do pogreške sustav ukazuje gdje je pogreška s odgovarajućom porukom.					

3.3.14 Ažuriranje čistača

Na slici 16. vidljiv je zahtjev ažuriranje čistača.



Slika 16. Zahtjev ažuriranje čistača

ID obrasca	2014	**Oznaka prioriteta**	3	**ID zahtjeva**	1014
Opis oblikovnog obrasca	Služi za ažuriranje čistača.				
Glavni sudionik	Administrator				
Ostali sudionici	--				
Preduvjeti za početak	--				
Rezultat po završetku	Ažuriranje čistača				
Glavni scenarij					
i. Administrator otvara aplikaciju. ii. Administrator navigacijom dolazi do ekrana ažuriranje čistača. iii. Administrator ažurira formu, vidljivu na slici 30. (forma za dodavanje i ažuriranje je ista). iv. Sustav izmjenjuje ažurirane podatke za čistača.					
Alternativni scenariji i proširenja					
iv. a) Ako je došlo do pogreške sustav ukazuje gdje je pogreška s odgovarajućom porukom.					

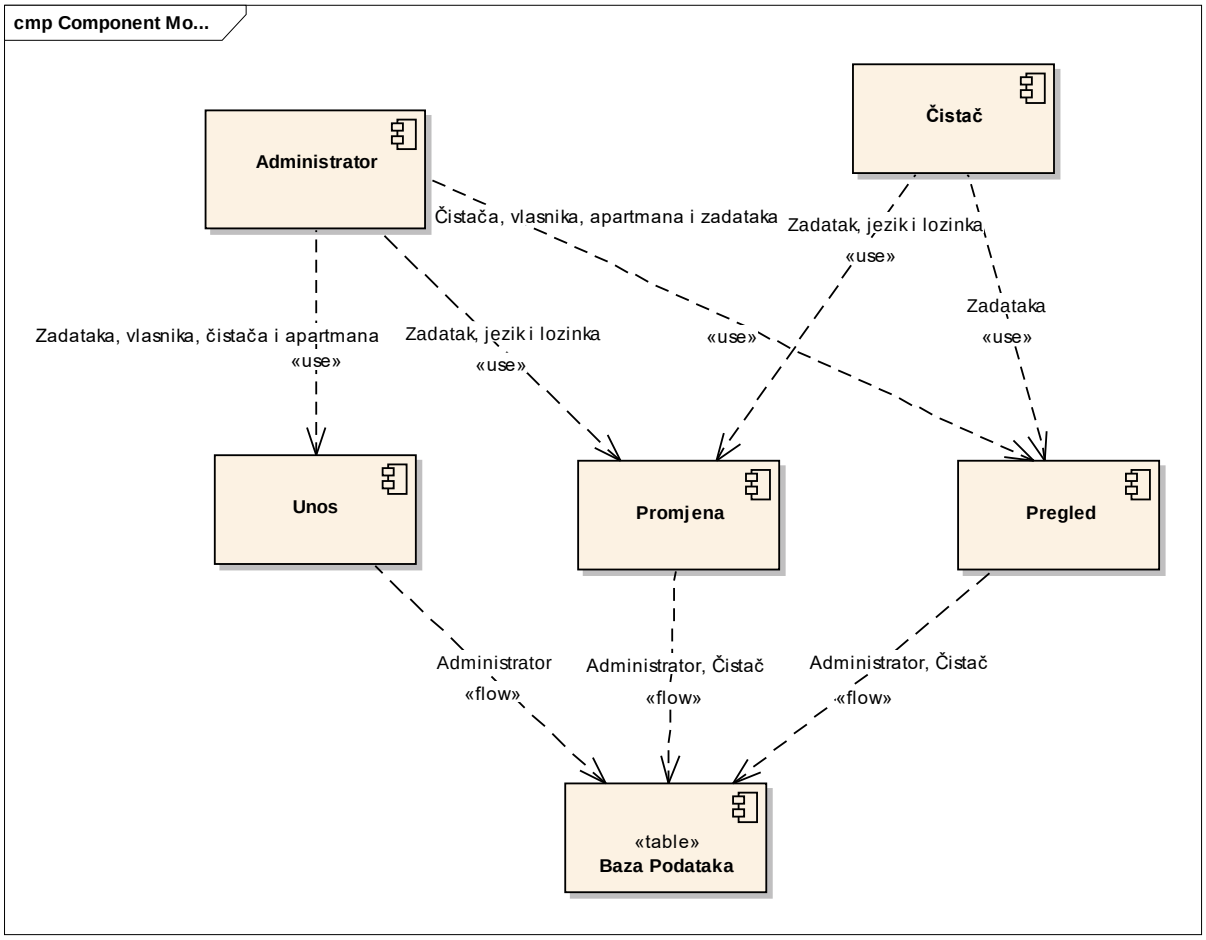
3.4 Ostali zahtjevi

Kriptiranje – lozinka se kriptira uz pomoć PHP funkcije password_hash s algoritmom bcrypt.

Podrška – podršku se obavlja putem emaila, pozivom ili kroz dokumentaciju.

4 PREGLED KOMPONENATA I VEZA

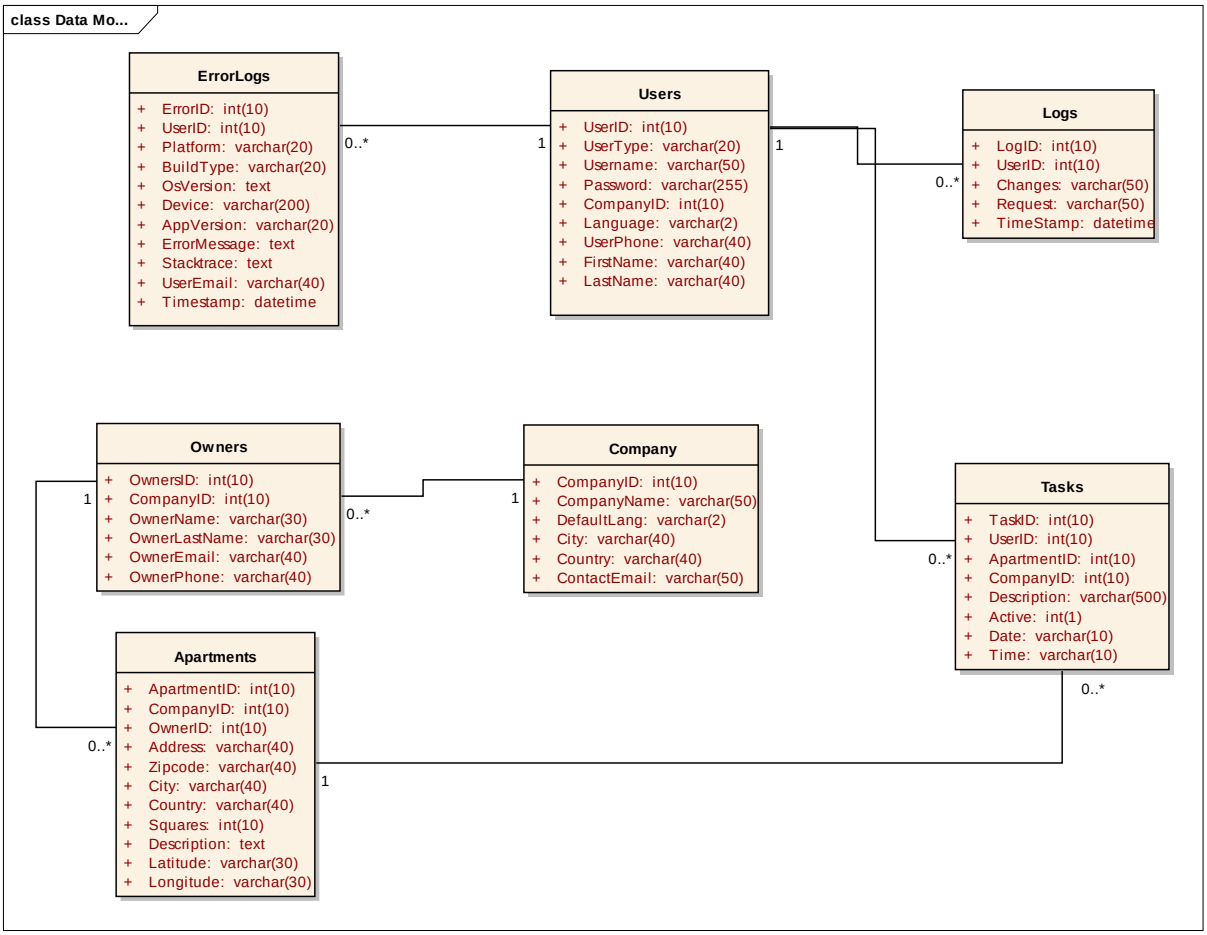
Na slici 17. prikazan je dijagram komponenti preko koje se ostvaruje uvid u rad sustava.



Slika 17. Dijagram komponenti

4.1 Logički model baze podataka

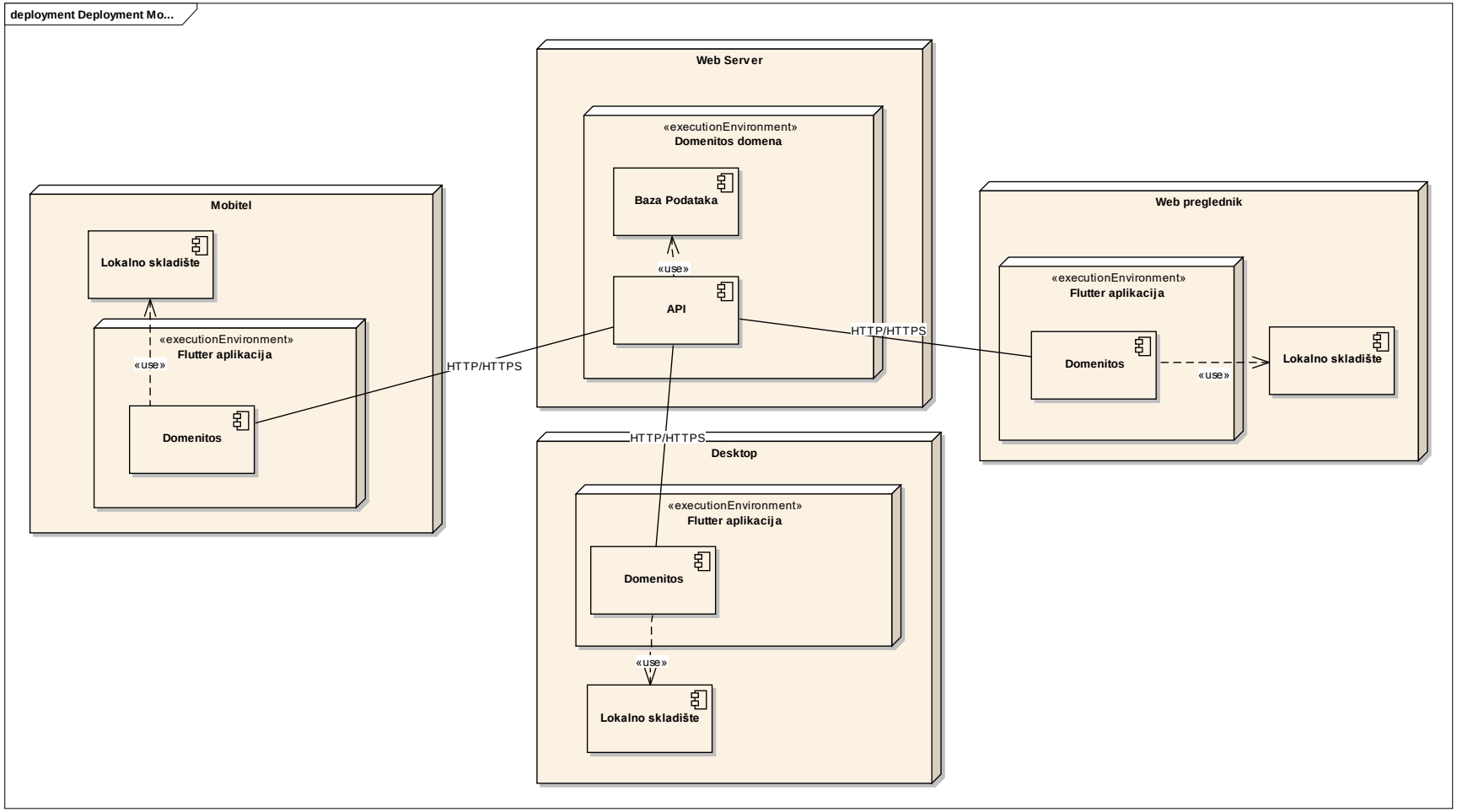
Na slici 18. prikazan je logički model baze podataka.



Slika 18. Model baze podataka

4.2 Implementacija sustava

Implementacija sustava se vrši na tri platforme: mobilna, desktop i web. Na slici 19. se vidi dijagram razmještaja aplikacije.

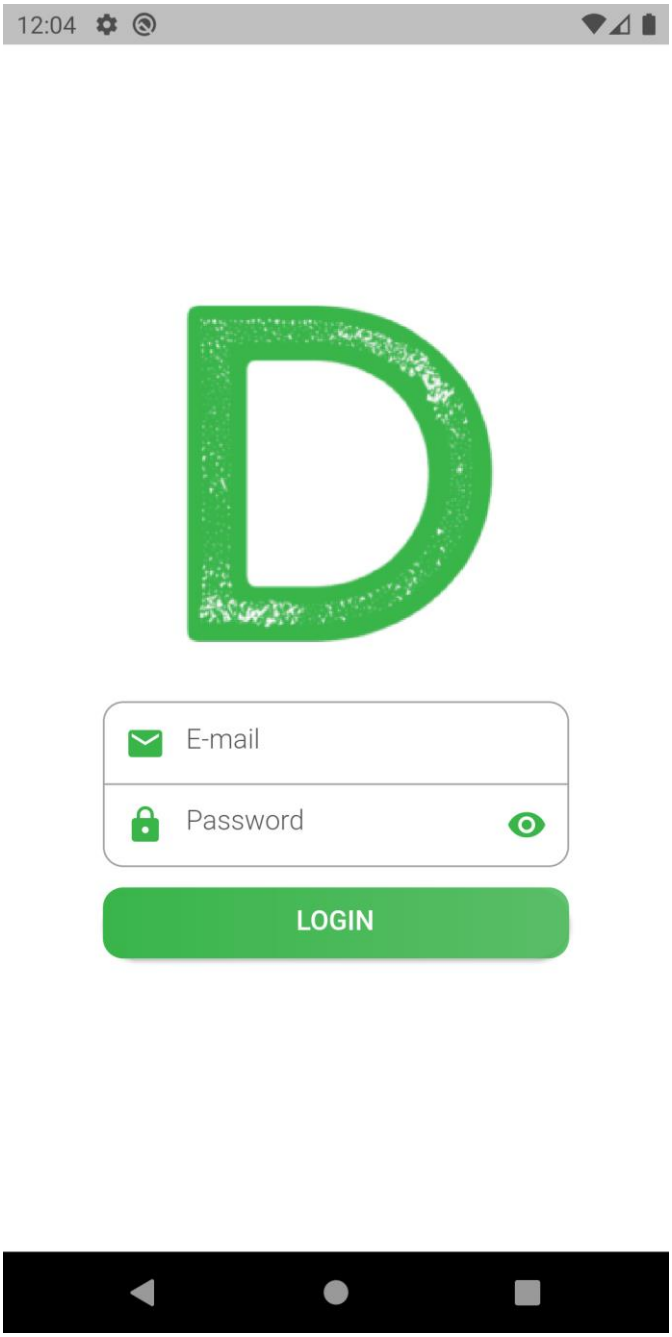


Slika 19. Dijagram razmještaja

5 OPIS RAZLIČITIH PLATFORMI

5.1 Mobilna aplikacija Domenitos

Funkcionalnosti aplikacije *Domenitos* će biti prikazane na primjeru mobilne aplikacije koja je prva izrađena korištenjem opisanog sustava. Podaci se spremaju u relacijskoj bazi na web poslužitelju. Mobilna aplikacija putem API-ja REST metodama komunicira s bazom i sprema odnosno dohvaća podatke. Funkcionalnosti se mogu podijeliti na dva podsustava administracijski i čistački. Baza podataka se nalazi na webu i preko napravljenog API-ja komunicira s mobilnom aplikacijom. Prije ulaska u administracijski ili čistački podsustav, prikazuje se Login ekran prikazan na slici 20.



Slika 20. Login ekran

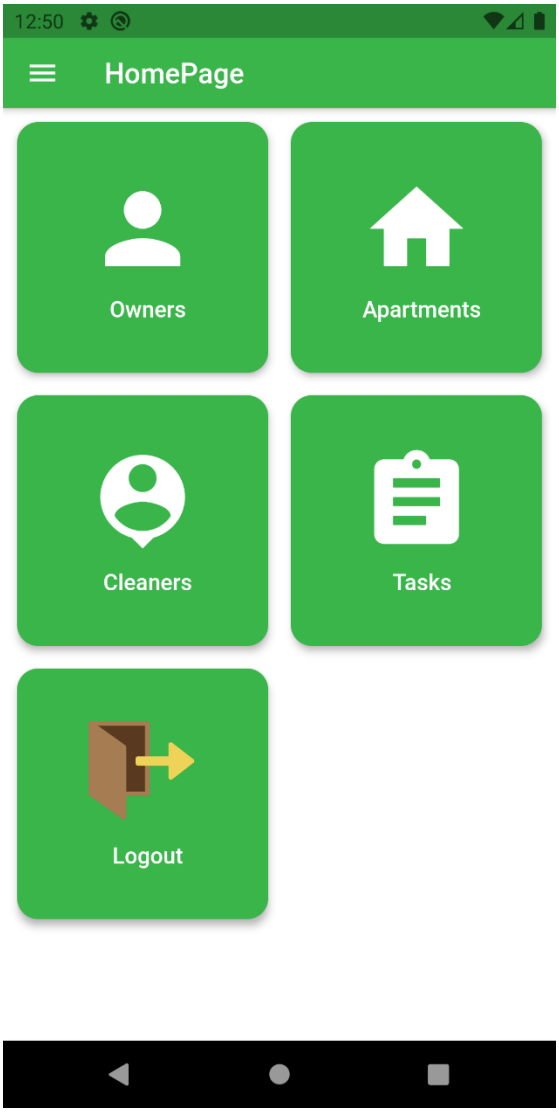
Nakon uspješne prijave u sustav podešavaju se svi relevantni podaci koji su potrebni za daljnje korištenje aplikacije poput zaporke i korisničkog imena koji se koristi za provjeru identiteta, jezik za postavke jezika, korisnički tip za prepoznavanje dozvola.

5.1.1 Administracijski podsustav

Ulaskom u administracijski podsustav omogućene su sve opcije sustava. Kroz sljedeća poglavlja ćete ih upoznati.

5.1.1.1 Početni ekran

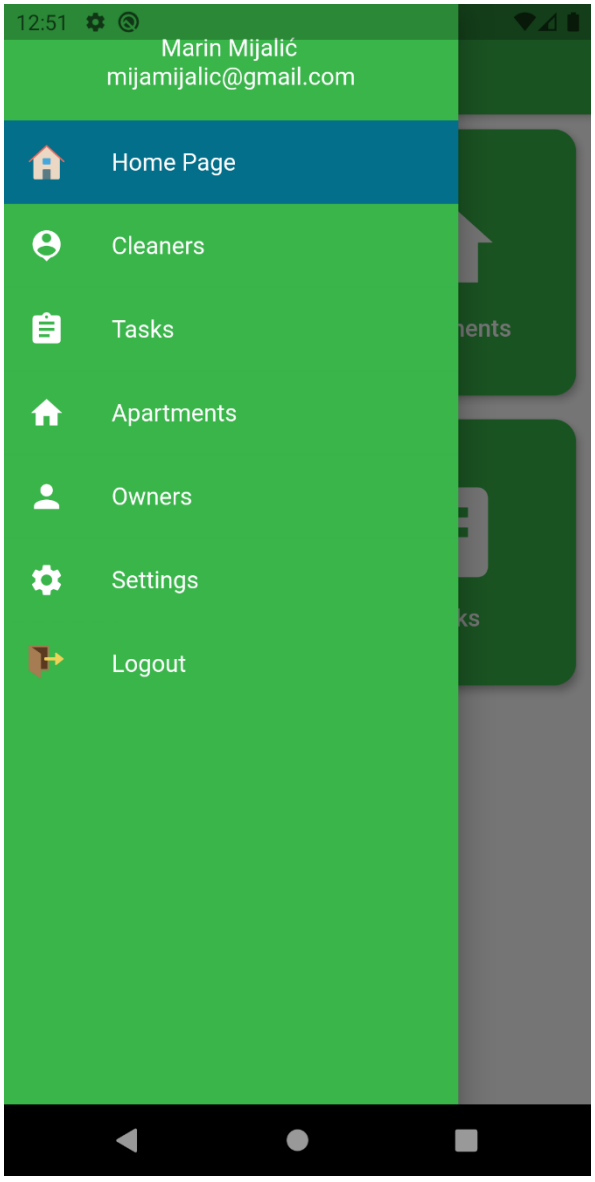
Nakon prijave korisnik dolazi na početni ekran na kojemu postoji pet opcija za kretanje kroz aplikaciju. Prva opcija je „vlasnici“ koja vodi do popisa svih vlasnika. „Apartmani“ koja vodi do popisa svih apartmana uz filter po vlasniku. „Čistači“ koji vodi do popisa svih čistača, „zadaci“ koja sadrži popise svih zadataka i „odjava“ koja odjavljuje korisnika. Na slici 21. vidi se početni ekran



Slika 21. Početni ekran

5.1.1.2 Drawer

U svakom dijelu aplikacije pritiskom na *hamburger* ikonu ili povlačenje prsta od lijevog dijela ekrana prema desnome se otvara izbornik za navigaciju kroz aplikaciju (engl. *drawer*). *Drawer* služi za olakšanu navigaciju kroz aplikaciju i uvid u dodatne dijelove aplikacije koji nestanu na početnom ekranu poput postavki. Na vrhu je vidljivo ime i prezime korisnika s njegovim emailom. Na slici 22. prikazan je *Drawer* aplikacije.

