VIŠJA STROKOVNA ŠOLA ACADEMIA

MARIBOR

Primerjava ogrodij Flutter in NET Maui na primeru aplikacije Divja odlagališča

Kandidat: Benjamin Kamenica

Vrsta študija: študent izrednega študija

Študijski program: Informatika

Mentor predavatelj: mag. Ervin Schaff

Mentor v podjetju: Martin Burgstaller, BEng

Lektor: Darja Medved, PRU slovenščine in angleščine

Maribor, 2022

IZJAVA O AVTORSTVU DIPLOMSKEGA DELA

Podpisani Benjamin Kamenica sem avtor diplomskega dela z naslovom Primerjava ogrodij Flutter in .NET Maui na primeru aplikacije Divja odlagališča, ki sem ga napisal pod mentorstvom mag. Ervina Schaff.

S svojim podpisom zagotavljam, da:

- je predloženo delo izključno rezultat mojega dela,
- sem poskrbel, da so dela in mnenja drugih avtorjev, ki jih uporabljam v predloženi nalogi, navedena oz. citirana skladno s pravili Višje strokovne šole Academia Maribor,
- se zavedam, da je plagiatorstvo – predstavljanje tujih del oz. misli kot mojih lastnih – kaznivo po Zakonu o avtorskih in sorodnih pravicah (Uradni list RS, št. 16/07 – uradno prečiščeno besedilo, 68/08, 110/13, 56/15 in 63/16 – ZKUASP); prekršek pa podleže tudi ukrepom Višje strokovne šole Academia Maribor, skladno z njenimi pravili,
- skladno z 32.a členom ZASP dovoljujem Višji strokovni šoli Academia Maribor objavo diplomskega dela na spletnem portalu šole.

Maribor, december 2022 Podpis študenta:

ZAHVALA

Zahvalil bi se mentorju mag. Ervin Schaffu za izjemno odzivnosti pri izdelavi diplomskega dela.

Zahvalil bi se tudi mami Mirsadi, očetu Osmanu in sestri Aleni.

Posebej pa bi se rad srčno zahvalil partnerici Alex, ki mi vedno stoji ob strani in pomaga. Hvala za ljubezen in podporo. Hvala, ker si.

POVZETEK

Flutter in .NET Maui sta ogrodji, ki omogočata razvoj aplikacij, ki so uporabne na različnih platformah. V sklopu priprave diplomskega dela sem se odločil, da razvijem mobilni aplikaciji, ki sta uporabni na platformi Android. Namen diplomske naloge je, razviti aplikacijo, ki bi pripomogla pri ozaveščanju ljudi glede divjih odlagališč. Izbral sem najbolj pogosto uporabljeni ogrodji, in sicer Flutter in .NET Maui ter ju primerjal z vidika razvijalca in uporabnika. Primerjal sem tudi kako v posameznih situacijah ogrodji delujeta.

Najprej sem raziskal področje razvoja mobilnih aplikacij nato pa tudi posamezno ogrodje, ki sem ju primerjal in se naučil programskega jezika. Prav tako sem raziskal področje divjih odlagališč in nato pripravil načrt potrebnih funkcij aplikacije, ki bi reševala problem divjih odlagališč.

Kot prvi korak sem si zastavil razvoj aplikacije s pomočjo ogrodja Flutter. Uporabil sem programski jezik Dart in implementiral aplikacijo s pomočjo različnih brezplačnih vmesnikov, le ti so mi pomagali pri implementaciji prijave, registracije, zemljevida, fotografiranja ipd. Za razvojno okolje sem uporabil Googlov Android Studio.

Naslednji korak je bil razviti identično aplikacijo še v ogrodju .NET Maui. Zaradi poznavanja jezika C# mi je bila implementacija lažja, predvsem kar se tiče pisanja kode. Sem pa vendarle naletel na težave, saj je ogrodje še ni dalj časa dostopno na spletu. Na težave sem naletel predvsem iz razloga, saj je veliko knjižnic in vmesnikov še nepopolnih ali nedostopnih. Pomagal sem si z implementacijo nekaj dodatnih funkcij, ki so v ogrodju Flutter že definirane, da bi dobil želene rezultate.

Ko sta bili aplikaciji končani sem lahko preizkusil razvoj aplikacij v obeh ogrodjih in argumentiral, katero ogrodje je primernejše za razvijalce po mojem mnenju. Testiral sem tudi samo delovanje posameznih aplikacij v posameznih scenarijih in tudi uporabniški vmesnik, ki nam pove, katera aplikacija je primernejša za vsakdanjega uporabnika.