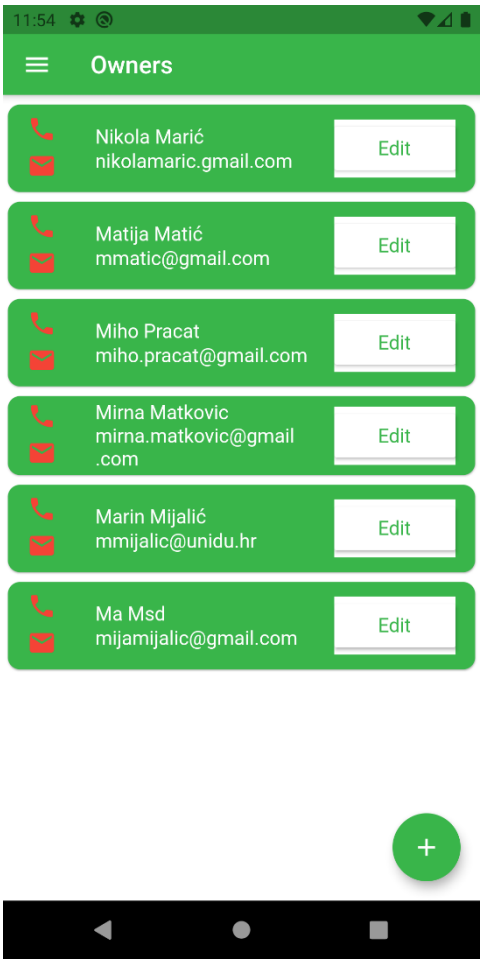


Slika 22. *Drawer*

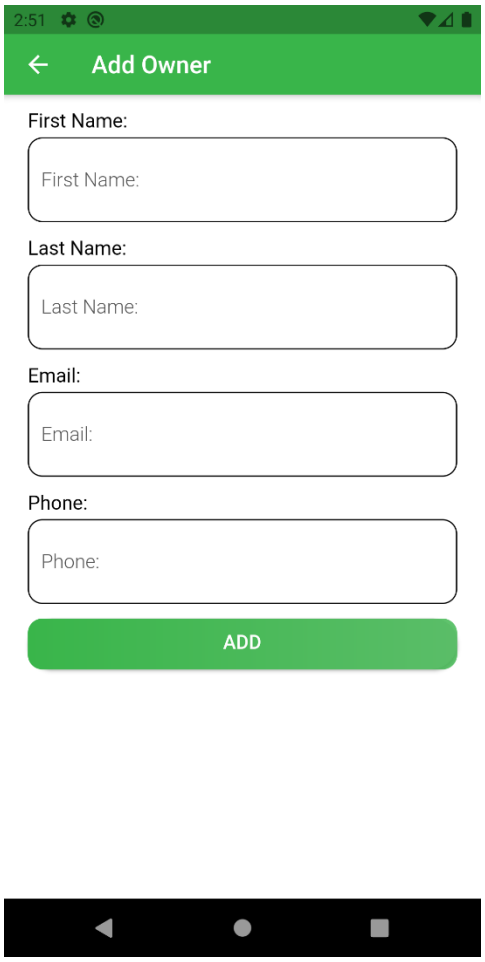
5.1.1.3 Ekran s podacima vlasnika

Vlasnici su fizičke osobe koje posjeduju apartmane kojima poduzeće pruža usluge čišćenja. Ekran vlasnici ima popis svih vlasnika unutar kompanije. Od vlasnikovih podataka vidljivi su njegovo ime i prezime te email adresa. Pritiskom na ikonu slušalica moguće je direktno nazvati vlasnika ili pritiskom ikone email adrese poslati email. Klikom na akcijsku tipku plus otvara se ekran na kojemu se može dodati novi vlasnik. Klikom na tipku Edit može se ažurirati vlasnik. Nakon ispunjenja forme i pritiskom tipke Add novi vlasnik se dodaje. Klikom na karticu

vlasnika, navigacija direktno vodi na ekran apartmani s listom apartmana tog vlasnika. Na slici 23. prikazan je ekran pregleda vlasnika, a na slici 24. ekran za dodavanje novog vlasnika.



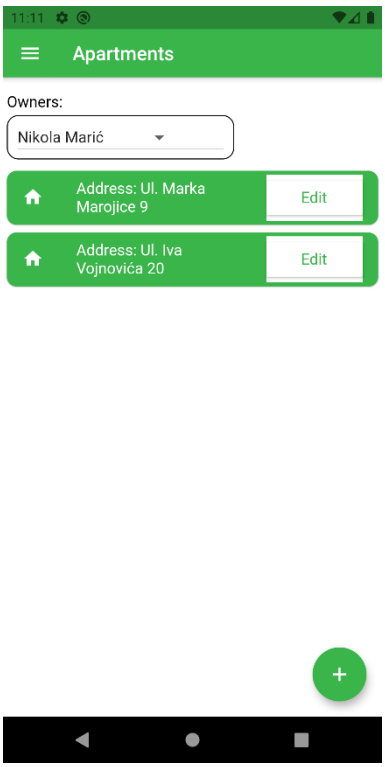
Slika 23. Ekran Vlasnici



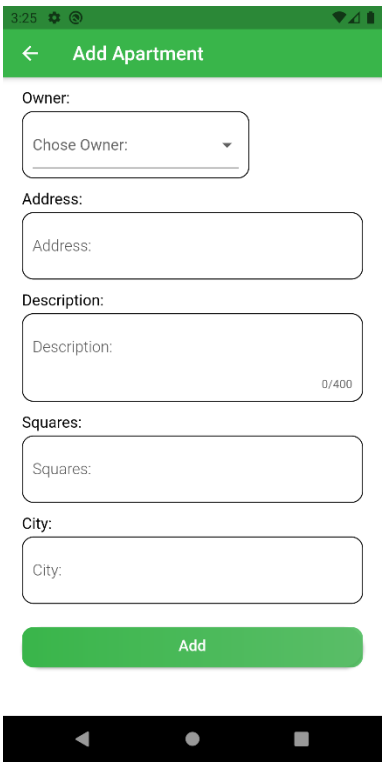
Slika 24. Ekran dodavanje vlasnika

5.1.1.4 Ekran Apartmani

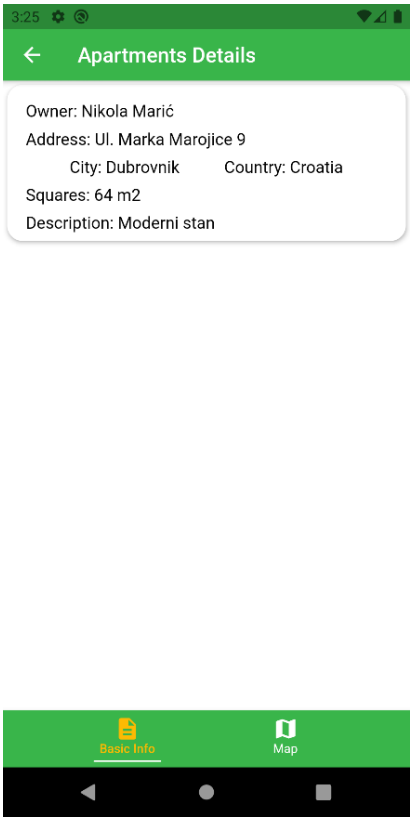
Ekran apartmani prikazuje listu svih apartmana u aplikaciji. Apartmani se listaju po filteru vlasnici. Filter je napunjen svim vlasnicima koji su u kompaniji. Za svaki apartman je ispisana adresa i pritiskom na njegovu karticu navigacija vodi do detalja o tom apartmanu i njegove Google Map lokacije. Pritiskom na akcijsku tipku plus otvara se forma za dodavanje novog apartmana na kojoj se može pridružiti novi apartman nekom vlasniku. Klikom na tipku Edit može se ažurirati apartman. Preko polja adrese i grada se pomoću *geocodera* nalaze podaci o geografskoj lokaciji (paralela i medijan) te se s tim podacima omogućuju funkcionalnost pregleda na Google Mapama. Na slici 25. vidljiv je ekran apartmani, na slici 26. dodavanje novog apartmana, na slici 27. detalji izabranog apartmana, a na slici 28. lokacija apartmana prikazana na Google Mapi.



Slika 25. Ekran apartman



Slika 26. Ekran dodaje apartmana

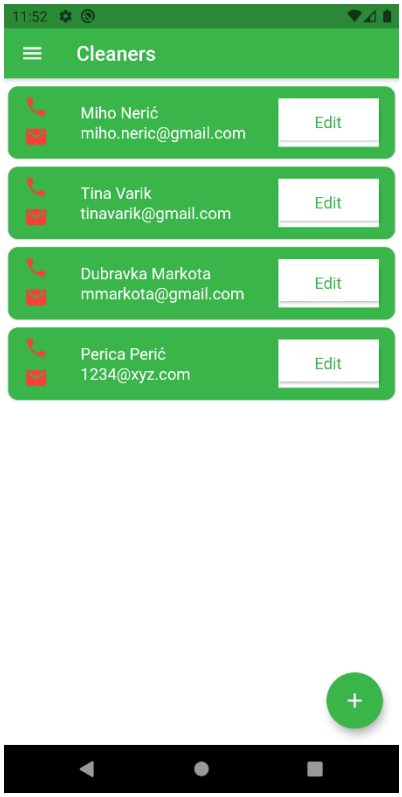


Slika 27. Detalji apartmana

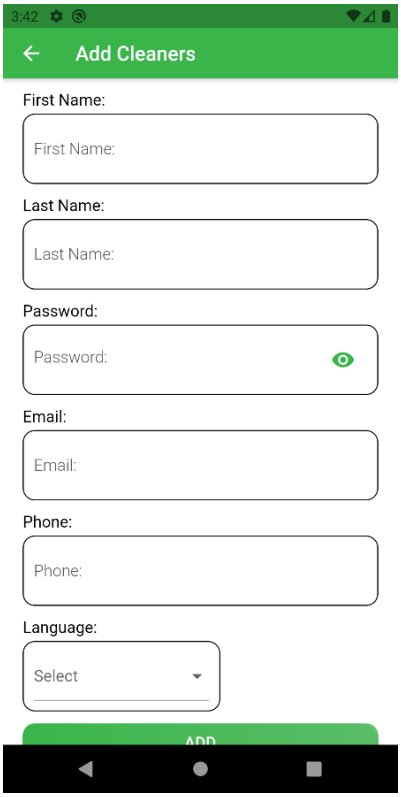
Slika 28. Google Map lokacija apartmana

5.1.1.5 Ekran Čistači

Ekran čistači prikazuje listu svih čistača u kompaniji i kao ekran vlasnici prikazuje ime i prezime te omogućuje direktan poziv ili slanje emaila čistaču. Uz pomoć akcijske tipke plus otvara se forma za dodavanje novoga čistača. Klikom na tipku Edit mogu se ažurirati podaci čistača. Potrebno je naglasiti da se prilikom dodavanja novog čistača ujedno i pravi novi korisnik aplikacije. Na slici 29. je vidljiv Ekran Čistači , a na slici 30. dodavanje podataka novog čistača.



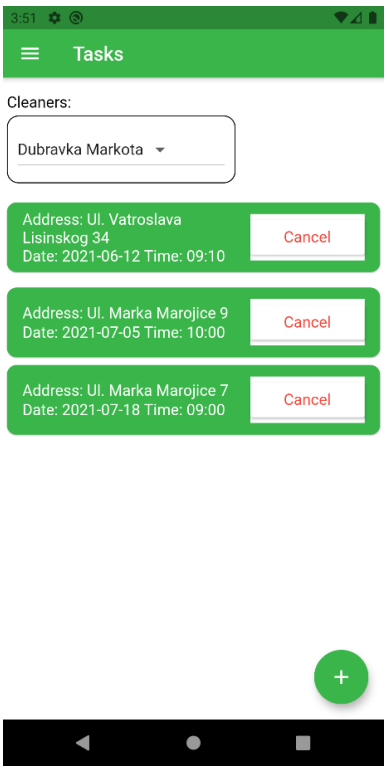
Slika 29. Ekran Čistači



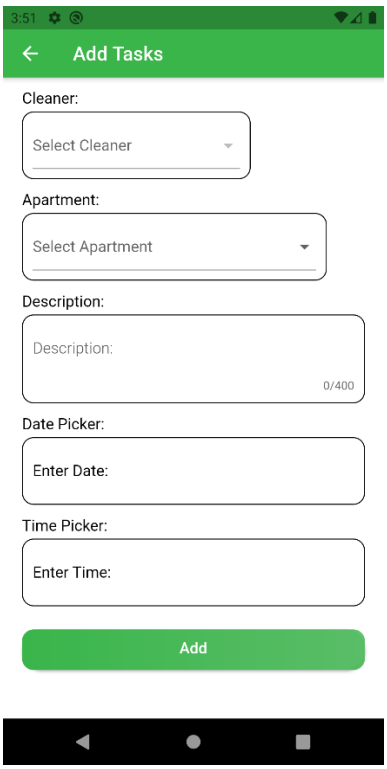
Slika 30. Dodaj Čistača

5.1.1.6 Ekran Zadaci

Ekran zadaci omogućava pregled svih aktivnih zadataka po filteru čistači. Za svaki zadatak je vidljiva adresa, vrijeme i datum te pritiskom na karticu se otvara Google Maps lokacija zadataka. Moguće je zaustaviti zadatak pritiskom na tipku Cancel. Pritiskom na akcijsku tipku plus otvara se forma za dodavanja novoga zadataka. Na slici 31. prikazan je Ekran Zadaci, a na slici 32. ekran za dodavanje novog zadatka.



Slika 31. Ekran Zadaci

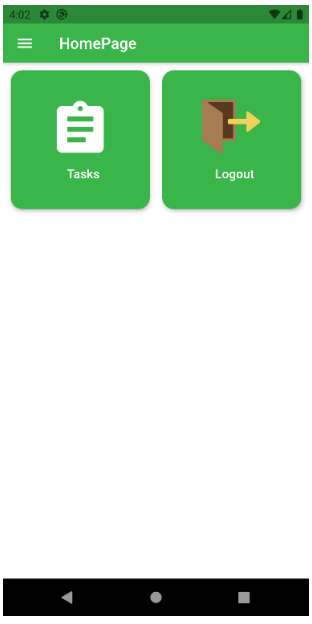


Slika 32. Dodaj zadatak

5.1.2 Podsustav Čistači

5.1.2.1 Početni ekran

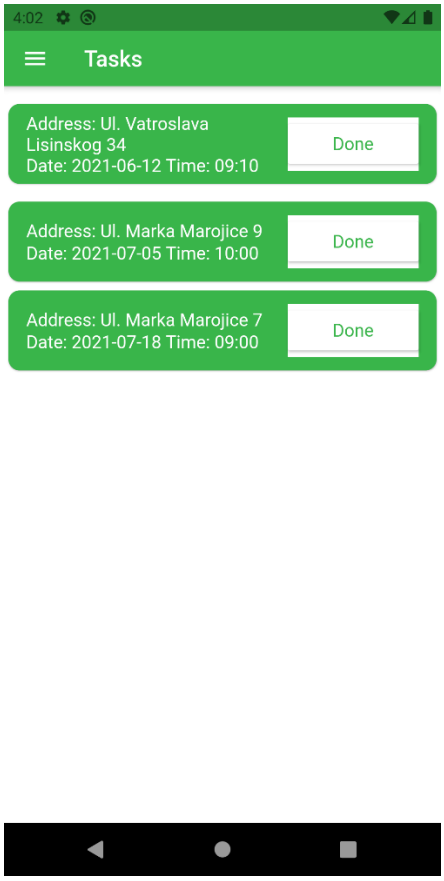
Za razliku od administratora početni ekran kod podsustava čistači ima samo uvid u zadatke i odjavu. Na slici 33. prikazan je početni ekran podsustava čistači.



Slika 33. Početni ekran podsustav čistači

5.1.2.2 Ekran zadaci-čistača

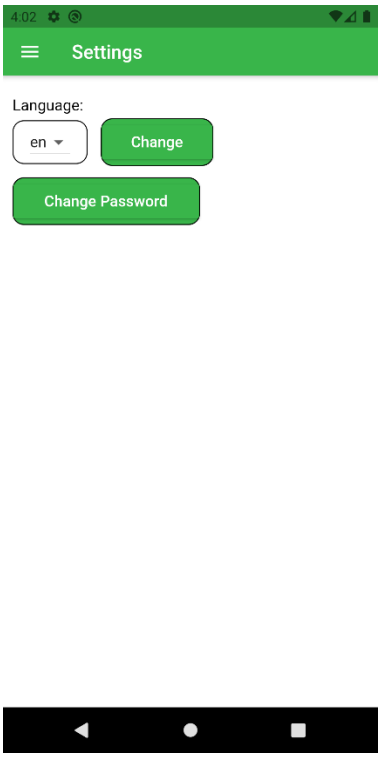
Na ekranu zadaci, čistač vidi popis svih aktivnih zadataka. Kao i kod administrativnog podsustava čistač može klikom na karticu vidjeti lokaciju zadataka. Pritiskom tipke Done čistač signalizira sustavu da je izvršio zadatak. Na slici 34. vidljiv je Ekran zadaci čistača.



Slika 34. Ekran zadaci čistača

5.1.3 Zajednički ekran postavki

Ekran postavki omogućuje administratoru ili čistaču promjenu jezika aplikacije ili promjenu svoje lozinke. Na slici 35. prikazan je izgled ekrana postavki.



Slika 35. Ekran postavki

5.1.4 Analiza koda

U ovome poglavlju ćemo objasniti neke dijelove izvornoga koda.