Rezultate testiranj in izkušnje razvoja sem združil in povzel v povezano celoto.

Ključne besede: Flutter, .NET Maui, mobilne aplikacije, razvoj android aplikacije, divja odlagališča.

ABSTRACT

Comparing Flutter and .NET Maui frameworks in regards to illegal dumping sites application

Flutter and .NET Maui are frameworks that allow applications development which can be used on several platforms. I've decided to develop mobile applications which can be used on the Android platform. The goal of this thesis is developing an application to help raise awareness among people about illegal dumping sites. I chose the most used frameworks which are Flutter and .NET Maui and compared them from a developer and user point of view. I also compared the wofkflows inside both frameworks.

First I did some research on mobile applications development, as well as the individual frameworks. I compared the frameworks and learned the programming language. I also looked into illegal dumping sites and prepared a plan for the necessary application functions which would help solving the illegal dumping sites problem.The first step was to develop an application, using the Flutter framework. Programming language called Dart was used in this process. Then I implemented an application, using different free plugins. They helped me with the implementation of sign ins, registrations, maps, photographing etc. For the Integrated development environment, I used Google Android Studio.

The next step was to develop an identical application in the .NET Maui framework. Since I know the C# language very well, implementing was easier, especially when it comes to writing the code. I only had issues because the framework is new, which means a lot of libraries and plugins are still not developed, are incomplete or inaccessible. For that reason, I had to implement a few additional functions which are already defined in the Flutter framework, to get the desired results.

This way, I could test the application development in both frameworks and establish, which framework is more suitable for developers in my opinion. I also tested the functions of individual applications in different scenarios and the user interface which tells us in the end which application works better for an everyday user.I have combined the results of the tests and the development experience into a coherent whole.

Keywords: Flutter, .NET Maui, mobile applications, developing android applications, illegal dumping sites.

KAZALO VSEBINE

KAZALO SLIK

KAZALO TABEL

Razlaga strokovnih izrazov in kratic

- Vroče ponovno nalaganje (ang. Hot reload) : omogoča ogled uporabljenih sprememb skoraj v trenutku, ne da bi izgubili trenutno stanje aplikacije. Roheel, (brez datuma)
- Zaslon z napako(ang. Error screen): zaslon, ki prikaže trenutno napako za lažjo predstavo težave.
- Čista arhitektura (ang. Clean Architecture): način organiziranega programiranja
- MVP (ang. model-view-presenter): arhitekturni način programiranja uporabniškega vmesnika za lažje testiranje.
- API (ang. Application programming interface): programerski vmesnik, pomaga pri sporazumevanju med programi.
- Gradniki (ang. Widget): grafični vmesnik, ki prikazuje različne informacije uporabniku.
- Sproščanje pomnilnika (ang. Garbage collection): funkcija, ki skrbi za sproščanje pomnilika, z brisanjem nedostopnih in starih objektov.
- SDK (ang. Software Development Kit): set orodij, ki omogočajo razvijalcu razvoj aplikacije.
- AOT (ang. Ahed of time): kompilacija višje programske kode v nižjo, da prihranimo čas pri zagonu aplikacije.
- JIT (ang. Just-in-time): kompilacija ob zahtevi, namesto predhodne kot pri AOT.
- Upravljalec (ang. Handler): funkcija, rutina ali metoda, ki je specializirana za poseben tip podatkov ali posebne naloge.
- Preslikovalnik (ang. Controler): je slovar, ki ima lastnosti definirane v vmesniku določenega kontrolnika, ki ponuja neposreden dostop do objekta izvirnega pogleda. (Basu, 2021)
- Virtualna naprava (ang.Virtual maschine):
- Razvojno okolje (ang. IDE): je aplikacija, ki združuje osnovna in napredna orodja v enoten grafični vmesnik (ang. GUI) in tako omogoča učinkovit razvoj novih aplikacij.
- Grafični vmesnik (ang. GUI): grafični vmesnik, ki kodo pretvori v enostavno, razumljivo sliko.
- Razhroščevanje (ang.Debugging): Je orodje, ki testira in razhrošča aplikacijo. Napako najde v izvorni kodi in jo grafično prikaže.
- Samodejno dokončevanja ukaza (ang. Auto-completion):
- Avtomatizacija gradnje programa (ang. Software build):

- VCS (ang. Version control systems): programska orodja, ki jih razvijalci uporabljajo za upravljanje sprememb v izvorni kodi skozi čas, kar omogoča hitrejše in pametnejše delo. Najbolj uporabljan primer je recimo Git.
- Paketi (ang.Packages): Paket je imenski prostor, ki organizira niz sorodnih razredov in vmesnikov.
- Ogrodja (ang. Framework): Okvir v programiranju je orodje, ki zagotavlja pripravljene komponente ali rešitve, ki se prilagodijo, da se pospeši razvoj.
- Vmesnik (ang. Plugin): Programska oprema, ki gostiteljskemu programu dodaja nove funkcije, ne da bi spremenila sam gostiteljski program.