5.1.4.1 Navigacija

Navigacija je napravljena uz pomoć konstrukta typedef *RouteFactory* u kojoj se definiraju svi ekrani unutar aplikacije i koje podatke primaju. Nakon definicije svaka ruta se može dohvatiti uz pomoć funkcije *Navigator*. Na slici 36. prikazan je programski kod *RouteFactory*.

```
RouteFactory routes(){
  return (settings) {
    final Map<String, dynamic> arguments = settings.arguments;
    Widget screen;
    switch (settings.name){
      case LoginRoute:
        screen = Login();
        break;
      case HomePageRoute:
        screen = HomePage(arguments['userLang'],arguments['userType']);
        break;
      case OwnersRoute:
        screen = Owners(arguments['userLang'],arguments['userType']);
        break;
```

Slika 36. Kod *RouteFactory*

5.1.4.2 JSON API

Aplikacija komunicira s bazom podataka na web poslužitelju metodama HTTP protokola, razmjenom podataka u JSON formatu. Cijeli API za komunikaciju se nalazi u jednoj datoteci handler.php. U njoj su definirani svi zahtjevi kao konstante te se prilikom svakog zahtjeva prvo radi provjera identiteta korisnika te se tek nakon toga obrađuje zahtjev i šalje povratna informacija. Aplikacijski dio šalje ili prima informacije. Informacije koje prima mora dekodirati iz JSON formata i iskoristiti ih po potrebi. Na slici 37. prikazan je programski kod slanja upita o pravilnosti prijave.

```
Future <bool> LoginConformation(String username,String password) async {

  var body = '{ "DomenitosLoginRQ": { "authentication": {"password":"$password","username":"$username"} } }';
  var response = await http.post(url,body: body);
  var check = false;

  if(response.statusCode == 200){
    Map<String,dynamic> myMap = json.decode(response.body);
    var mapResponse =  myMap["DomenitosLoginRS"]["success"];
    if(mapResponse == null){
      check = false;
    }else{
      check = true;
    }
  }
  return check;
}
```

Slika 37. Login conformation

5.1.4.3 ErrorLogger

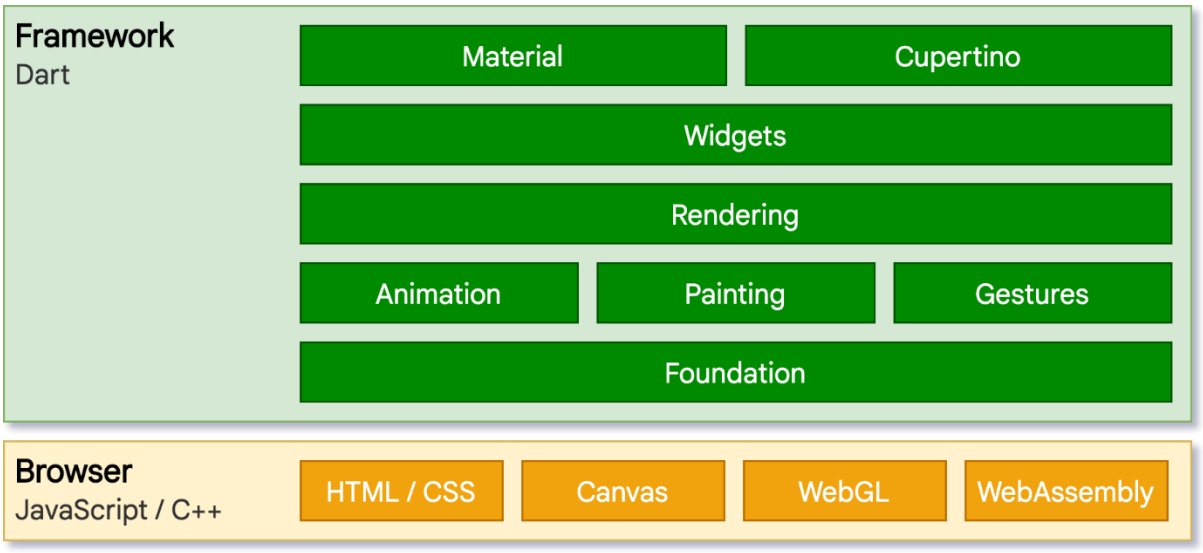
ErrorLogger je klasa koja omogućuje praćenje svih grešaka unutar aplikacije te njihovo slanje prema serveru i upisivanje u bazu podataka. Klasa se nalazi u Error_Logger.dart datoteci.

5.2 Web aplikacija *Domenitos*

Flutter omogućuje izradu web aplikacije s istom kodnom bazom korištenom kod već izrađene mobilne aplikacije opisane u prethodnim poglavljima. Kroz ovo poglavlje cilj je dobiti uvid u tehnologije u pozadini *Fluttera* koje omogućavaju izradu web stranica, proces implementacije, izgradnje i postavljanje aplikacije na poslužitelj.

5.2.1 Tehnologije web *Fluttera*

Da bi *Flutter* web aplikacija donijela isto iskustvo krajnjem korisniku na mobilnoj i web aplikaciji koristi se arhitektura prikazana na slici 38.



Slika 38. Arhitektura web *Fluttera* [8]

Na gornjoj slici arhitektura je podijeljena na dva dijela. Prvi dio je okvir programskog jezika *Dart* u kojem se izvršava sva programska logika, izgled, dizajn i konekcija sa poslužiteljem putem API-ja. Drugi dio arhitekture se tiče iscrtavanja sučelja u web pregledniku [9]. *Flutter* web koristi cijeli zaslon kao platno i stvara vlastite HTML elemente i tako daje programeru potpunu kontrolu nad svakim pikselom. Pošto u *Flutteru* iscrtavanje na strani poslužitelja nije moguće jer aplikacija komunicira s pozadinom sustava API-jima, *Flutter* koristi svoje *rendering engine*:

DomCavas je osnovi model temeljen na HTML-u. To je objektni model koji kombinira tehnologije HTML, CSS, JS i API *Canvas* za izradu izgleda i bojanje *Flutter* widgeta na webu.

CanvasKit koristi grafički *engine* Skia napravljen od strane Googlea koji omogućuje 2D grafiku na Google Chromeu, Chrome OS-u, Androidu, Flutteru, Mozilla Firefoxu, Firefox OS-u i mnogim drugim programskim sustavima [10]. U našem slučaju ona koristi tehnologije Web Assembly i WebGL što omogućuje web pregledniku bolje korištenje hardverskog ubrzanja [11].

5.2.2 Performanse *rendering engine*

Performanse weba se mjere dvjema stavkama. Prva stavka je sposobnost generiranja i manipulacija velikim količinama podataka, a druga stavka su prijelazi, animacije i efekti. Programski jezik *Dart* dobro podnosi velike liste s minimalnim zastajanjima. U prethodnome poglavlju smo naveli dva modela za renderiranje DomCanvas i CanvasKit. DomCanvas će imati četiri puta manju veličinu preuzimanja pri inicijalnom otvaranju web stranice u odnosu na CanvasKit, što znači da će se DomCanvas znatno brže učitati pri ulasku na web stranicu. CanvasKit je dosta veći ali omogućuje brže buduće učitavanje i glatkije animacije, efekte i prijelaze unutar web sjedišta. Odluka koji *rendering engine* treba koristi ovisi primarno o veličini projekta i njegovoj složenosti. Primjerice web stranice koje se sastoje od jedne stranice su dosta male te bi se na njima trebao koristiti DomCanvas. Za složenije sustave bolje je koristiti CanvasKit [11].

5.2.3 Korištenje web *Fluttera*

Flutter pruža kvalitetno i učinkovito korisničko iskustvo na svim modernim web preglednicima. Iako je moguće napraviti svaku vrstu web stranica web *Flutter* je najpoželjniji u sljedećim situacijama:

Progresivna web aplikacija – Uz pomoć *Fluttera* mogu se izraditi kvalitetne progresivne web aplikacije, koje su integrirane s korisničkim okruženjem uključujući instalaciju, izvanmrežnu podršku i prilagođeno korisničko iskustvo.

Jednostrane web aplikacije – aplikacije koje koriste dinamičko prepisivanje trenutačne web stranice novim podacima s web poslužitelja.

Već postojeće mobilne aplikacije – ako postoji potreba za prebacivanjem mobilne aplikacije za prikaz na web poslužiteljima.

Sustavi – svaki sustav koji treba imati web i mobilnu aplikaciju koje su dosta slične.

Nepoželjne situacije u kojima se ne bi trebao koristi web *Flutter* su kompleksne web stranice koje zahtijevaju odličan SEO (Optimizacija pretraživača engl. *Search Engine Optimization* - SEO). Web stranice trenutačno ne mogu imati dobar SEO jer se mogu samo posložiti meta tagovi na indeks stranici. Na ostalim mjestima to nije moguće jer se inicijalizacija ostalih web stranica unutar web sjedišta izvodi kroz hashove pa ih trenutačni Googleov pretraživač nikada neće inicijalizirati. Taj podatak isključuje web stranice poput foruma, blogova, novina, agencijske web stranice i slično [9].

5.2.4 Implementacija web aplikacije *Domenitos*

Treći ožujka 2021. plasiran je Flutter 2 koji je omogućio stabilnu verziju web *Fluttera*. Implementacija web aplikacije počinje s detekcijom aktivnih uređaja. Bez aktivnih uređaja nije moguće izvršiti kod i testirati ga. Na slici 39. je vidljiva naredbeni redak za detekciju aktivnih uređaja.

```
D:\diplomski\flutter_app>flutter devices
2 connected devices:

Chrome (web) • chrome • web-javascript • Google Chrome 92.0.4515.159
Edge (web)   • edge   • web-javascript • Microsoft Edge 92.0.902.84
```

Slika 39. Naredbeni redak za detekciju uređaja

Bitno je naglasiti da, dok je moguću prikazati i pokrenuti aplikaciju na svim web preglednicima samo se Google Chrome koristi za otklanjana pogrešaka. Razlog tome je što funkcije za ispis i pronalazak pogrešaka rade samo uz pomoć Google Chrome developer alata.

Sljedeći korak je napraviti prvu verziju web aplikacije. Na slici 40. vidljiv je naredbeni redak za izvođenje web aplikacije.

```
D:\diplomski\flutter_app>flutter run -d chrome
```

Slika 40. Naredbeni redak za izvođenje web aplikacije

5.2.5 Problemi tijekom izrade web aplikacije

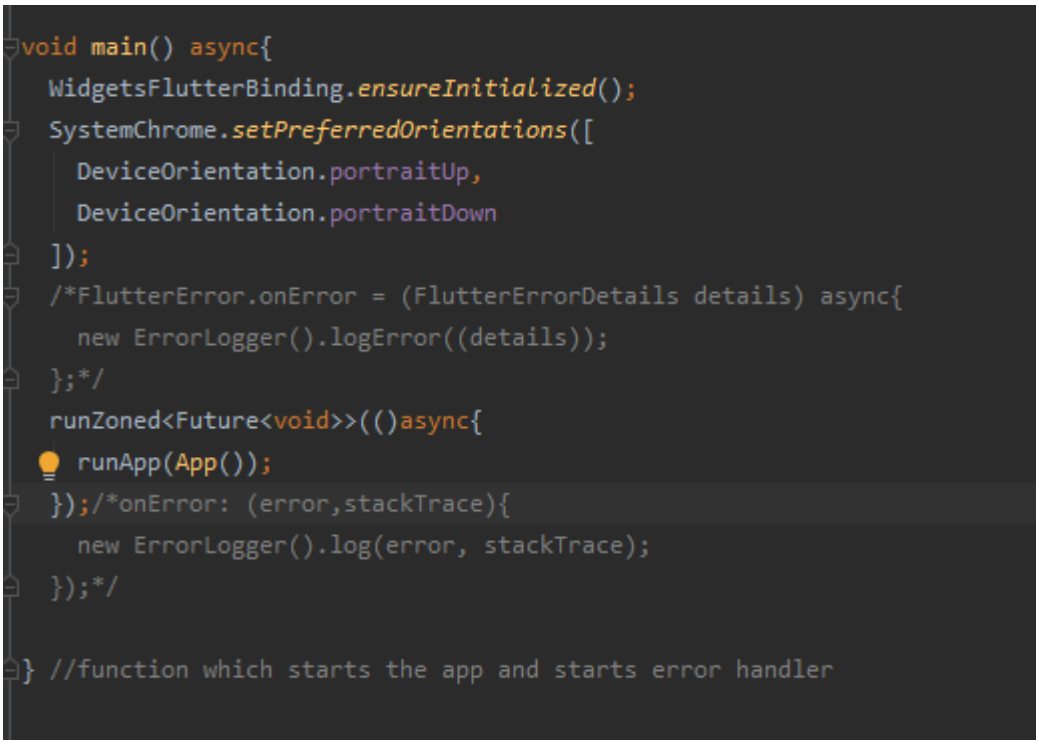
Prva izvedba naredbenog retka na slici 40. je izvršena neuspješno i ispisane su stotine redova grešaka. Većina pogrešaka je bila vezana s korištenim bibliotekama. Na slici 41. su vidljive sve neosnovne biblioteke korištene u projektu.

```
dependencies:
  flutter:
    sdk: flutter
  http: ^0.13.1
  url_launcher: ^6.0.6
  #device_info: ^2.0.2
  #package_info: ^2.0.2
  #flutter_secure_storage: ^4.2.0
  shared_preferences: ^2.0.7
  progress_dialog: ^1.2.0
  fluttertoast: ^8.0.7
  flutter_svg: ^0.22.0
  google_maps_flutter: ^2.0.6
  geocoder: ^0.2.1
  flutter_datetime_picker: ^1.5.1
  flutter_launcher_icons: ^0.9.0
```

Slika 41. Korištene biblioteke

Device_info i package_info biblioteke su koje se isključivo koriste u mobilnoj izradi aplikacije *Domenitos*. Služile su za moguću detekciju pogrešaka tijekom cijelog korištenje mobilne aplikacije u obliku klase ErorrLogger unutar datoteke Error_Logger.dart. Svako korištenje te klase se moralo zakomentirati. Specifično, biblioteka device_info je prikupljala sve relevantne podatke o uređaju na kojem se dogodila pogreška. Package_info je davala je povratne informacije o imenu, verziji i ostalim bitnim stvarima o aplikaciji. Uz pomoć ove dvije biblioteke jednostavno je dobiti cijeli *stacktrace* pogrešaka i shvatiti iz njega na kojem ekranu, uređaju i kojem trenutku se dogodila pogreška. Svaka detektirana pogreška je automatski poslana API-jem u bazu podataka. Zbog nemogućnosti korištenja navedenih biblioteka prikupljanje pogrešaka na sučelju više nije automatsko nego se po potrebi treba dodavati try i catch funkcije.

Taj problem najbolje je vidljiv u main.dart datoteci. Na slici 42. vidljiv je zakomentirani dio main.dart datoteke.



Slika 42. Zakomentirani dio main.dart datoteke

Web *Flutter* zahtjeva samostalno prilagođeno rješenje za bilježenje pogrešaka u bazu podataka. To može znatno produžiti razvojno vrijeme ovisno o složenosti sustava.

Biblioteka flutter_secure_storage daje mogućnost šifriranog lokalnog spremišta. Iako u dokumentaciji biblioteke piše da se može koristiti za web, u korištena beta verzija stalno je izbacivala pogrešku „*TypeError: Cannot read property 'Symbol(dartx.is Not Empty)' of null*“. Pogreška znači da se ništa ne sprema u lokalno spremište i s time se nikakav podatak koji je korišten čitanjem iz lokalnog spremišta nije mogao učitati. Zbog ove pogreške potrebno je prijeći na biblioteku shared_preferences koja radi bez pogreške na webu i mobilnoj aplikaciji. Bitno je naglasiti da biblioteka shared_preferences ne omogućava šifriranje lokalnih podataka što ju čini nesigurnijom opcijom. Na slici 43. vidljiva je razlika u kodu između biblioteka.

```
/* storage.write(key: 'firstName', value: firstName);
 storage.write(key: 'lastName', value: lastName);
 storage.write(key: 'userType', value: userType);
 storage.write(key: 'language', value: language);
 storage.write(key: 'userPhone', value: userPhone);
 storage.write(key: 'password', value: password);
 storage.write(key: 'username', value: username);
 storage.write(key: 'session', value: session);
 storage.write(key: 'companyID', value: companyId);
 storage.write(key: 'userID', value: userID);*/
 prefs.setString('firstName',firstName);
 prefs.setString('lastName',lastName);
 prefs.setString('userType',userType);
 prefs.setString('language',language);
 prefs.setString('userPhone',userPhone);
 prefs.setString('password',password);
 prefs.setString('username',username);
 prefs.setString('session',session);
 prefs.setString('companyID',companyId);
 prefs.setString('userID',userID);
```

Slika 43. Razlika između dvije biblioteke u kodu

Sljedeća pogreška je bila „*Error: XMLHttpRequest error.*“, koja ukazuje na to da ne prolaze HTTP zahtjevi. Ova pogreška će se pojaviti svima koji pokušaju koristiti HTTP zahtjeve iz web *Fluttera* i nije ju moguće ispraviti sa strane *Fluttera* tj. u programskom jeziku *Dart* nego se mora riješiti na strani poslužitelja. Uzrok leži u korištenje CORS-a koji predstavlja mehanizam zasnovan na HTTP zaglavlju, a koji omogućuje poslužitelju da obilježi koje domene, sheme ili portove osim vlastitog može primiti [12]. Problem se rješava dodavanjem linije „header("Access-Control-Allow-Origin: *");“ unutar API rukovatelja datoteke. U slučaju *Domentios* aplikacije to je datoteka handler.php. Na slici 44. prikazano je umetanje PHP funkcije koja rješava problem.

```
1  <?php
2  include('connection.php');
3  header("Access-Control-Allow-Origin: *");
```

Slika 44. Dodavanje PHP funkcije header

Funkcija header mora biti pozvana prije rukovanja s bilo kojim pozivom API-ja aplikacije. Gore navedene postavke zaglavlja dozvoljavaju API pozive sa svih mogućih izvora.

5.2.6 Osvrt na probleme u razvoju

Iako pogrešaka nema previše one predstavljaju problem u kontinuiranom razvoju aplikacije. Najveći problem je razlika u korištenim bibliotekama. Za svaku drugačiju biblioteku

potrebno je pisanje posebnog koda i komentiranje tog istog koda u prelazu iz razvoja mobilnih aplikacija na web aplikaciju. Što aplikacija postane složenija broj razlika će postajati veći i s tim povećati trajanje razvoja. Pogreška kod HTTP zahtjeva nije prihvatljiva za proizvod koji je prije šest mjeseci izašao iz beta faze razvoja. To ukazuje na relativnu nedovršenost projekta koja će se dodatno prokomentirati u poglavlju analize *Fluttera* i njegovih konkurenata.

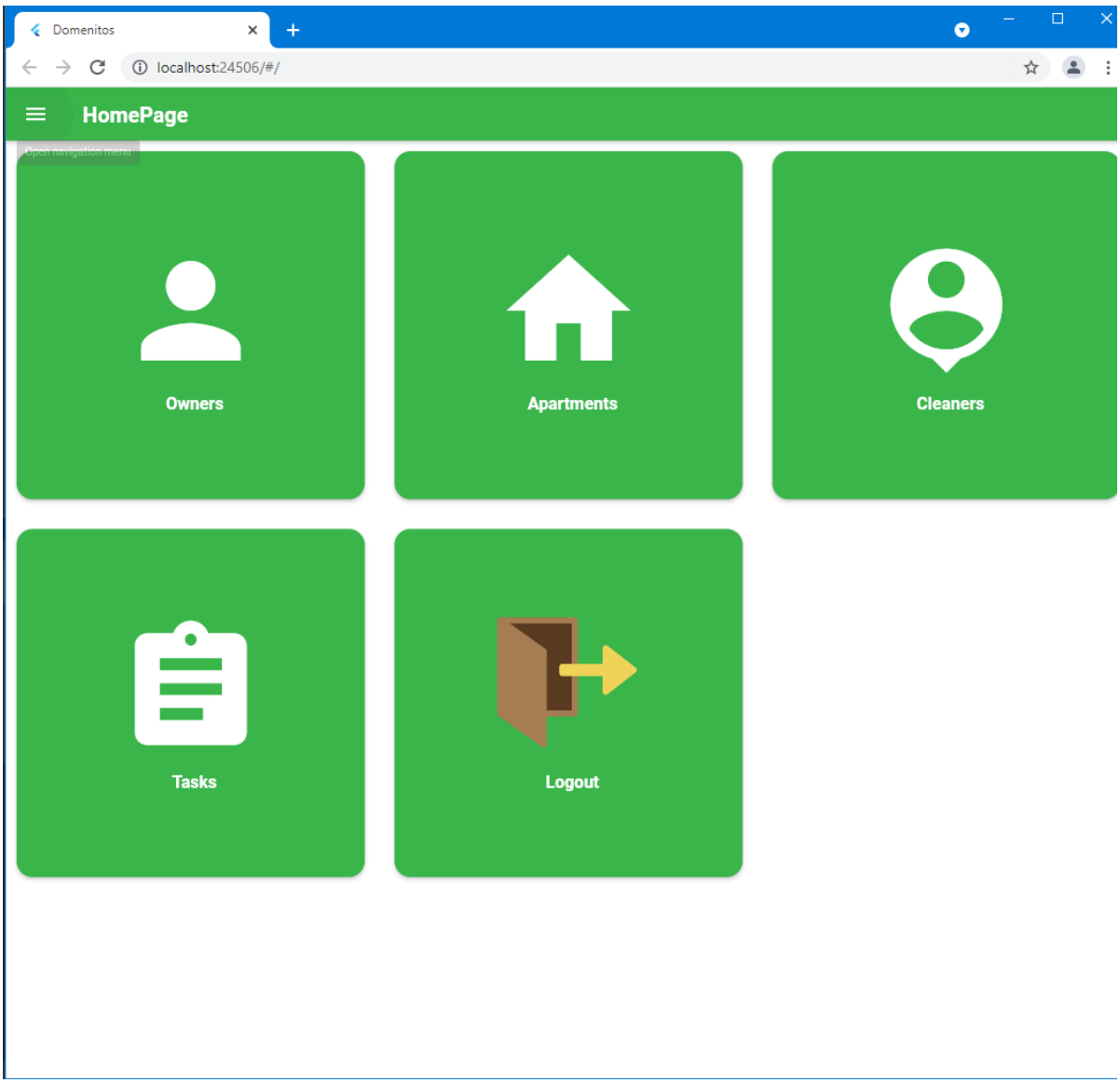
5.2.7 Dizajn web aplikacije *Domenitos*

Nakon pokretanja web aplikacije *Domenitos* vrlo brzo je bilo očito da dizajn mobilne aplikacije treba prilagoditi za web. Najlakši način prilagodbe za web je nadogradnja responzivnog dizajna koji već postoji za mobitelima i tabletu. Tijekom izrade mobilne aplikacije korištena je funkcija: MediaQuery.of(context).size.width koja uzima širinu ekrana, dodana je još jedna prijelomna točka od 1200 piksela i s time riješene prilagodba za web dizajn. Na slici 45. je vidljiv programski kod koji koristi navedenu funkciju.

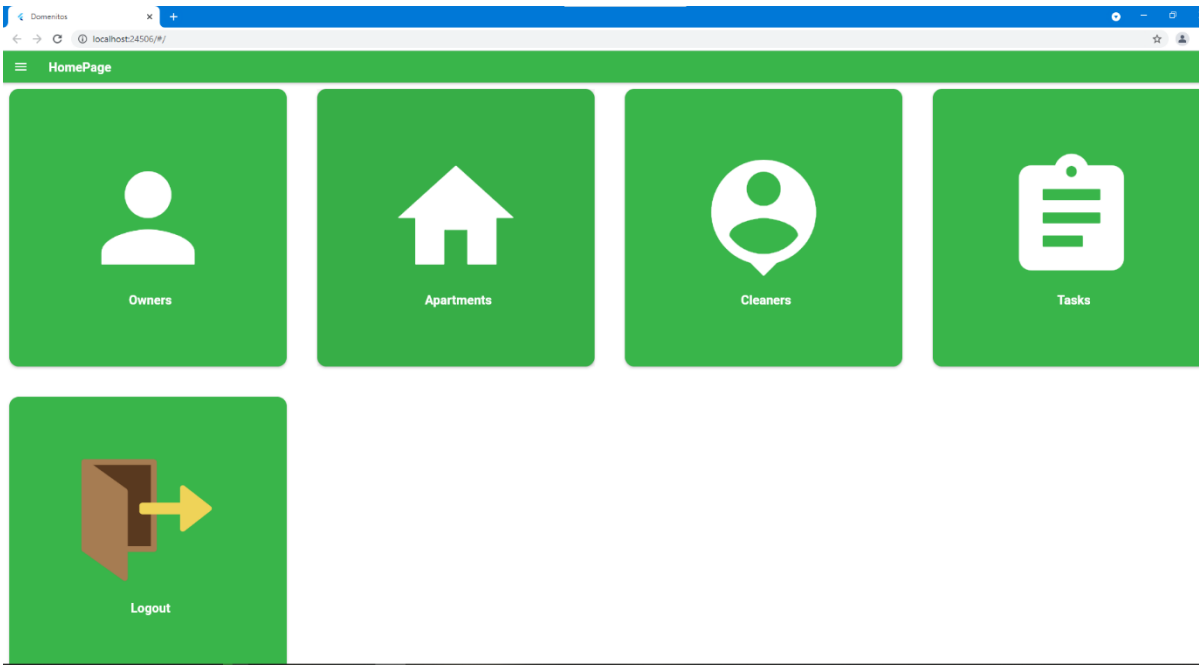
```
double width = MediaQuery
    .of(context)
    .size
    .width;
if(width > 1200 ){
  iconSize = 220.0;
  axisSpacing = 44.0;
  gridText = GridTabletTextStyle;
  crossAxisCount =4;
}else if(width <= 575){
  iconSize = 80.0;
  axisSpacing = 12.0;
  gridText = GridTextStyle;
  crossAxisCount = 2;
}else if (width >= 576){
  iconSize = 140.0;
  axisSpacing = 24.0;
  gridText = GridTextStyle;
  crossAxisCount = 3;
}
```

Slika 45. Korištenje funkcije MediaQuery

Programski kod na slici 45. s obzirom na širinu ekrana stavlja u varijable veličinu ikona, širinu između rasporeda ćelija i broj koliko po retku ima ćelija. Na slici 46. i slici 47. vidljiv je responzivan dizajn za web početnog zaslona a na slici 21. se vidi responzivan dizajn za mobilni početni ekran.



Slika 46. Responzivan dizajn širine veće od 575 piksela



Slika 47. Responzivan dizajn širine duže od 1200 piksela

5.2.8 Distribucija web aplikacije Domenitos

Nakon završetka razvoja aplikacije treba izraditi verziju aplikacije za web. Za izradu verzije koristi se naredba flutter build web. Može se odabrati koji *render engine* će se koristi uz nadodavanje –web-renderer canvaskit ili html. Razlike između renderera su objašnjene u poglavlju 5.2.1. Nakon izvršenja naredbe generira se aplikacija te se datoteke smještaju /build/web unutar direktorija samog projekta. Te se mape mogu poslati na odgovarajuće mjesto uz pomoć FTP ili sličnih protokola.

5.3 Windows aplikacija Domenitos

Podrška za stolna računala omogućuje *Flutteru* izradu desktop aplikacija za Windows, MacOS ili Linux. Desktop podrška se plasiranjem Flutter 2.0 našla u stabilno verziji *Flutter* projekta što je olakšalo pristup izradi desktop aplikacija. Bitno je naglasiti da se u stabilnoj verziji nalaze starije verzije desktop podrške jer je ovaj projekt još uvijek u beta fazi pa se njegova glavna verzija nalazi na beta verziji *Fluttera.*

5.3.1 Implementacija Windows aplikacije Domenitos

Osim standardnog flutter SDK-a za korištenje je potreban Visual Studio 2019 i to njegov specifični dio „Desktop development with C++“. Ovo je potrebno kako bi se mogao interpretirati i prevesti *Dart* kod u C++ kod [8]. Postavljanje desktop *Fluttera* započinje omogućavanjem konfiguracije uz pomoć naredbenog retka na slici 48.

```
D:\diplomski\flutter_app>flutter config --enable-windows-desktop
Setting "enable-windows-desktop" value to "true".

You may need to restart any open editors for them to read new settings.
```

Slika 48. Izvođenje naredbenog retka konfiguracije windows aplikacije

Nakon resetiranje Andoroid Studija pokreće se naredba flutter devices i kao na slici 39. vide se aktivni uređaji.

5.3.2 Kreiranje Windows aplikacije Domenitos

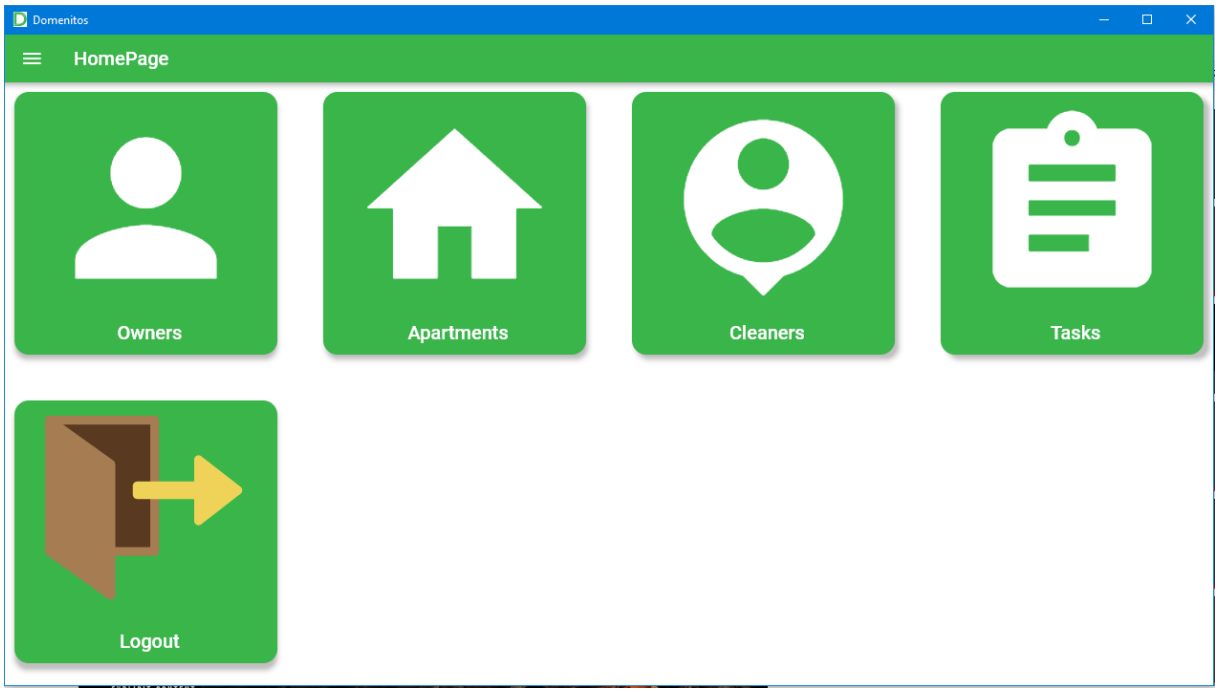
U poglavlju 5.2.5 objašnjeni su problemi s bibliotekama kod razvoja web verzije aplikacije, a isti problemi postoje na istim mjestima i pri razvoju u Windows verzije aplikacije. Dodatni problem je nastao pokušajem pokretanja aplikacije. Na slici 49. prikazane su pogreške pri pokretanju Windows aplikacije.

```
D:\diplomski\flutter_app>flutter run -d windows
Launching lib\main.dart on Windows in debug mode...
Exception: Building with plugins requires symlink support.

Please enable Developer Mode in your system settings. Run
  start ms-settings:developers
to open settings.
```

Slika 49. Pogreška pri pokretanju Windows aplikacije

Trenutačno u dokumentaciju *Fluttera* nije opisano kako dodati desktop podršku na ispravan način pa sam morao istraživati funkcionalnosti koje nisu opisane u dokumentaciji. Naredba u terminalu koja to izvodi je „flutter create .“. Ova naredba omogućuje stvaranje svih mapa koje su omogućene u konfiguraciji projekta. Nakon izvođenja ove naredbe Windows aplikacija postaje funkcionalna. Na slici 50. vidljiva je Windows aplikacija *Domenitos* i njen početni ekran.



Slika 50. Windows aplikacija Domenitos

Responzivni dizajn Windows aplikacije Domenitos se rješava na isti način kao i web dizajn, a to je objašnjeno u poglavlju 5.2.7.

5.3.3 Distribucija Windows aplikacije Domenitos

Flutter ne preporučuje distribuciju desktop aplikacija sve dok desktop podrška ne izađe iz beta faze i uđe u stabilnu fazu. Ako želimo distribuirati Windows aplikaciju postoje dva rješenja. Za to je potrebno pokrenuti naredbu flutter build windows koja pravi sve potrebne mape i ubacuje ih u projekt pod lokacijom build/windows.

Prva opcija je korištenje MSIX (Mikrosoft Windows aplikacijski format). To je dodatak koji se kad se ubaci u pubspec.yaml datoteku u projektu i koji omogućuje pakiranje uz pomoć naredbe flutter pub run msix:create. Nakon uspješnog izvođenja nastaje MSIX instalacijska datoteka koju je moguće distribuirati na Windows aplikacijsku trgovinu. Za trgovinu je potreban .pfx certifikat kojeg je moguće dobiti uz pomoć OpenSSla.

Druga opcija je ručno pravljenje zip datoteke. U zip datoteku treba staviti .exe izvršnu datoteku koja se nalazi na lokaciji projekt/build/windows/runner/release, a uz nju se u istom direktoriju nalazi sve potrebne .dll datoteke i cijela mapa aplikacije [8].

5.3.4 Problemi s podrškom za razvoj Windows aplikacija

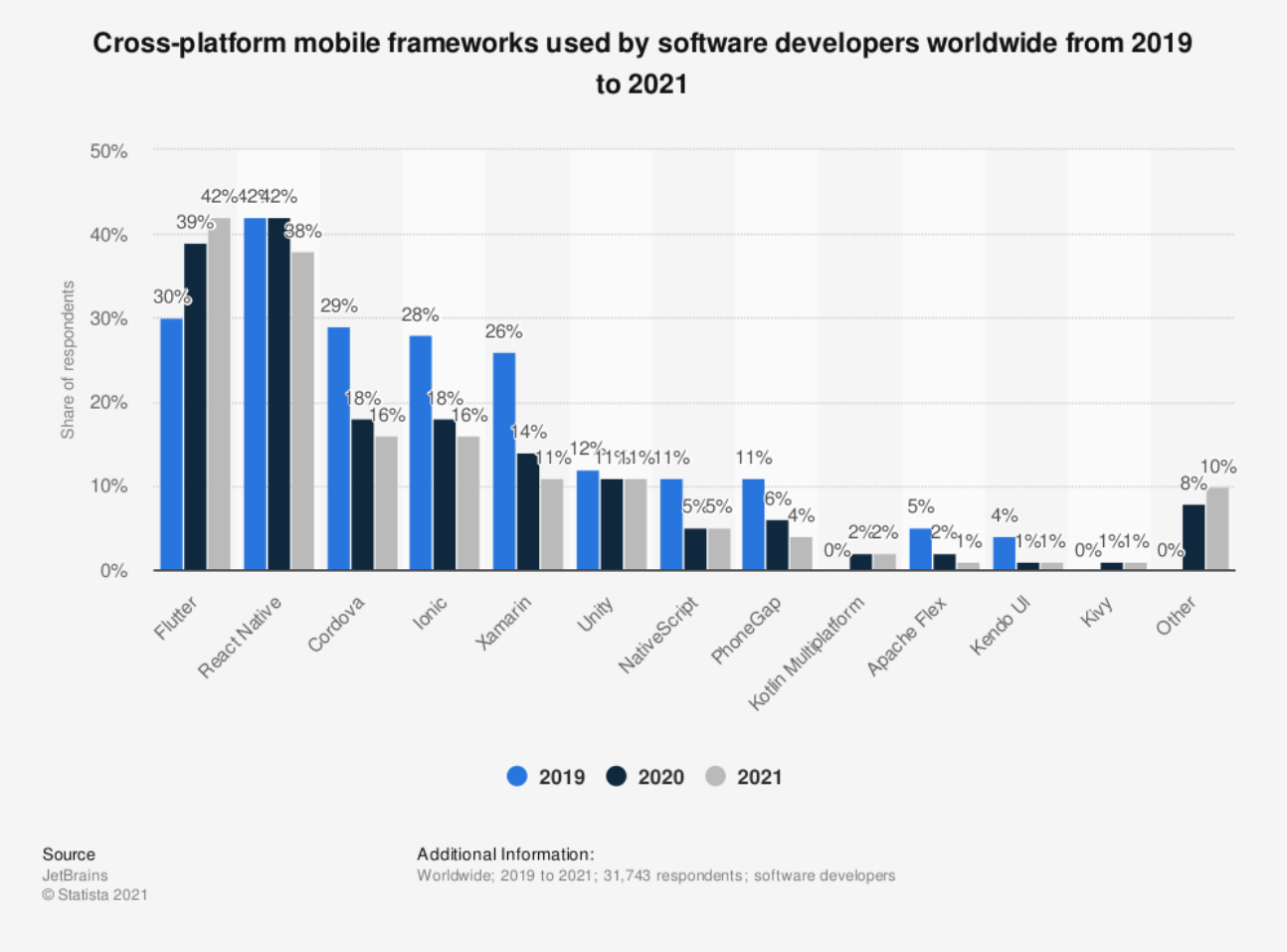
Trenutačno Windows podrška ima dosta problema. Većina problema je posljedica beta faze razvoja. Tri najvažnija problema su koja sam uočio su:

- Jednom kad se pritisne desni Shift cijelo vrijeme je uključen Caps Lock u aplikaciji koji se ne može isključiti. Potrebno je ponovno pokretanje aplikacije za popravak. Ovo je veliki problem kod prijavljivanja ako lozinka ili korisničko ime imaju veliko slovo unutar sebe.

- AltGr onemogući funkcioniranje tipkovnice unutar aplikacije. Ovo je velik problem jer znakovi poput @ se npr. kod lokaliziranih tipkovnica poput hrvatske često koriste, te zbog navedenog problema jednostavno može doći do isključenja aplikacije.
- Veliko opterećenje procesora.

6 ANALIZA FLUTTERA I KONKURENCIJE

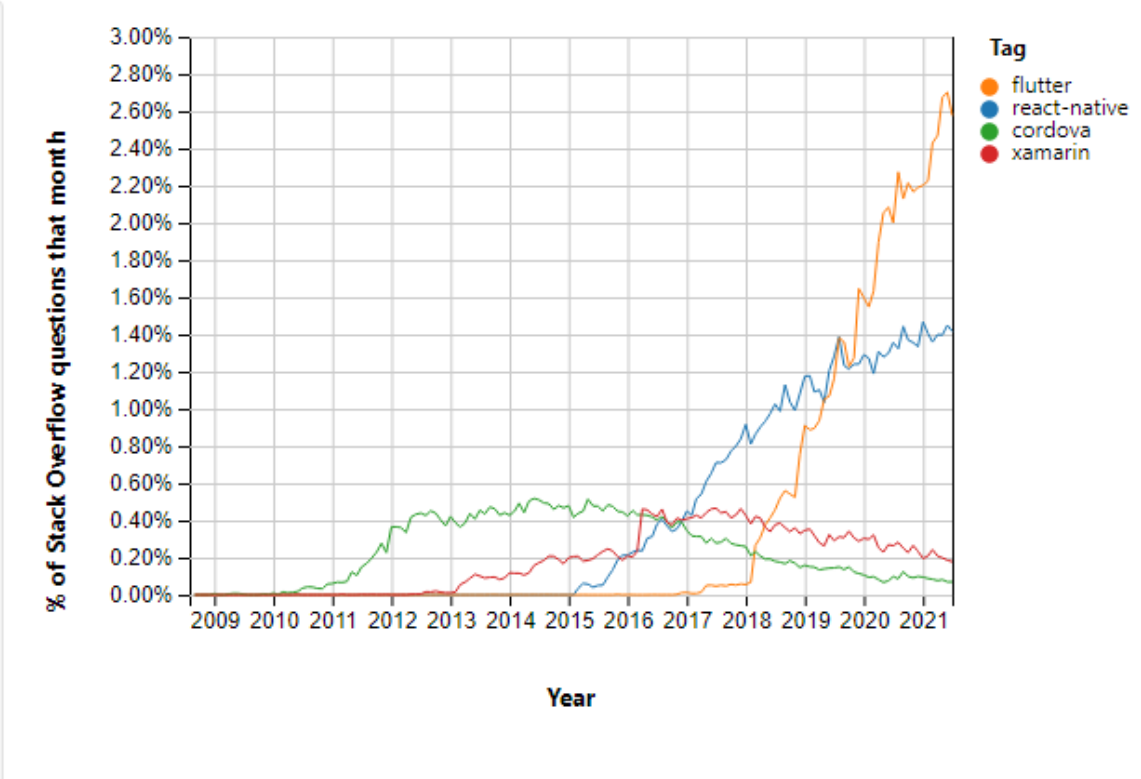
Flutter je najbrže rastuća platforma za izradu mobilnih aplikacija za više platformi. Na slici 51. prikazan je graf najpopularnijih mobilnih okvira za razvoj na više platformi u razdoblju od 2019 do 2021.



Slika 51. Graf najpopularnijih mobilnih okvira za više platformi u razdoblju od 2019. do 2021. [13]

Iz grafa je vidljivo da je *Flutter* u zadnje tri godine jedina platforma koja je rasla i dobila dominantan položaj na tržištu. Slični rezultati se mogu vidjeti iz grafa na slici 52. koji prikazuje postavljenja pitanja na stranici Stack Overflow koje se prati preko tagova.

Glavni konkurent *Flutteru* je *React Native* te će kroz daljnji dio ovog poglavlja biti uspoređivan s njime.