1 UVOD

1.1 Opis področja in opredelitev problema

Na področju razvoja mobilnih aplikacij prevladujeta podjetji Google in Microsoft. Google je s svojim ogrodjem Flutter in jezikom Dart postal eden bolj uporabljenih ogrodij za razvoj aplikacij za mobilne platforme. (Roheel, brez datuma) Slednji je pariral s Xamarinom in v letošnjem letu izdal .NET Maui, ki je postal novi predstavnik Microsofta v razvijanju mobilnih aplikacijah. (What is .NET MAUI?, 2022)

Z razvojem mobilne aplikacije Divja odlagališča sem poskušal pripomoči k ozaveščanju ljudi glede problematike divjih odlagališč. V Sloveniji imamo veliko gozdov in neokrnjene narave, kar pa marsikdo uporabi za divje odlaganje smeti. Zabeleženih je okoli 15.000 divjih odlagališč, ki so večinoma v jamah, breznih, gozdovih, mnogokrat pa tudi morje naplavi kopico odpadkov. Velik problem je, saj se na divjih odlagališčih pojavljajo tudi strupeni odpadki. Zaradi modernega življenjskega sloga se je povečala količina odpadnih in zavrženih stvari. Divji odpadki predstavljajo grožnjo živalim in ljudem, saj pogosto odpadki vsebujejo nevarne snovi, ki prodrejo v podtalnico in posledično onesnažujejo pitno vodo.

S pomočjo aplikacije lahko posamezniki sami označujejo divja odlagališča v okolici in jih dokumentirajo. Aplikacija bi se lahko uporabila tudi v različnih čistilnih akcijah, kot npr. Očistimo Slovenijo.

Z razvojem dveh aplikacij sem poizkušal primerjati razvoj, uporabnost in hitrost, ki bi najbolj ustrezala tako razvijalcu kot uporabniku. Področje raziskave, ki sem jo opravljal v sklopu diplomskega dela, je izdelava androidnih aplikacij v ogrodjih Flutter in .NET Maui ter preverjanje učinkovitosti uporabe kamere in geolokacije. V diplomskem delu sem prav tako primerjal hitrost in odzivnost uporabe teh senzorjev. Obenem pa sem si zastavil cilj, raziskati, katero ogrodje je primernejše za začetnike oziroma katero orodje je lažje učljivo. Prav tako pa sem primerjal razvojna jezika in stroške, ki nastanejo ob implementaciji aplikacij v posameznem ogrodju.

Na spletu je več dostopnega gradiva za ogrodje Flutter, saj je dalj časa na voljo uporabnikom. Prav tako je za Flutter dostopnih več vmesnikov, saj za .NET Maui šele prehajajo iz ogrodja Xamarin, kar posledično pomeni, da še veliko vmesnikov za .NET Maui ni optimiziranih. Zaradi novejše narave .NET Maui je posledično manj učnega gradiva in primerov uporabe. Obe ogrodji tako Flutter kakor tudi .NET Maui pa imata svoje dokumentacije (ang. Docs). Prav tako

je implementacija enake aplikacije v .NET Maui veliko zahtevnejša kot v ogrodju Flutter. Tudi programski jezik c# je bolj uporabljen in znan kot programski jezik Dart, ki se uporablja v Flutter-ju.

1.2 Namen, cilji in osnovne trditve

Namen raziskave je bil, odgovoriti na vprašanja:

- Katero orodje je primernejše za začetnika? Odgovorjeno v poglavju 2.4.
- Katero ogrodje je bolj uporabljeno oz. ima večjo skupnost? Odgovorjeno v poglavju 2.4.
- Ali so na spletu dostopni primerljivi učni pripomočki in gradiva za učenje ogrodja Flutter (Dart) in .NET Maui (C#)? Odgovorjeno v poglavju 2.3.
- Katero ogrodje je pri izdelavi manjših aplikacij stroškovno bolj dostopno? Odgovorjeno v poglavju 2.4.
- Kako bi implementiral enostavno aplikacijo za divja odlagališča? Odgovorjeno v poglavju 3.1.
- Kakšne funkcionalnosti bi taka aplikacija potrebovala? Odgovorjeno v poglavju 3.1.
- Kako bi izmeril performanse aplikacij v obeh ogrodjih? Odgovorjeno v poglavju 4.
- Kako deluje uporaba geolokacije v obeh ogrodjih? Odgovorjeno v poglavjih 3.3 in 3.4.
- Kako bi izmeril odzivnost aplikacije, ki uporablja geolokacijo? Odgovorjeno v poglavju 4.