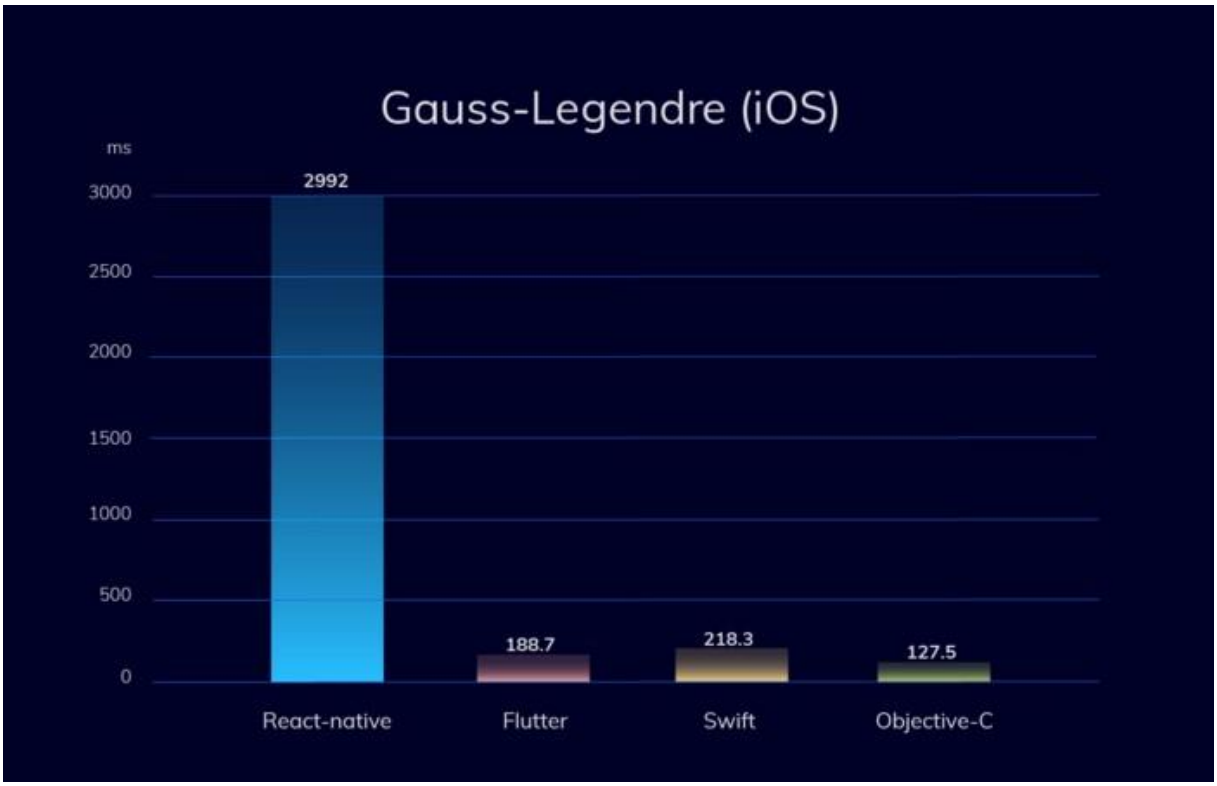
Slika 52. Graf postavljenih pitanja za tehnologije [14]

6.1 Mobilne performanse

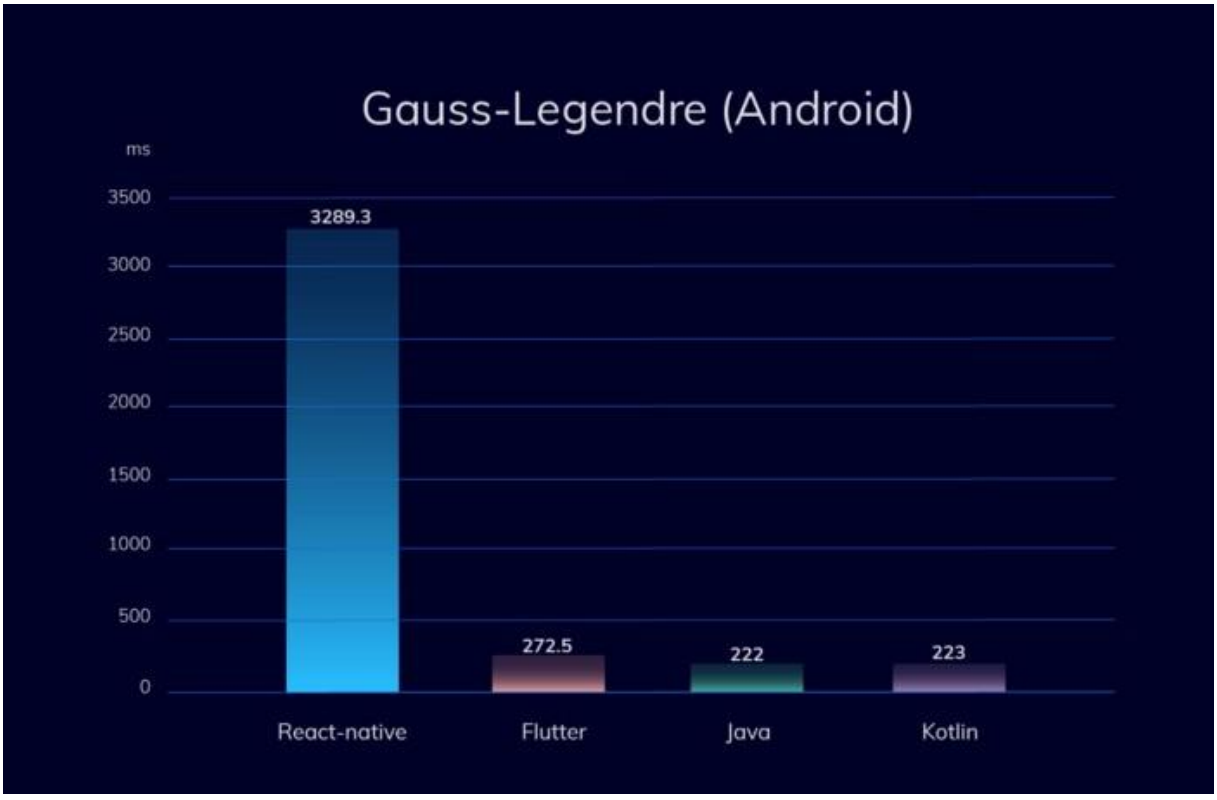
U današnjem svijetu najpopularnije rješenja za izradu mobilnih aplikaciju su pristupi na više platformi poput *React Native* i *Fluttera*. Kod tehničkih rješenja koje zahtijevaju najbolje performanse poput video igrica koristit će se izvorne tehnologije na platformama iOS-a i Androida. Najbitnije performanse su:

- Interakcije uređaja s API-jem,
- Brzina prikazivanja,
- Brzina izvršavanja poslovne logike.

Na slici 53. prikazani su rezultati procesno intenzivanog testa za iOS, a na slici 54. procesno intenzivanog testa za Android.



Slika 53. Procesno intenzivan test za iOS [15]



Slika 54. Procesno intenzivan test za Android [15]

U oba slučajeva korišten je Gauss-Legendre algoritam koji je izračunava znamenke π. Vrlo je efikasan u toj izvedbi jer brzo konvergira pa sa samo 25 iteracija dolazi do 45 milijuna točnih znamenki π.

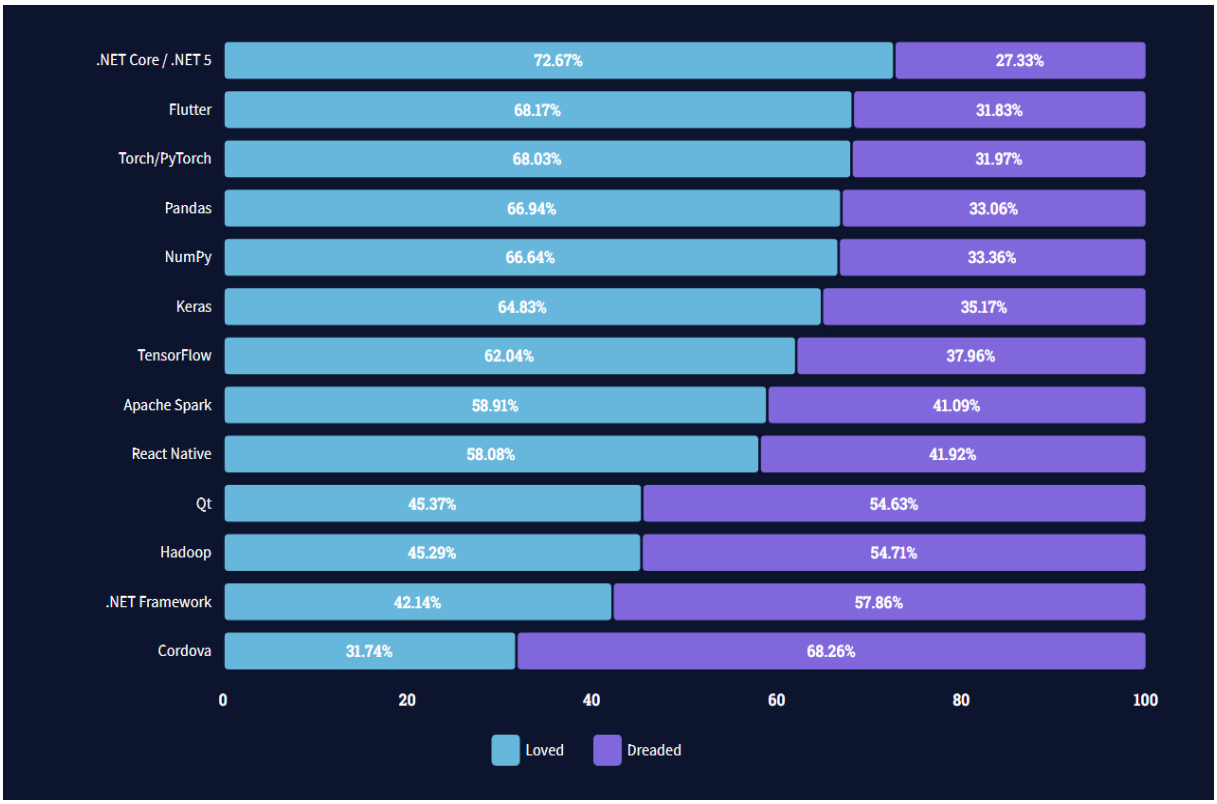
Svi su testovi izvedeni nekoliko puta i uzet je prosječni rezultat. Iz grafova možemo zaključiti da je brzina izvođenja *Fluttera* znatno bolja od *React Nativa* i konkurentna s izvornim tehnologijama. Razlog za to je što *React Native* koristi JS-BRIDGE koji prvo prevodi JavaScript programski kod prije prevođenja u izvršni, a isto radi i s korisničkim sučeljima dok *Flutter* nema toga nego iz izvornog programskog koda *Darta* automatski kompajlira.

6.2 Web performanse

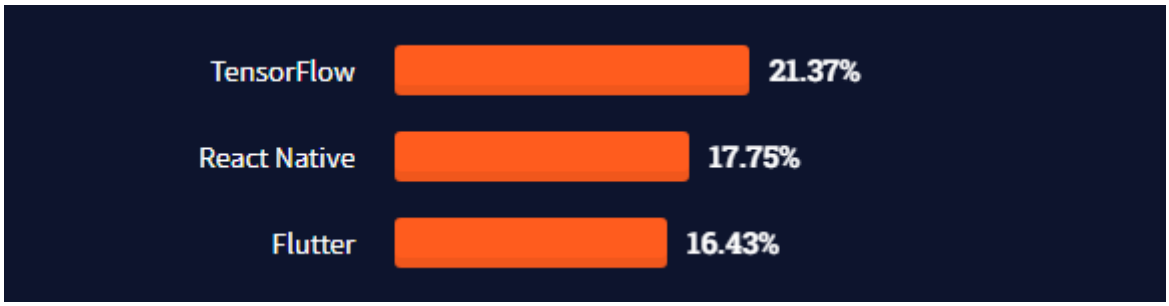
Web performanse će biti slične jer *Flutter* i *React Native* koriste JS, HTML i CSS. Velika razlika nastaje u tome što *React Native* ima pristup većem broju biblioteka i samo programiranje se radi u JavaScriptu što omogućuje veću kontrolu nad performansama. Brzina razvoja je brža kod *React Nativa* jer se programerski kod direktno izvodi iz JavaScripte dok se *Dart* prvo mora prevesti u JavaScript pa tek onda izvoditi.

6.3 Developer utisak i želja za učenjem tehnologija

Svake godine Stack Overflow skuplja podatke o mišljenjima developera o tehnologijama koje koriste. Na slici 55. prikazan je graf utiska tehnologije a na slici 56. graf želje za učenjem tehnologije iz istraživanja provedenog 2021. godine.



Slika 55. Graf utiska tehnologije okvira za razvoj aplikacija koje se nekoriste na webu [16]

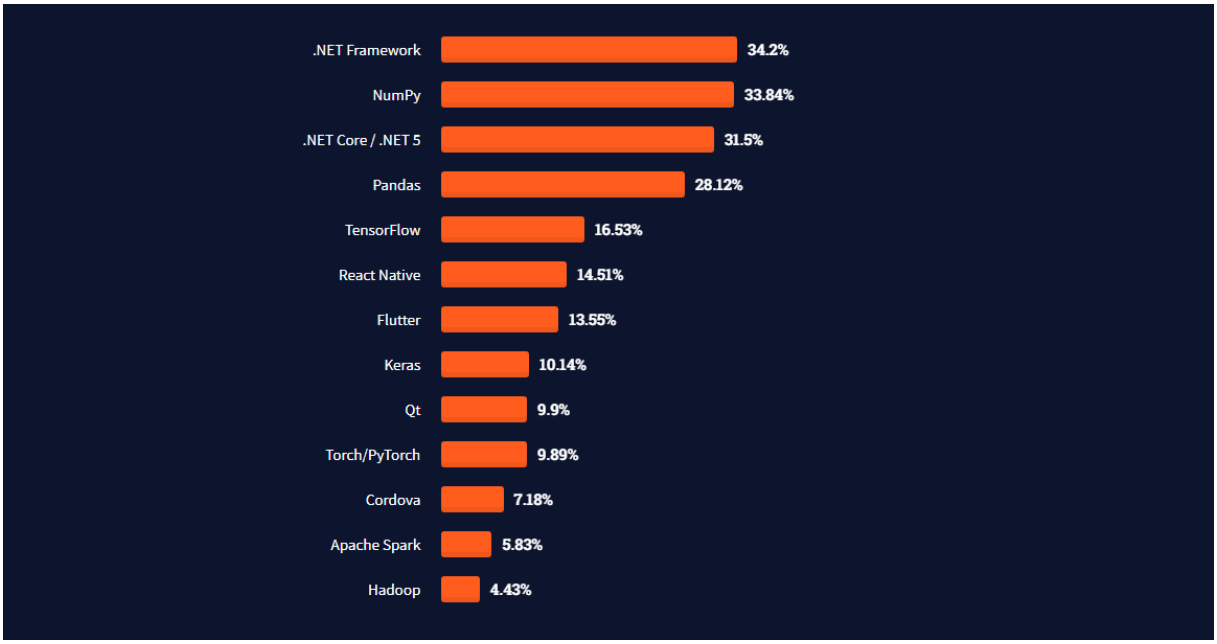


Slika 56. Graf željom za učenjem okvira za razvoj aplikacija koje se nekoriste na webu [16]

Trenutačno je anketu popunilo preko 58,000 programera. Iz grafa utiska tehnologije je vidljivo da 68% programera koji koriste *Flutter* vole tu tehnologiju dok je za *React Native* taj postotak nešto manji i iznosi 58%. Razlika od deset posto nije prevelika da bi se mogle povući značajne teze ali je prva u nizu činjenica koje ukazuju na percepciju zajednice. Iz grafa želje za učenjem vidimo da *Flutter* sa 16.43% i *React Native* 17,75% su jedni od najtraženijih tehnologija koje bi programeri htjeli naučiti.

6.4 Popularnost

Prema uvidima u anketama StackOverflowa koje je ispunilo preko 59,000 programera, *React Native* se nalazi na 6. mjestu najčešće korištenih okvira ili alata koji nisu samo web okviri s postotkom korištenja od 14,51%. *Flutter* se nalazi na 7. mjestu s 13,55%. Te dvije platforme su dvije najpopularnije *cross-platform* tehnologije za razvoj mobilnih aplikacija. Na slici 57. vidljiv je Graf popularnosti okvira za razvoj aplikacija koje se ne koriste na webu.

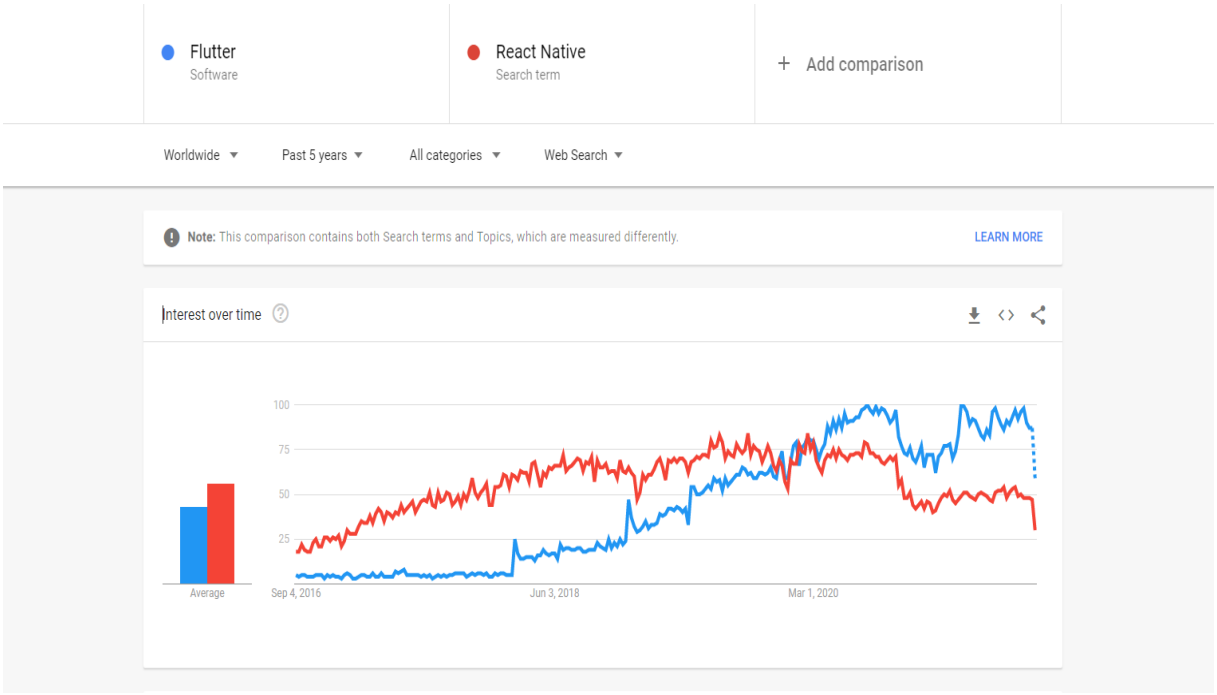


Slika 57. Graf popularnosti okvira za razvoj aplikacija koje se nekoriste na webu [16]

6.5 Tržišni trendovi

U današnjem modernom svijetu nove tehnologije se brzo razvijaju, a stanje na tržištu se radikalno mijenja iz godine u godinu. Vrijedi pravilo da tko se prvi pojavi na tržištu uglavnom

dobije veći udio u tom tržištu. *React Native* je pokrenut prije *Fluttera* i to je glavni razlog zašto on dominira tržištem istovremenog razvoja na više platformi. Međutim *Flutter* je na pravom putu da ga prestigne. Ako se trendovi nastave, samo je pitanje vremena kada će *Flutter* preuzeti vodstvo na tržištu. Na donjem grafu Google trendova vidljivo je da *Flutter* od ožujka 2020. godine postaje više tražen na Google tražilici što ukazuje na veću zainteresiranost developera za tehnologiju. Na slici 58. prikazan je graf Google trendova za *Flutter* i *React Native*.



Slika 58. Graf Google trendova između *Fluttera* i *React Nativa*

6.6 Krivulja učenja i produktivnost

Flutter je relativno nov programerski jezik, stoga iako nije težak za naučiti, zahtjeva veći napor za učenje dok *React Native* koji je temeljen na JavaScriptu zahtjeva manju krivulju učenja jer su programeri često prethodno upoznati s JavaScriptom.

Produktivnost se može podijeliti u tri kategorije:

- Tehnologije koje pomažu pri programiranju,
- Instalacija i konfiguracija,
- Struktura programskog koda.

Glavna tehnologija koja pomaže pri programiranje je *Hot Reloading* funkcionalnost koja omogućuje programeru uvid u promjene napravljene u pozadini ili na sučelju s jednim pritiskom tipke. Pritiskom tipke cijeli projekt se neće ponovno prevoditi nego će se samo prikazati novi podaci ili promjene na sučelju što će smanjiti vrijeme programiranja jer nema vrijeme stalnog kompiliranja koda. *Flutter* i *React Native* podržavaju funkcionalnost *Hot Relodinga*.

Flutter je napredni razvojni softver koji programeri mogu jednostavno instalirati i pokrenuti koristeći korak po korak vodič u dokumentaciji. *React Native* je isto lagano instalirati

ali on pri instalaciji stvara paket umjesto kompletnog razvojnog softvera pa je način upotrebe različit.

Dart je programski jezik koji koristi *Flutter, a* prilagođen je stvaranju izgleda korisničkog sučelja aplikacije i omogućuje programerima stvaranje platformi, struktura i widgeta koji se jednostavno mogu koristiti kroz cijeli projekt. S druge strane *React Native* je sličan JavaScriptu koji programerima omogućuje jednostavno razdvajane izvedbenog koda u stilove različitih klasa prema zahtjevima razvoja aplikacije [17].

6.7 Vrijeme izgradnje aplikacije i pronalazak programera

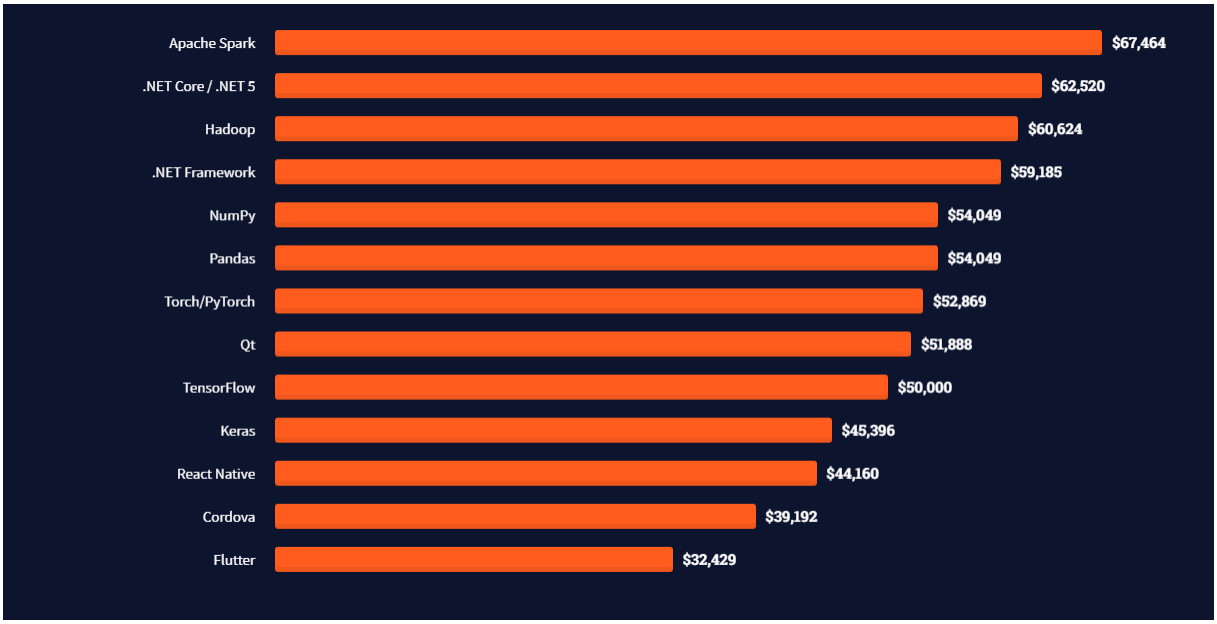
Vrijeme izgradnje aplikacije ili takozvano vrijeme potrebno do plasiranja aplikacije na tržište i pronalazak programera ovisi o:

- Zajednici koja podržava tehnologiju i pomaže kod rješavanja netrivijalnih razvojnih pitanja,
- Tome za koju tehnologiju je lakše naći programere.

U ovome poglavlju smo prošli kroz Google trendove i postavljanja pitanja na Stack Overflowu i iz tih podataka može se vidjeti da je zajednica koja podržava *Flutter* u zadnjih nekoliko godina narasla i prerasla zajednicu koja podržava *React Native.*

Pronalazak programera ovisi o tome na kojem se tržištu nalazimo. Uglavnom vrijedi pravilo da u svakom dijelu svijeta trenutačno ima manje *Flutter* programera nego *React Native* programera. Ako je cilj zaposliti veći tim programera, lakše je naći *React Native* programere. Samostalno kretanje u razvoj aplikacije je moguć s obje platforme, a izbor će ovisiti o osobnim sklonostima.

Na slici 59. prikazan je graf srednji godišnjih plaća u USD. U anketi je sudjelovalo preko 26,000 programera i jasno je vidljivo da su *Flutter* programeri najjeftiniji s prosjekom od 32,429$ dok su *React Native* programeri skuplji sa 44,160$ godišnjom plaćom.



Slika 59. Graf srednjih godišnjih plaća u USD [16]

Vrlo je teško doći do konkretnih zaključaka iz ovoga grafa zbog toga što je zajednica *Stack Overflowa* globalna. Ta činjenica, kao i to što životni standard nije isti svugdje na svijetu, dovodi do zaključka da ne možemo definitivno tvrditi da će programeri *Fluttera* uvijek i u svakoj situaciji biti manje plaćeni od kolega *React Nativa.* Ipak možemo tvrditi da će globalno tim programera *React Nativa* biti oko 10% 20% bolje plaćen od njih.

6.8 Primjeri aplikacija

S gledišta usvajanja *React Native* trenutačno vodi. Iako je *Flutter* pokazao nagao rast u relativnom kratkom vremenu još dosta zaostaje za *React Nativom* kod gledišta broja aplikacija koje se nalaze na tržištu. Jasniju sliku nam mogu dati primjeri aplikacija koje koriste opisane tehnologije. Neke od najznačajnijih aplikacija razvijene u *Flutteru* su:

- BMW aplikacija,
- Nubank,
- Tencent,
- Google Ads,
- Google Assistant,
- Square,
- eBay,
- The New York Times.

Neke od najznačajnijih *React Native* aplikacija su:

- Facebook,
- Instagram,
- Coinbase,
- Shopify,
- Discord,
- Skype,
- Tesla,
- Pinterest,
- Walmart,
- Uber Eats.

Iz prethodne dvije liste vidljivo je da *React Native* koriste više raznolikih firmi i da ima veći prodor na tržište.

6.9 Sažetak analize

Iz provedene analize vidljivi su trendovi koji vode k tome da će udio *Fluttera* na tržištu razvoja višeplatformskih aplikacije rasti. Ostaje otvoreno pitanje na koliko će se taj udio povećati. Udio na tržištu ovisit će o kontinuiranom ažuriranju verzija *Flutter* projekta. Kroz analizu performansi na mobilnim platformama jasno je da *Flutter* najefikasnije rješenje i da zaostaje za izvornim platformama za samo 10% -20% što je ogromno postignuće jer se preko 90% programskog koda u svim slučajevima koristi za razvoj Androida i iOS aplikacija.

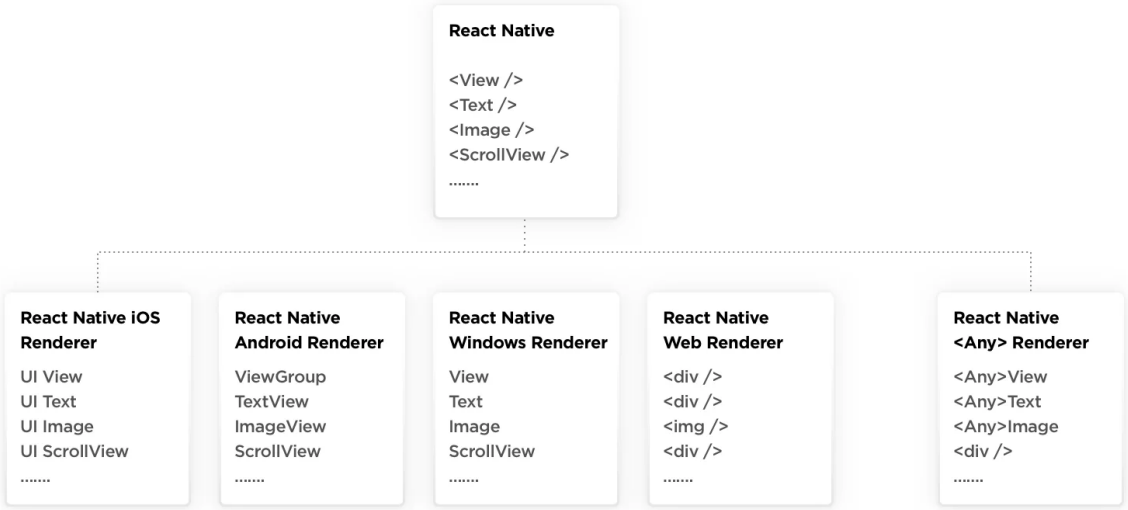
Trenutačno performanse na webu su slične kao i kod *React Nativa* s time da on ima prednost jer se programira u tehnologijama koje su primarno napravljene za izradu web stranica i aplikacija. Desktop *Flutter* aplikacije su još u beta fazi izgradnje pa o njima nema relevantnih podataka za analizu ali se već mogu vidjeti koristi responzivnog dizajna, jednostavne izrade korisničkog sučelja i brzog rada pod velikim opterećenjima. Sustavi s ogromnom količinom podataka mogli bi postati brži i efikasniji. Zajednica *Flutter* programera raste i jedna je od najzadovoljnijih i najaktivnijih zajednica u davanju podrške.

7 PREDNOSTI I NEDOSTACI FLUTTERA

Svako programsko okruženje ili programski jezik imaju svoje prednosti i nedostatke, ali samo kroz njih možemo bolje razumjeti u kojim situacijama su pogodni za korištenje, a u kojima ih treba izbjegavati.

7.1 Isto korisničko sučelje i poslovna logika

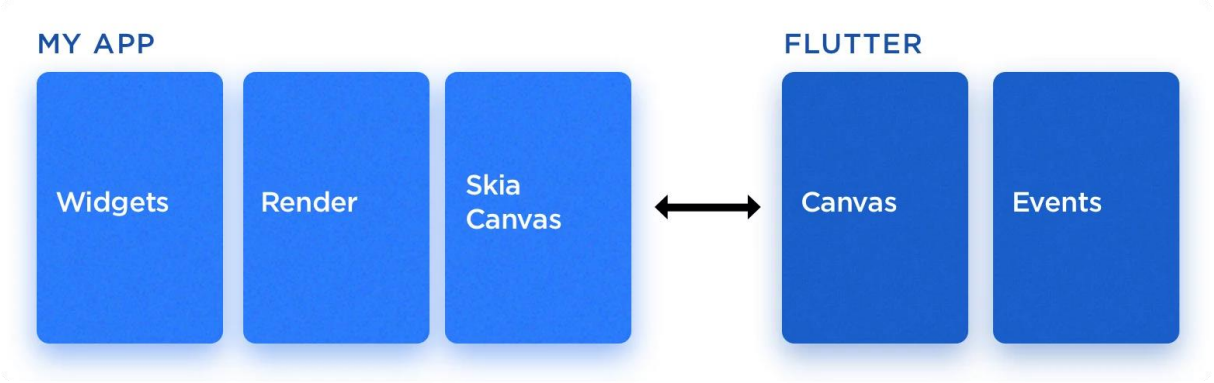
Pretpostavka kod razvoja višeplatformskih aplikacija je da okvir programskog okruženja omogućava dijeljenje izvornog koda između različitih platformi. Jedina platforma od promatranih koja dijeli izvorni kod programskog sučelja i samo programsko sučelje je *Flutter*. Da bismo to demonstrirali na slici 60. je prikazan okvir prikazivanja korisničkog sučelja *React Nativa*.



Slika 60. Okvir prikazivanja korisničkog sučelja *React Nativa* [18]

Ovakav postupak generiranja korisničkog sučelja kod izrade aplikacije pojednostavljuje proces. Oslanjanje na komponente specifične za platformu za generiranje korisničkog sučelja stvara potrebu za slojem mapiranja svih *widgeta* platforme i njihovu sinkronizaciju da bi se uspostavio kontinuirani izgled na svim platformama.

Za razliku od drugih platformi *Flutteru* ne treba nijedna komponenta korisničkog sučelja specifična za platformu da bi prikazao svoje korisničko sučelje. Jedina stvar koja mu je potrebna da bi iscrtao svoje korisničko sučelje je platno (engl. *canvas*). Na slici 61. vidljiv je proces stvaranja korisničkog sučelja kod *Flutter* aplikacije.



Slika 61. Stvaranje korisničkog sučelja [18]

Flutter koristi *widgete* i svoje unutrašnje platno za prikaz korisničkog sučelja tijekom izrade aplikacije. To funkcionira na način da *Flutter* gradi stablo objekata po prioritetu (na vrhu temeljni objekti poput My App, pri dnu *widgeti* poput stupaca redaka, ikona i sl.) za iscrtavanje. Nakon toga pokreće se *Flutter* engine koji inicijalizira korisničko sučelje te komunicira s ugrađivačem koji na temelju odabrane platforme prevodi programski kod u poželjan programski jezik [18].

7.2 Napravljeni i prilagođeni, widgeti i animacije

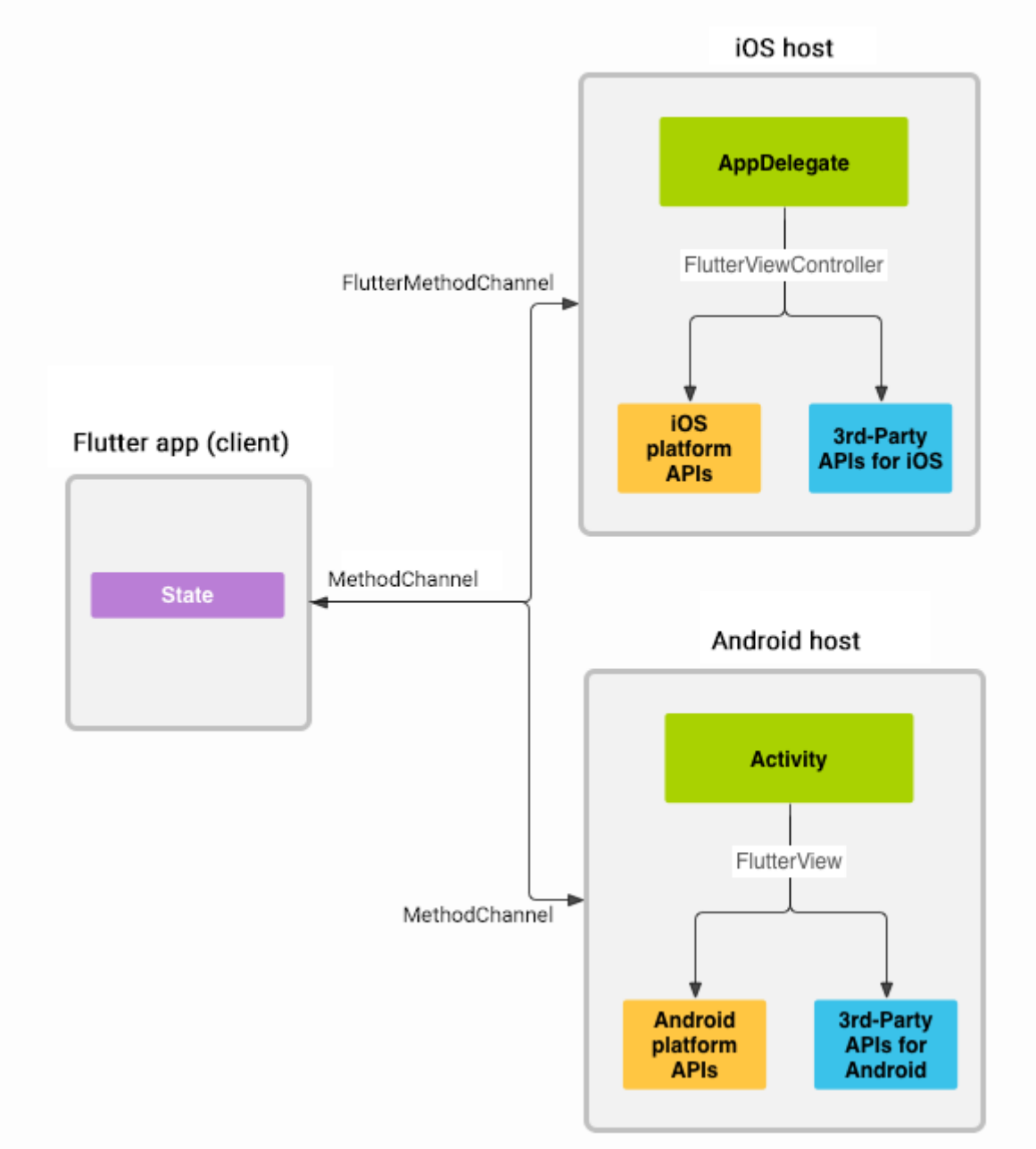
Svaki objekt unutar *Fluttera* je widget sastavljen od tipki, fontova, punjenja i izgleda. *Flutter* ima veliku količinu widgeta koje slijede *Android* materijalni dizajn i Appleov Cupertino dizajn. Svaki dio korisničkog sučelja može se prilagoditi potrebama specifične aplikacije. Izrada prilagođenih *widgeta* je jednostavna i brza. Svaki *widget* unutar sebe ima opcije za oblik, boju, sjene, transformacije, animacije te sve ostale dijelove potrebne za moderno korisničko sučelje. S mogućnosti pravljenja prilagođenih *widgeta* smanjuje se vrijeme programiranja različitih elemenata korisničkog sučelja koji su potrebni kod drugih okvira koji nude razvoj aplikacija za više platformi korištenjem iste baze koda.

7.3 Jednostavna implementacija logike specifične za platformu

Osim korisničkog sučelja veliki broj mobilnih aplikacija oslanja se na tehnologije na razini operativnog sustava da bi funkcionirale. Neke od takvih tehnologija su:

- GPS koordinate,
- *Bluetooth* komunikacija,
- Podaci prikupljeni senzorima,
- Rukovanje dozvolama,
- Provjera identiteta,
- Upravljanje kamerom.

Za svaku bitnu tehnologiju postoji biblioteka koja je jednostavna za korištenje koju direktno podržava Google koja funkcionira i na Androidu i iOS-u. Ako u nekom slučaju ne postoji biblioteka za traženu tehnologiju ili dolazi do potrebe za pisanjem koda specifičnog za određenu platformu to se može ostvariti korištenjem platformnih kanala. Na slici 62. prikazana je arhitektura platformnih kanala



Slika 62. Arhitektura platformnih kanala [19]

Poruke se šalju između korisničkog sučelja s klijentske strane prema platformi korištenjem Method kanala za Android i FlutterMethod kanala za iOS. Poruke i odazivi se šalju asinkrono da bi aplikacija ostala responzivna. Standardni platformni kanali omogućuju slanje svih standardnih tipova podataka za tu izvornu platformu poput logičkih vrijednosti, brojeva, nizova, lista, skupova itd.. Serijalizacija i deserijalizacija podataka u porukama i iz njih događa se automatski kad šaljemo ili primamo podatke [19].

7.4 *Dart* prednosti

Dart je jednostavan i efektivan programerski jezik koji bi svaki Java programer trebao moći lagano naučiti. Programerski jezik nije ograničen platformom. Glavne prednosti *Darta* su:

- **AOT i JIT tipovi prevođenja** – AOT je stil prevođenja gdje se program prevodi tijekom pisanja koda, a prije izvođenja što znači da su performanse bolje ali je vrijeme razvijanja duže jer se stalno kod svake promjene treba ponovno prevesti. JIT način prevođenja koda je da prevodi neposredno prije izvođenja ubrzava vrijeme izrade aplikacije ali pošto prevoditelj radi analiziranje prije izvršavanja koda, performanse su lošije. *Flutter* za vrijeme razvijanja aplikacije koristi JIT način prevoženja a za izdavanje aplikacije koristi AOT.
- **Nema potrebe za XML datotekama**- Kod *Fluttera* izgled i objekti se nalaze na istom mjestu dok se kod Androida izgled radi u XML datotekama s pogledima koji se referenciraju u Java kodu. Pošto se sve nalazi na istom mjestu lakše je uočiti pogreške i ubrzava se vrijeme razvoja.
- **Bolje performanse** – *Dart* ne koristi nikakve mostove spajanja na platforme nego direktno crta svoje korisničko sučelje i poslovnu logiku na platno platforme [19].

7.5 Googleovo jamstvo dugoročne podrške

Google nema najbolju povijest pružanja podrške svojim proizvodima. Postoje preko dvijesto stotine aplikacija, hardvera i usluga kojima je ukinuta podrška ili su maknuti iz proizvodnje. Neki od ukinutih su: Google Glasses, Google TV, AngularJS, Google Now itd.. *Flutter* je u posebnoj poziciji jer je projekt otvorenog izvora s podrškom Googlea. Google je počeo prebacivati sve svoje mobilne aplikacije na *Flutter* i možemo pretpostaviti da ako prebacuju sve svoje usluge na tu tehnologiju, da će je onda i podržavati u daleku budućnost.

7.6 Portabilnost

Zbog *Dartovih* mogućnosti prenosivosti i kompilacijskih mogućnosti, ista kodna baza može se primijeniti na pet glavnih operativnih sustava: iOS, Android, Linux, MacOS i Windows. Google radi na proširenju kako bi mogli pokriti automobile, pametne televizore, pametne kućanske aparate sl..

7.7 Internacionalnost i pristupačnost

Google uz pomoć osnovnih biblioteka pruža mogućnosti da se izrađena aplikacija učini dostupnom širem krugu korisnika. Obično ako aplikaciju želimo izvodi na različitim jezicima i da se koristi u različitim regijama, potrebno je lokalizirati programski kod i taj proces se zove internacionalizacija. Uz pomoć intl biblioteke taj proces je pojednostavljen i trenutačno podržava 78 jezika, valuta, mjerne jedinice i datuma.