Poskusil sem dokazati sledeče hipoteze:

- **H1:** Za implementacijo enostavne aplikacije je začetniku lažja uporaba ogrodja Flutter. Potrjeno v poglavju 2.4.
- **H2:** Za izdelavo enostavnih aplikacij je uporaba ogrodja Flutter stroškovno ugodnejša. Potrjeno v poglavju 2.4
- **H3:** Ob uporabi geolokacije ima aplikacija v ogrodju .NET Maui boljše performanse. Ovrženo v poglavju 4.

- **H4:** V obeh ogrodjih je možno implementirati aplikacijo, ki bi lahko pripomogla pri zmanjšanju onesnaževanja okolja. Potrjeno v poglavjih 3.3 in 3.4.

1.3 Predpostavke in omejitve

Pri pisanju diplomskega dela sem se omejil na ogrodji Flutter in .NET Maui. Omejil sem se tudi na prosto dostopna razvojna okolja, ki sta Android Studio in Microsoft Visual Studio. Aplikacijo sem testiral na lokalnem Android Device Emulator-ju. Omejil sem se tudi na uporabo paketov in razvojnih orodij, ki niso plačljiva.

1.4 Uporabljene raziskovalne metode

Pri izdelavi diplomskega dela si bom pomagal z literaturo primarnega in sekundarnega izvora, kot so strokovne knjige, internetni članki in osebni zapiski. V teoretičnem delu bom za opisovanje ogrodij uporabil deskriptivno metodo in komparativno metodo, ko bom ogrodji med seboj primerjal. V empiričnem delu bom s pomočjo metode abstrakcije predstavil implementirani aplikaciji in uporabil metodo analize, s katero bom analiziral rezultate meritev aplikacij v obeh ogrodjih. Na koncu bom rezultate predstavil z deskriptivno metodo in jih združil v zaokroženo celoto.

Področje raziskovanja je programiranje androidnih aplikacij v ogrodjih, ki so primerna za različne platforme. Raziskal bom tudi učinkovitost uporabe kamere in geolokacije v obeh ogrodjih in ju primerjal.

2 UPORABA IN UČENJE OGRODIJ

2.1 Flutter

Flutter je odprtokodno ogrodje, ki je namenjeno razvoju programske opreme (ang. SDK) in temelji na jeziku Dart. Je platforma za razvoj multiplatformskih aplikacij. Ogrodje Flutter je bilo predstavljeno leta 2015 s strani Googla. Razvit je bil z namenom hitrih gradenj aplikacij za iOS in Android. Pomembna odločitev, ki jo je sprejel Google, ko je načrtoval ogrodje Flutter, je prav to, da le to ogrodje upodablja lastne komponente uporabniškega vmesnika. Prav ta značilnost ga naredi drugačnega od drugih ogrodij. Za GUI uporablja grafično knjižnico Skia (C++), ki ga med drugim uporabljajo tudi Google Chrome, Chrome OS, Android, Mozilla Firefox in Firefox OS in številni drugi izdelki. Projekt Skia izvira že iz leta 1996, vendar ga je Google kupil šele leta 2005. Čeprav je le ta izdelan pod licenco BSD, ga lahko uporablja vsak. Flutter temelji na arhitekturi »enosmernega pretoka podatkov« ali reaktivnem programiranju. (Bellinaso, 2018) Ogrodje Flutter je sestavljeno iz dveh ključnih delov, in sicer iz Software Development Kit-a, kar so različa orodja, ki nam pomagajo pri implementaciji aplikacije in iz ogrodij, pri katerih gre za UI knjižnice, ki so sestavljene iz gradnikov.

Flutter omogoča komunikacijo z različnimi platformami, saj so zapakirane na enak način. Tako so vnosi, izvozi, sporočila in druge storitve napisane v skladnem jeziku s ciljno platformo. Za iOS objective C in C++, za Android C++ in Java, ter za Windows C++.

Motor Flutterja je zapisan v jeziku C++ in podpira primitivne in potrebne procese za podporo vseh Flutter aplikacij. Omogoča nizko raven osnovnega API-ja, vključno z grafičnim vmesnikom Skia.