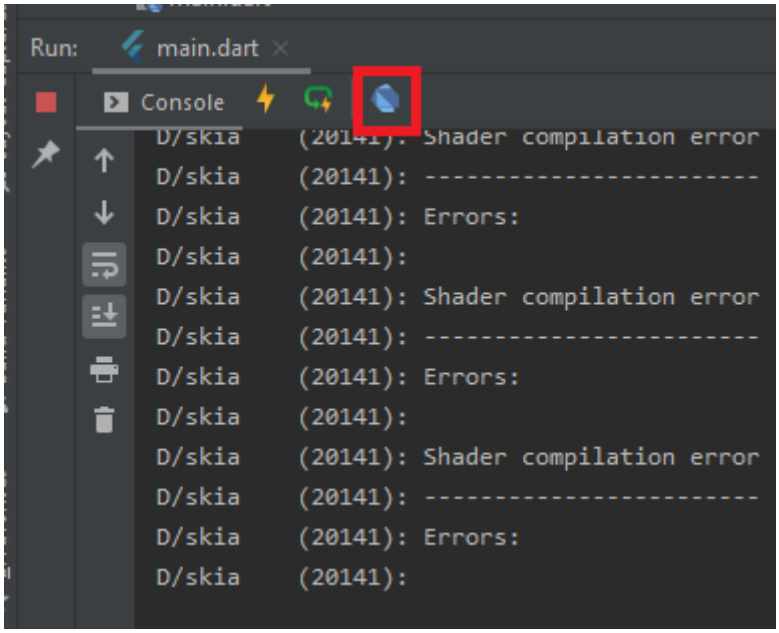
Flutter osigurava web pristupačnost i podržava ove tri komponente:

- **Veliki fontovi** – prilagođava veličine fonta onima koje korisnik navede u postavkama operacijskog sustava,
- **Čitači zaslona** – pruža govorne povratne informacije o elementima korisničkog sučelja,
- **Dovoljan kontrast** – tekst se lakše čita.

Cijeli proces pristupačnosti je automatiziran ali bi svakako trebalo testirati svaki ekran dali se uklapa veliki font na aplikaciju [18].

7.8 DevTools

DevTools je paket alata za provjeru performansi i otklanjanje pogrešaka aplikacije u *Flutteru*. Da bi se pristupilo alatu prvo se treba instalirati izvođenjem naredbenog retka „flutter pub global activate devtools“. Nakon aktivacije svako pristupanje se vrši ili direktno iz razvojnog okruženja ili uz izvođenje naredbe „flutter pub global run devtools“. Na slici 63. prikaz u crvenom kvadratu odnosi se na pokretanja DevToolsa unutar razvojnog okruženja AndroidStudio.



Slika 63. Pokretanje DevToolsa preko AndroidStudija

Stvari koje se mogu koristiti s DevTools:

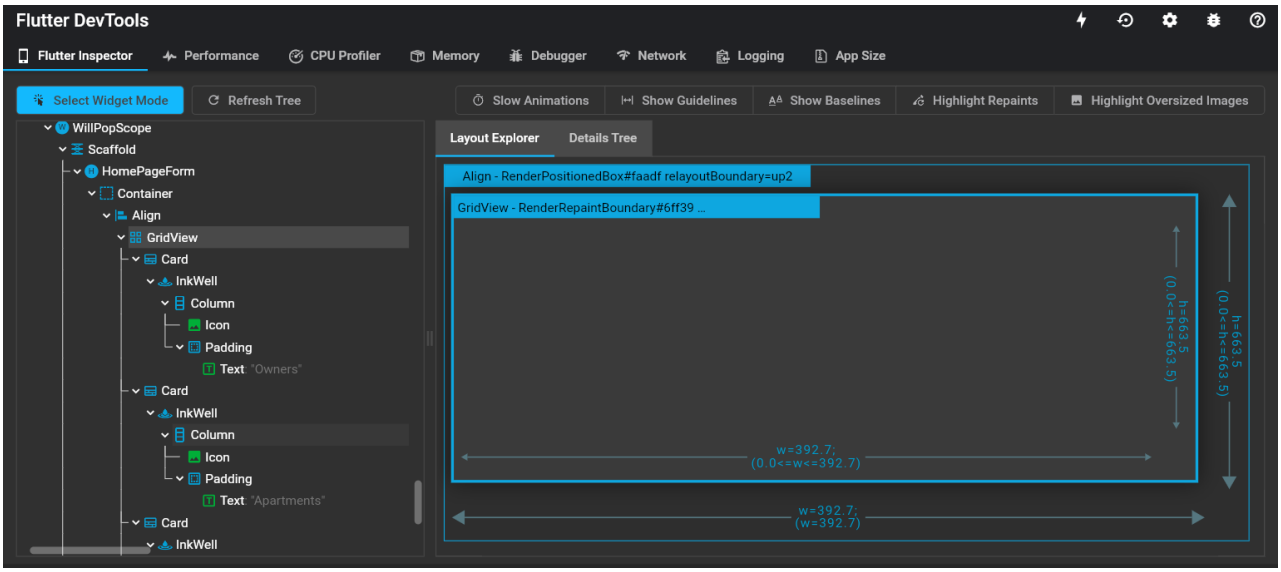
- Provjera izgleda korisničkog sučelja i stanja aplikacije,
- Dijagnosticiranje problema s performansama,
- Profiliranje rada aplikacije,
- Mrežno profiliranje rada za aplikacije,
- Otklanjanje pogrešaka na razini izvornog koda aplikacije,
- Otklanjanje pogrešaka u memoriji aplikacije,
- Analiziranje programskog koda i veličine aplikacije,
- Pregled zapisnika događaja u aplikaciji.

7.8.1 *Flutter* inspektor

Inspektor widgeta *Fluttera* je alat za vizualizaciju i istraživanje *Flutter widget* stabala. Pomaže vizualizirati i istražiti stabla *widgeta Fluttera*, a može se koristiti za sljedeće:

- Razumijevanje postojećeg izgleda,
- Dijagnosticiranje problema u izgledu.

Na slici 64. vidljivo je cijelo korisničko sučelje *Flutter* inspektora.



Slika 64. Korisničko sučelje *Flutter* inspektora

Značajke dostupne na alatnoj traci inspektora:

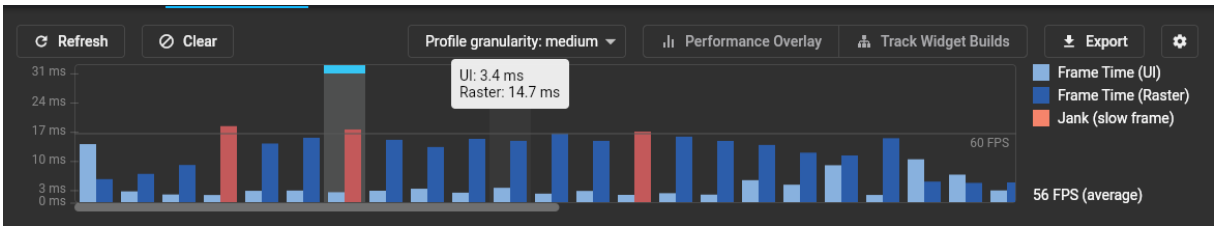
- **Odaberi *widget*** - na slici 64. je označen, a u upaljenoj aplikaciji na emulatoru mobitela je označen GridView početnog ekrana. Layout Explorer služi za provjeru visine, duljine, pregleda i stanja odabranog *widgeta*. S njime se mogu dodavati poravnanja, promjene pozicije i slične dizajnerske promjene unutar koda s kojim se automatski promjene odvijaju na emulatoru.
- **Osvježi stablo** - ponovno učitava trenutačne podatke o *widgetima*.
- **Spore animacije** – animacije su sporije 5 puta da bi se lakše podesile potrebama aplikacije.
- **Prikaz smjernica** – smjernice su strelice koje pokazuju okvire *widgeta* i smjerove u kojima se ti *widgeti* mogu kretati. Primjer je obična lista, za elemente unutar liste strjelice će pokazivati prem dole jer se kroz listu može pomicati.
- **Prikaz polazišta** – prikazuje linije uz pomoć kojih se može vidjeti jeli tekst unutar aplikacije poravnat.
- **Istaknuti više puta iscrtana područja**– prikazuje mjesta na ekranu koja se crtaju više puta pa se ti elementi lakše mogu popraviti.
- **Istaknuti prevelike slike** – prikazuje slike u obrnute ako zauzimaju previše memorije.

7.8.2 Pregled performansi

Pregled performansi nudi vrijeme i informacije o izvedbi aktivnosti u aplikaciji. Sastoji se od dva dijela:

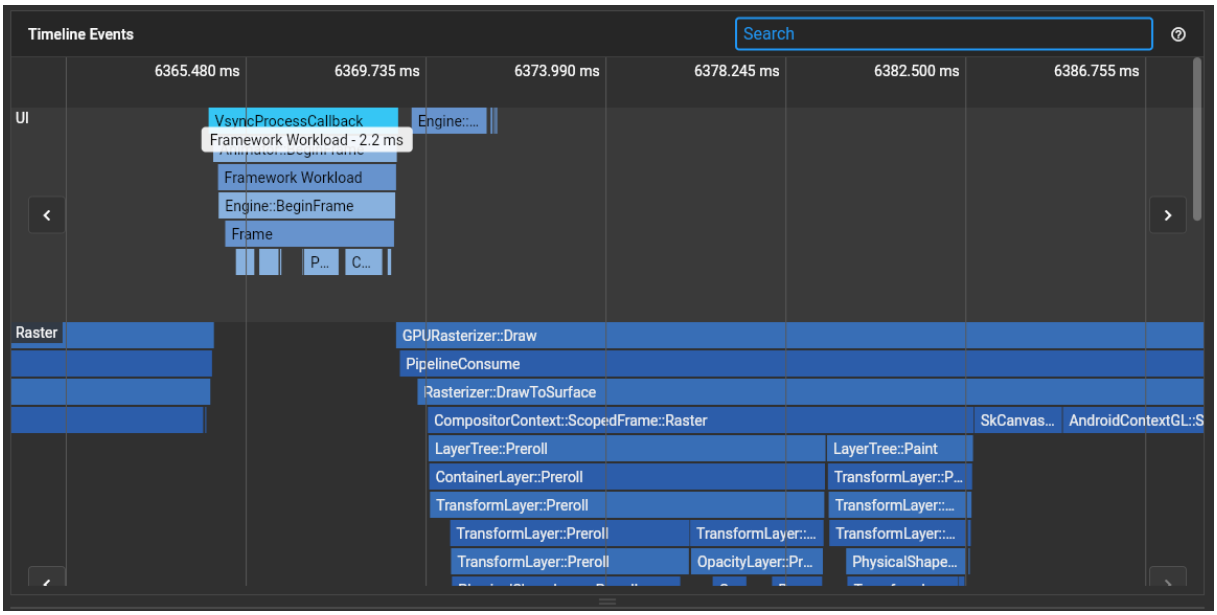
- *Flutter* graf frekvencija okvira,
- Grafikon događaja na vremenskoj traci.

Na slici 65. se vidi graf frekvencije okvira



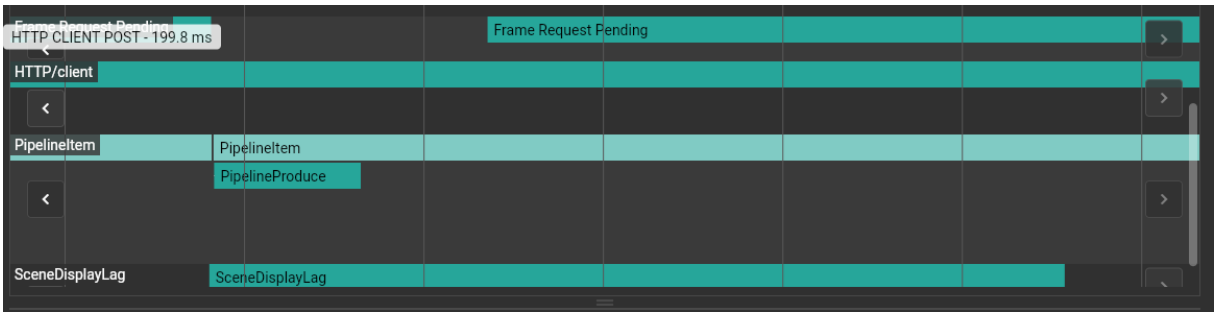
Slika 65. Graf frekvencije okvira

Frekvencija okvira je mjera koliki broj slika se prikazuje u jednoj sekundi. *Flutterov* cilj je pružanje frekvencijskog okvira od 60 slika po sekundi. Za svaku sličicu postoji vremenski okvir od 16 milisekundi u kojemu se treba iscrtati korisničko sučelje, odraditi dio animacije ili poslovne logike. Ako se zadani zadaci ne izvrše u vremenskom okviri performanse se smanjuju jer se zadaci prebacuju na sljedeći okvir. U aplikaciji se to očitava u obliku sporih interakcija, učitavanja i zaostajanja. Graf sadrži informacije o frekvencijama okvira u *Flutteru* tijekom odvijanja aplikacije. Svaki stupac predstavlja jednu frekvenciju okvira. Stupci su obojeni bojama, svijetlo plavi stupac predstavlja vrijeme radnje u korisničkom sučelju dok tamno plavi stupac predstavlja rad na grafičkom sučelju. *Flutter* ima dvije niti. Prva nit izvršava Dart kod koji je napisan u aplikaciji te cijeli *Flutter* okvir. Kada aplikacija prikaže i stvori ekran tada nit stvori stablo izgleda kojega pošalje drugoj niti. Grafička nit prima stablo izgleda i njega ispunjava poslovnom logikom i podacima. Crveni stupac predstavlja neoptimalnu radnju u jednoj frekvenciji kada je veća od 16 milisekunda za uređaje kojima je pregled 60 frekvencija okvira po sekundi. Pritiskom na bilo koji stupac dobijemo detalje o operacijama obavljenim za vrijeme te frekvencije. Na slici 66. vidljive su korisničke i grafičke operacije nad odabranim stupcem.



Slika 66. Pregled funkcija obavljenih u jednom kadru

UI nit izvršava *Dart* kod, to uključuje kod aplikacije kao i *Flutterov* okvir. Raster nit izvršava sve grafičke zadatka korištenje prikladnog *engina*. Na slici 67. prikazan je grafikon događaja na vremenskoj traci.

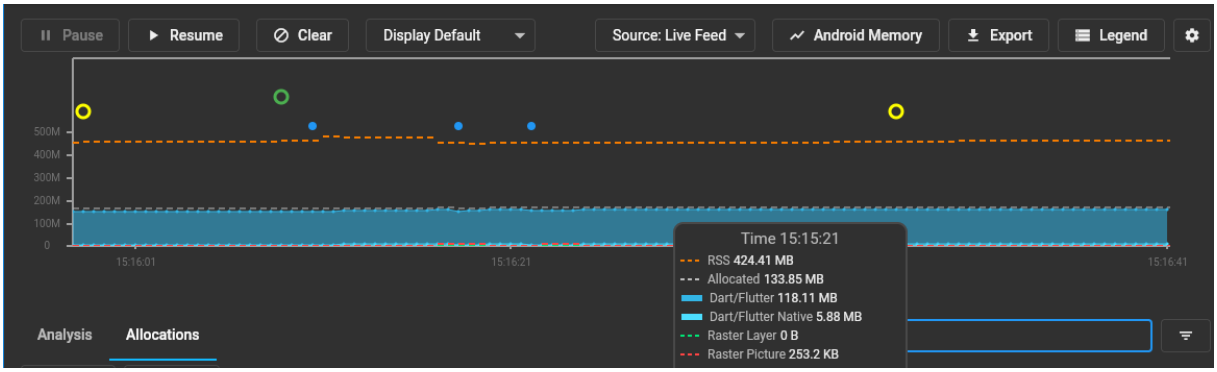


Slika 67. Grafikon događaja na vremenskoj traci

Gornji grafikon prikazuje sve događaje koji su povezani s aplikacijom. Grafikon se popunjava tako da *Flutter* prati sve događaje koji su povezani za izgradnju kadrova, ekrana i HTTP prometa. Na gornjem grafikonu vidljiv je HTTP promet koji je trajao 199,8 milisekundi što znači da se izvodio tijekom jedne petine frekvencijskog okvira.

7.8.3 Pregled radne memorije

DevTools omogućuje pregled memorije u bilo kojem trenutku rada aplikacije. Cilj je omogućiti programerima uvid u probleme u memoriji poput curenja memorije. Omogućene su opcije snimka pojedinog trenutka ili praćenje posebnih dijelova. Na slici 68. vidljivo je praćenje memorije.

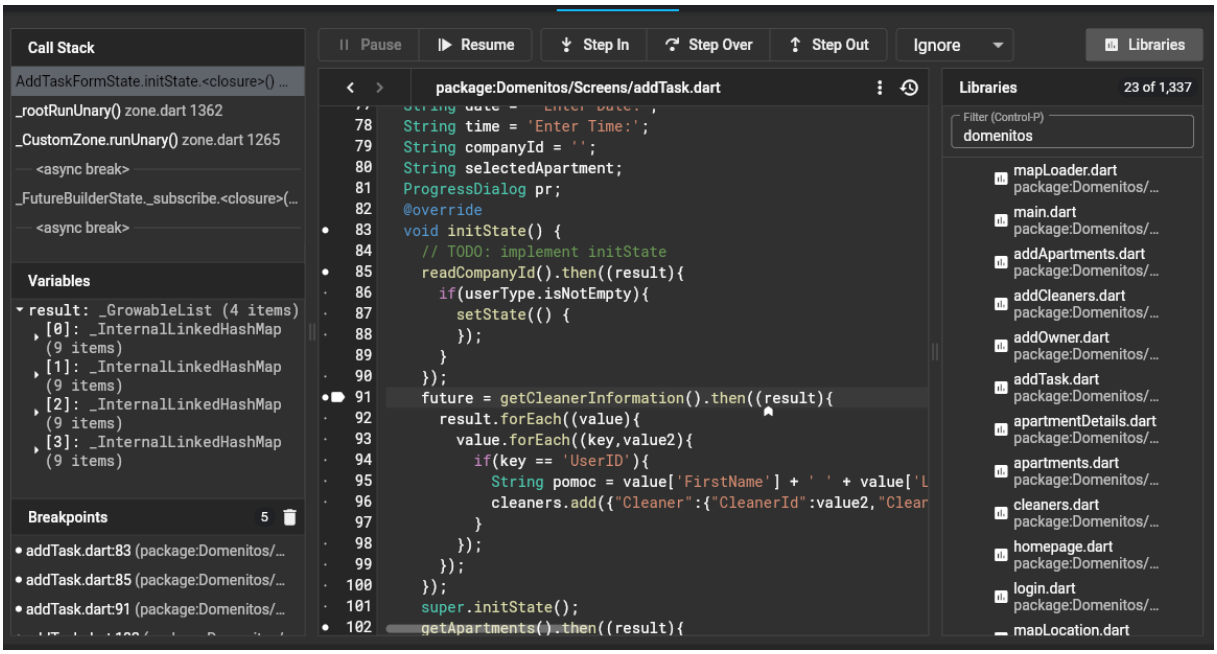


Slika 68. Praćenje memorije

Na gornjoj slici vidljivo je da trenutačna aplikacija koristi 133,85 MB memorije. Žuti kružići predstavljaju praćenja događanja unutar memorije dok zeleni kružići predstavlja snimak u tom zadanom trenutku. Plave točkice podrazumijevaju trenutke u kojima se aktivira sakupljač smeća i oslobađa nekorištene resurse aplikacije. Primjer aktiviranja sakupljača je prijelaz s jednog ekrana na drugi.

7.8.4 Ispravljač pogrešaka

U ispravljanje pogrešaka su uključene prekidne točke, koraci, i pregledi varijabli. Na slici 69. Vidljiv je ispravljač pogrešaka.

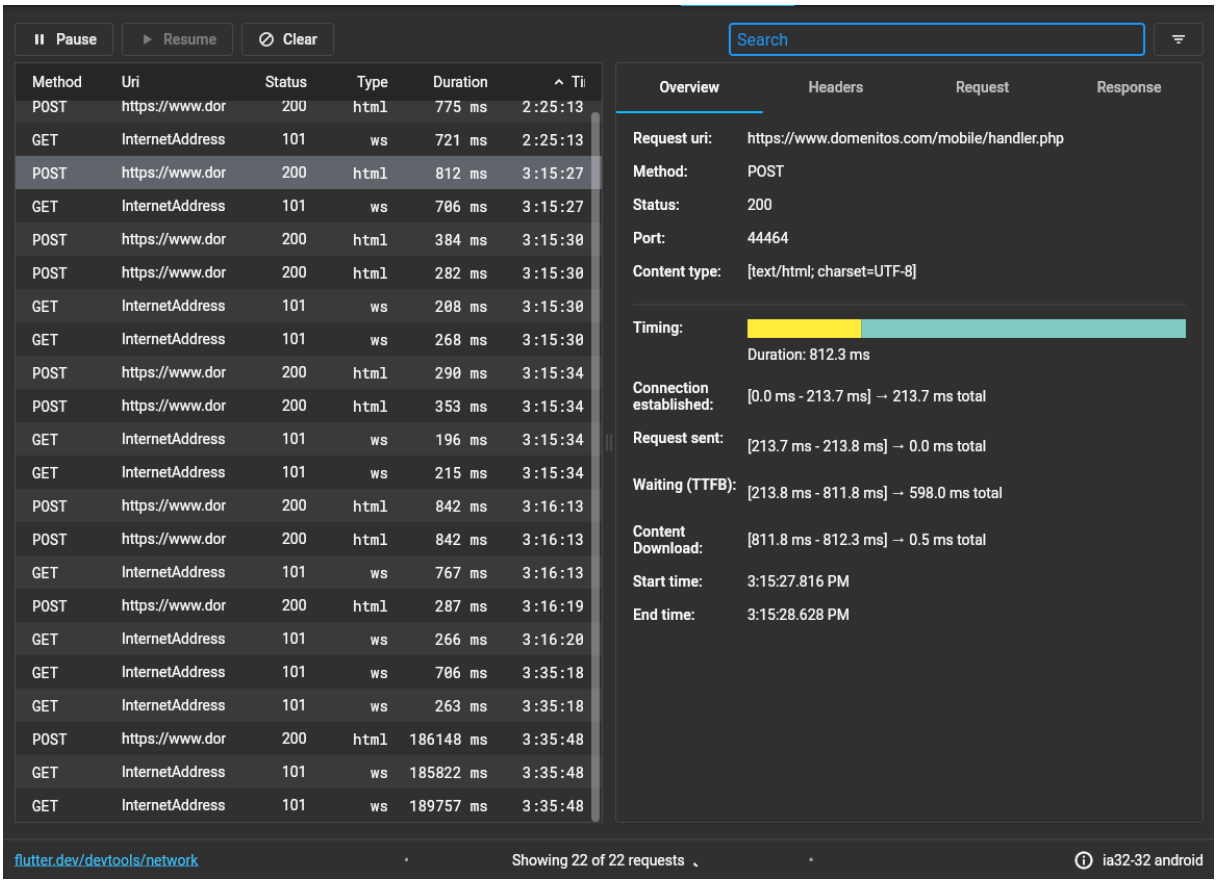


Slika 69. Ispravljač pogrešaka

Na gornjoj slici je vidljiva selektirana datoteka addTask.dart ona je pronađena preko pretraživača biblioteka. Postavljeno je par prijelomnih točki pri inicijalizaciji klase. Prijelomne točke gledaju na HTTP poziv za dobivanje svih informacija o čistačima kompanije.Preko varijabla vidimo da je su napunjene interne mape s kojima će se kasnije u aplikaciji popuniti padajući izbornik za odabir čistača za novi zadatak.

7.8.5 Pregled mreže

Preglednik mreže omogućava inspekciju HTTP, HTTPs i ostali web promet u *Flutter* aplikaciji. Na slici 70. vidljiv je pregled mreže.

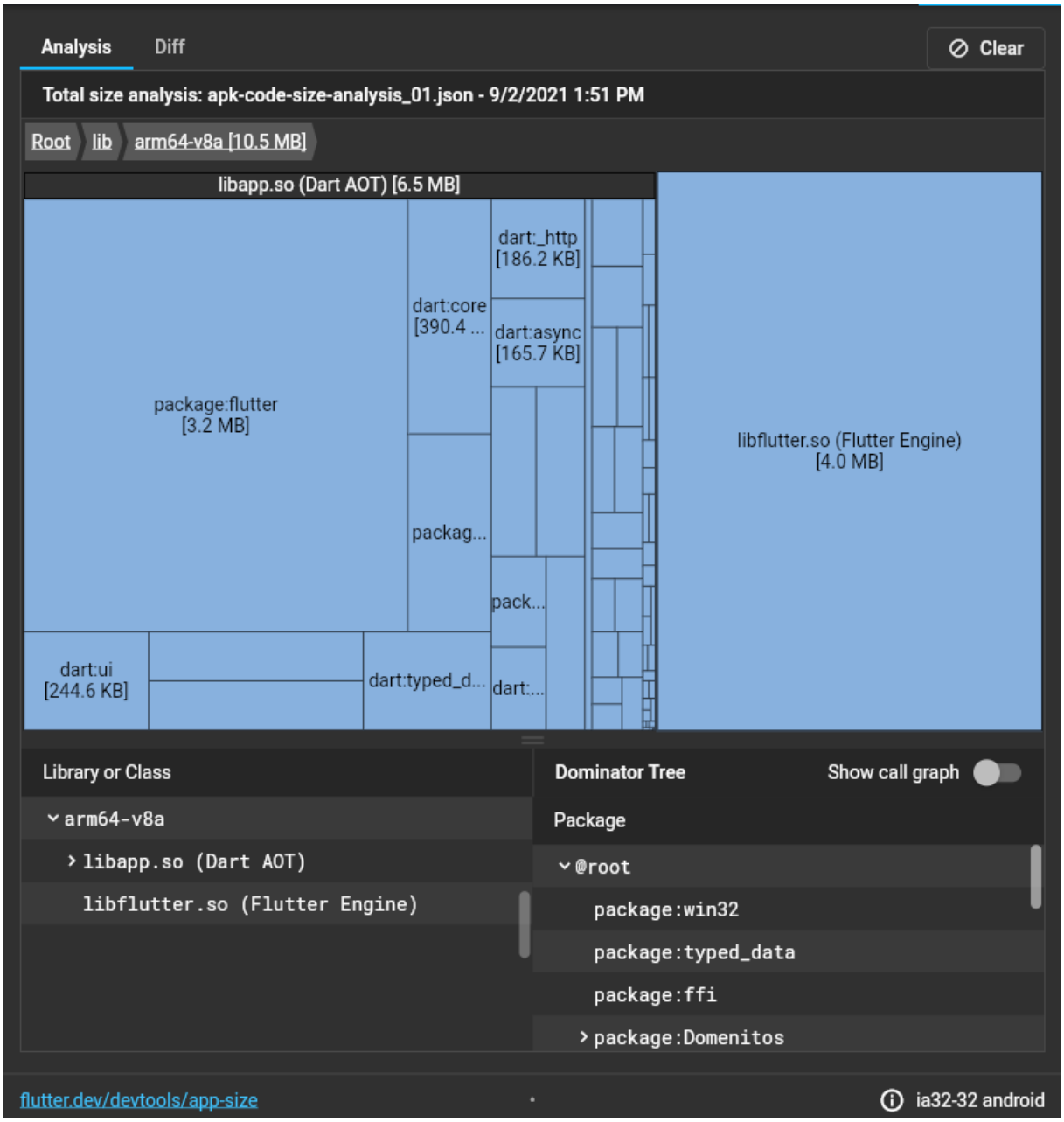


Slika 70. Pregled mreže

Na gornjoj slici imamo listu svih zahtjeva prema mreži koji su postignuti od pokretanja aplikacije. Za svaki zahtjev je vidljiv statusni kod, vrsta, tip, metoda, vrijeme izvođenja i trajanje izvođenje. Zahtjev ima svoje detalje od kojih su najkorisniji sami poslani zahtjev i odgovor web servera. S ovim je olakšano praćenje i ispravljanje pogrešaka kod zahtjeva.

7.8.6 Alat za analizu veličine aplikacije

Jedan od glavnih problema *Flutter* aplikacije je njihova veličina, zbog toga ovaj alat omogućava pronalazak suvišnih datoteka, prevelikih dijelova aplikacije. Za analizu je potrebno izvesti naredbenu liniju „flutter build apk --analyze-size --target-platform=android-arm64“ koja generira datoteku koja se analizira. Na slici 71. vidljiv je alat za pregled veličine aplikacije.



Slika 71. Alat za pregled veličine aplikacije

Alat omogućuje detaljni pregled svih elemenata aplikacije. Omogućuje uvid u neizbrisane biblioteke ili klase koje se ne koriste. Analiza se može raditi i usporedbom s kojom je moguće vidjeti sve promjene u veličini između dvije verzije aplikacije.

7.9 Nedostatci *Darta*

Nedostatci *Darta* su u pronalasku dovoljno programera koji ga znaju jer taj programski jezik koristi samu u *Flutteru.* Za razliku od klasičnih programskih jezika poput JavaScripta, C#-a, Jave ili Objective-Ca *Dart* je noviji programerski jezik s paradigmom koja može biti poznata samo Java i C# programerima. Trenutačno se još uvijek razvija što uvijek ostavlja šansu da će se neki API promijeniti i da dokumentacija neće biti puna i točna. Cjelokupna količina materijala na internetu je znatno manja nego kod drugih popularnijih programskih jezika.

7.10 Veličina aplikacija

Svakom programeru mobilnih aplikacija cilj je napraviti što manju aplikaciju kako ju korisnici ne bi izbrisali zbog relativnog malog prostora na mobilnim uređajima. *Flutter* aplikacije će biti veće od izvornih aplikacija napravljenih u izvornoj platformi, a razlog za to je što on koristi svoje *widgete* umjesto platformno specifičnih *widgeta*. Najmanja aplikacija će biti 5MB za razliku od 2MB za izvorne platforme, glavni razlog tome je *Flutter engine* čija je veličina fiksna i iznosi 4 MB vidljivo na slici 71. Problem se neće riješiti na većim aplikacijama nego će veličina ostati veća za do 30%.

7.11 Manjak biblioteka

Biblioteke i paketi napravljeni od stane neovisnih proizvođača igraju veliku ulogu u automatizaciji razvoja softvera i oslobađanje od potrebe programiranja svega od nule. Biblioteke su uglavnom otvorenog koda, lagano dostupne i dobro istestirane. Za većinu starijih tehnologija i programerskih jezika broj takvih biblioteka je nekoliko puta veći od broja biblioteka *Fluttera*. Što je manji broj biblioteka to je veća šansa da će se nešto morati programirati od početka što usporava vrijeme razvoja aplikacije.

8 ZAKLJUČAK

U diplomskom radu tijekom uspješne izrade aplikacije Domenitos na web, mobilnoj i desktop platformi pokazano je znanje *Fluttera* i programskog jezika *Dart*, za izradu klijentskog dijela aplikacije, te znanje PHP-a u pravljenju API-ja kao poslužiteljskog dijela aplikacije.

Proučen je pristup razvoju višeplatformskih aplikacija korištenjem iste baze koda te su analizirane i uspoređene dvije danas najpopularnije platforme za takav razvoj: *Flutter* i *React Native*. Analizirane su prednosti i nedostaci takvog razvoja i zaključeno je da je razvoj višeplatformskih aplikacija koristan u velikom broju slučajeva. Glavna stavka koja odlučuje hoće li neko poduzeće koristiti višeplatformske tehnologije su performanse u kojima su te tehnologije znatno lošije od izvornih tehnologija za te platforme. *Flutter* i *React Native* su dosta slični i odabir tehnologije će ovisiti o trenutačnom dostupnom ljudskom kadru. *Flutterova* je prednost njegova moguća buduća upotreba na raznim platformama na kojima zbog savitljivog *Dart* programskog jezika.

Tijekom izrade stečeno je dublje znanje korištenja DevTools alata koji je jedan od najboljih alata za analizu, pronalazak pogrešaka unutar aplikacije. Kroz analizu tržišta tehnologija koje podržavaju više platformsku izradu aplikacija utvrđeno je odlična pozicija *Fluttera* na trenutačnom tržištu te njegov mogući veći udio u skoroj budućnosti. Kroz pregled pogodnosti i nedostataka *Fluttera* utvrđeno je veliki broj prednosti u usporedbi s trenutačnom konkurencijom. Nadogradnja aplikacije bi uključivala više mogućnosti s Google mapama tako da se čistačima omogući najkraći put sa sadašnje lokacije do lokacije zadataka.

Literatura

[1] »Flutter,« [Mrežno]. Available: https://flutter.dev/. [Pokušaj pristupa 5 rujan 2021].

[2] »Dart,« [Mrežno]. Available: https://dart.dev/. [Pokušaj pristupa 6 rujan 2021].

[3] »Flutter.dev,« [Mrežno]. Available: https://flutter.dev/docs/resources/faq. [Pokušaj pristupa Ponedeljak lipanj 2021].

[4] »Dart.dev,« [Mrežno]. Available: https://dart.dev/overview. [Pokušaj pristupa 5 lipanj 2021].

[5] »Flutter,« [Mrežno]. Available: https://flutter.dev/docs/resources/architectural-overview. [Pokušaj pristupa 25 kolovoz 2021].

[6] »Flutter,« [Mrežno]. Available: https://flutter.dev/docs/development/tools/devtools/overview. [Pokušaj pristupa 8 rujan 2021].

[7] »developer.android,« [Mrežno]. Available: https://developer.android.com/studio/intro. [Pokušaj pristupa 2 6 2021].

[8] »Flutter,« [Mrežno]. Available: https://flutter.dev/desktop. [Pokušaj pristupa 24 kolovoz 2021].

[9] »Flutter,« [Mrežno]. Available: https://flutter.dev/web. [Pokušaj pristupa 15 kolovoz 2021].

[10] »Skia,« [Mrežno]. Available: https://skia.org/. [Pokušaj pristupa 16 kolovoz 2021].

[11] K. Shah, »SoluteLabs,« [Mrežno]. Available: https://www.solutelabs.com/blog/flutter-for-web-an-ultimate-guide. [Pokušaj pristupa 17 kolovoz 2021].

[12] »MDN WEB Docs,« [Mrežno]. Available: https://developer.mozilla.org/en-US/docs/Web/HTTP/CORS. [Pokušaj pristupa 20 kolovoz 2021].

[13] »statista,« [Mrežno]. Available: https://www.statista.com/statistics/869224/worldwide-software-developer-working-hours/. [Pokušaj pristupa 28 kolovoz 2021].

[14] »Stack Overflow,« [Mrežno]. Available: https://insights.stackoverflow.com/trends?tags=react-native%2Cflutter%2Ccordova%2Cxamarin. [Pokušaj pristupa 29 kolovoz 2021].

[15] »Medium,« [Mrežno]. Available: https://medium.com/swlh/flutter-vs-native-vs-react-native-examining-performance-31338f081980. [Pokušaj pristupa 28 kolovoz 2021].

[16] »Stack Overflow,« [Mrežno]. Available: https://insights.stackoverflow.com/survey/2021#technology. [Pokušaj pristupa 30 kolovoz 2021].

[17] »CodeBurst,« [Mrežno]. Available: https://codeburst.io/flutter-or-react-native-which-is-the-best-choice-for-2020-355d9473edde. [Pokušaj pristupa 31 kolovoz 2021].

[18] »Relevant,« [Mrežno]. Available: https://relevant.software/blog/top-8-flutter-advantages-and-why-you-should-try-flutter-on-your-next-project/. [Pokušaj pristupa 22 kolovoz 2021].

[19] »Flutter,« [Mrežno]. Available: https://flutter.dev/docs/development/platform-integration/platform-channels. [Pokušaj pristupa 31 kolovoz 2021].

9 PRILOZI

9.1 Popis slika

Slika 1. Arhitektura *Fluttera* [4] 3
Slika 2. Struktura projekta u Android Studiju 5
Slika 3. Zahtjev pregled zadataka 7
Slika 4. Zahtjev dodavanje zadataka 8
Slika 5. Zahtjev pregled čistača 9
Slika 6. Zahtjev dodavanje čistača 10
Slika 7. Zahtjev pregled apartmana 11
Slika 8. Zahtjev dodavanje apartmana 12
Slika 9. Zahtjev izvršenje zadatka 13
Slika 10. Zahtjev pregled vlasnika 14
Slika 11. Zahtjev dodavanje vlasnika 15
Slika 12. Zahtjev promjena jezika 16
Slika 13. Zahtjev promjena lozinke 17
Slika 14. Zahtjev ažuriranje apartmana 18
Slika 15. Zahtjev ažuriranje vlasnika 19
Slika 16. Zahtjev ažuriranje čistača 20
Slika 17. Dijagram komponenti 22
Slika 18. Model baze podataka 23
Slika 19. Dijagram razmještaja 24
Slika 20. Login ekran 25
Slika 21. Početni ekran 26
Slika 22. *Drawer* 27
Slika 23. Ekran Vlasnici 28
Slika 24. Ekran dodavanje vlasnika 28
Slika 25. Ekran apartman 29
Slika 26. Ekran dodaje apartmana 29
Slika 27. Detalji apartmana 29
Slika 28. Google Map lokacija apartmana 29
Slika 29. Ekran Čistači 30

Slika 30. Dodaj Čistača 30
Slika 31. Ekran Zadaci 31
Slika 32. Dodaj zadatak 31
Slika 33. Početni ekran podsustav čistači 31
Slika 34. Ekran zadaci čistača 32
Slika 35. Ekran postavki 33
Slika 36. Kod *RouteFactory* 33
Slika 37. Login conformation 34
Slika 38. Arhitektura web *Fluttera* [8] 35
Slika 39. Naredbeni redak za detekciju uređaja 36
Slika 40. Naredbeni redak za izvođenje web aplikacije 37
Slika 41. Korištene biblioteke 37
Slika 42. Zakomentirani dio main.dart datoteke 38
Slika 43. Razlika između dvije biblioteke u kodu 39
Slika 44. Dodavanje PHP funkcije header 39
Slika 45. Korištenje funkcije MediaQuery 40
Slika 46. Responzivan dizajn širine veće od 575 piksela 41
Slika 47. Responzivan dizajn širine duže od 1200 piksela 42
Slika 48. Izvođenje naredbenog retka konfiguracije windows aplikacije 43
Slika 49. Pogreška pri pokretanju Windows aplikacije 43
Slika 50. Windows aplikacija Domenitos 44
Slika 51. Graf najpopularnijih mobilnih okvira za više platformi u razdoblju od 2019. do 2021. [13] 46
Slika 52. Graf postavljenih pitanja za tehnologije [14] 47
Slika 53. Procesno intenzivan test za iOS [15] 48
Slika 54. Procesno intenzivan test za Android [15] 48
Slika 55. Graf utiska tehnologije okvira za razvoj aplikacija koje se nekoriste na webu [16] 49
Slika 56. Graf željom za učenjem okvira za razvoj aplikacija koje se nekoriste na webu [16] 50
Slika 57. Graf popularnosti okvira za razvoj aplikacija koje se nekoriste na webu [16] 50
Slika 58. Graf Google trendova između *Fluttera* i *React Nativa* 51
Slika 59. Graf srednjih godišnjih plaća u USD [16] 52
Slika 60. Okvir prikazivanja korisničkog sučelja *React Nativa* [18] 55

Slika 61. Stvaranje korisničkog sučelja [18] 56
Slika 62. Arhitektura platformnih kanala [19] 57
Slika 63. Pokretanje DevToolsa preko AndroidStudija 59
Slika 64. Korisničko sučelje *Flutter* inspektora 60
Slika 65. Graf frekvencije okvira 61
Slika 66. Pregled funkcija obavljenih u jednom kadru 61
Slika 67. Grafikon događaja na vremenskoj traci 62
Slika 68. Praćenje memorije 62
Slika 69. Ispravljač pogrešaka 63
Slika 70. Pregled mreže 64
Slika 71. Alat za pregled veličine aplikacije 65

9.2 Popis tablica

Tablica 1. Funckionalni zahtjevi 7

9.3 Isporuka sustava

9.3.1 Izvorni kod programskog rješenja

Izvorni kod programskog rješenja se nalazi na linku datoteka se zove Domenitos.rar. Nakon raspakiranja glavne datoteke se nalaze na lokaciji flutter_app/lib/. Izvorni kod API-a se nalazi na liku i datoteka se zove handler.php u kojemu se nalazi API.

9.3.2 Upute za instalaciju

Za instalaciju mobilne aplikacije Domenitos treba preuzeti apk na dostupan na adresi. Desktop aplikacija se nalazi na linku i u njoj se nalazi datoteka desktop_aplikacija_domenitos.rar koja kad se raspakira može se pokrenuti preko .exe datoteke. Web aplikacija se nalazi na linku. Pristupni podaci za testiranje su:

Tip	Username	Password
Administrator	mijamijalic@gmail.com	Marin
Administrator	testing@gmail.com	test
Čistač	mmarkota@gmail.com	password
Čistač	miho.neric@gmail.com	password

Čistač	1234@xyz.com	password

IZJAVA

Izjavljujem pod punom moralnom odgovornošću da sam diplomski rad izradio samostalno, isključivo znanjem stečenim na studijima Sveučilišta u Dubrovniku, služeći se navedenim izvorima podataka i uz stručno vodstvo mentora doc. dr. sc. Krunoslav Žubrinić kome se još jednom srdačno zahvaljujem.

Marin Mijalić