Ogrodje Flutter je sestavljen iz gradnikov (ang. widget). Glavna ideja Googla je bila, da uporabniški vmesnik zgradimo z več gradniki, ki opišejo, kakšno je njihovo trenutno stanje in konfiguracija. V primeru, da se stanje gradnikov v ogrodju spremeni, se le ta gradnik ponovno zgradi. To je pomembno, da se lahko uporabijo le minimalne spremembe za prehod iz enega stanja v drugo. (Zaccagnino, 2020)

Ogrodje Flutter je razširljiv z vtičniki tretjih oseb, ki lahko razvijejo funkcije, ki so specifične za platformo, ki jih vgrajeni razredi še ne morejo pokrivati, kot so npr. za video in avdio, monetizacijo, shranjevanje, virtualno resničnost, kamero, strojno učenje in tako naprej. Lahko pa tudi dodajajo nove, po meri narisane komponente uporabniškega vmesnika. Flutter olajša pisanje kod, ki so specifične za platformo, torej da izvaja drugo kodo po preverjanju s funkcijo Platform.isIOS ter Platform.isAndroid. Za pisanje in ustvarjanje aplikacij se lahko uporabi katerikoli urejevalnik besedil, vseeno pa se priporoča uporaba enega od urejevalnikov, ki podpirajo vtičnik Flutter. Le ti so Android Studio, VS Code ali Intelli J Idea, ki omogočajo intelliSense, samodokončanje in nekatera orodja za odpravljanje napak ter druga, ki so potrebna za uporabo ukazne vrstice, prevajanje ali zagon aplikacije.

Leta 2017 je Google med konferenco razvijalcev Google IO javno izdal alfa različico lastne večplatformske tehnologije, ki se uporablja za mobilne aplikacije Flutter (0.0.6). (Bellinaso, 2018)

Flutter pa je svoj prvi zagon doživel šele v letu 2018, takrat je bila izdana verzija 1.0. Omenjena verzija je bila dostopna vsem razvijalcem.

V letošnjem letu je bil izdan Flutter 3, ki predstavlja ogromen mejnik, saj vključuje stabilno podporo za vse vrste platform, prav tako pa se zagon te izdaje ni upočasnil. Ta posodobitev ogrodja Flutter prinaša posodobitve za Web Flutter, izboljšanje delovanja obdelave besedila in podobno. Flutter 3.3 pa nudi še izboljšano podporo za vnos s sledilno ploščico, kar zagotavlja bolj gladki in bogatejši nadzor, prav tako pa zmanjša tudi napačno razlago v določenih primerih. Neverjetnim prispevkom člana skupnosti fbcouch pa sedaj Flutter podpira tudi vnos rokopisa Scribble z uporabo Apple Pencil v sistemu iPadOS. Ta funkcija pa je privzeto omogočena v CupertinoTextField, TextField in EditableText. Nova dodelana izdaja pa povečuje zmogljivost nalaganja, preprečuje nastanek kopij, ter zmanjša zbiranje smeti v kodi Dart (GC). (Chisholm, brez datuma)

2.1.1 Programski jezik Dart

Za delovanje mobilnih in spletnih aplikacij, katerih implementacijo omogoča Flutter, se uporablja programski jezik Dart, ki je bil leta 2011 ustvarjen s strani Googla. Uradna različica 1.0 pa je bila izdana šele leta 2013. Namen Googla je bil, da bi Dart nadomestil JavaScript. Večjo pozornost je programski jezik Dart dobil šele leta 2017, ko je Google predstavil ogrodje Flutter. Dart je programski jezik, ki je objektivno naravnan in temelji na razredih in je namenjen množični uporabi. Je strogo tipiziran in potreben prevajanja.

Naloga Darta je, da skrbi za sproščanje pomnilnika (ang. Garbage collection). Koncept Darta je podoben znanim programskim jezikom kot so C#, C, Java ter omenjenemu JavaScriptu, katerega namen je imel Dart nadomestiti. Sintaksa Darta je zastavljena na jeziku C. Običajne funkcije programskih jezikov so podprte s strani Darta-a, to so razredi mixin, abstraktni razredi in vmesniki. Dart podpirajo tudi orodja razvijalcev, kot so Android Studio, Visual Studio Code in IntelliJ IDEA. Dart predstavlja platformo za razvijalce spletnih in mobilnih aplikacij ter deluje obojestransko, in sicer kot strežnik in odjemalec.

Dart je sestavljen iz številnih uporabnih vgrajenih knjižnic, vključno s SDK (ang. Software Development Kit), core, math, async, math, convert, html, IO in tako dalje. Omogoča tudi možnost organiziranja kode v knjižnice z ustreznim imenskim prostorom. Le ta se lahko ponovno uporabi z uvoznim stavkom.

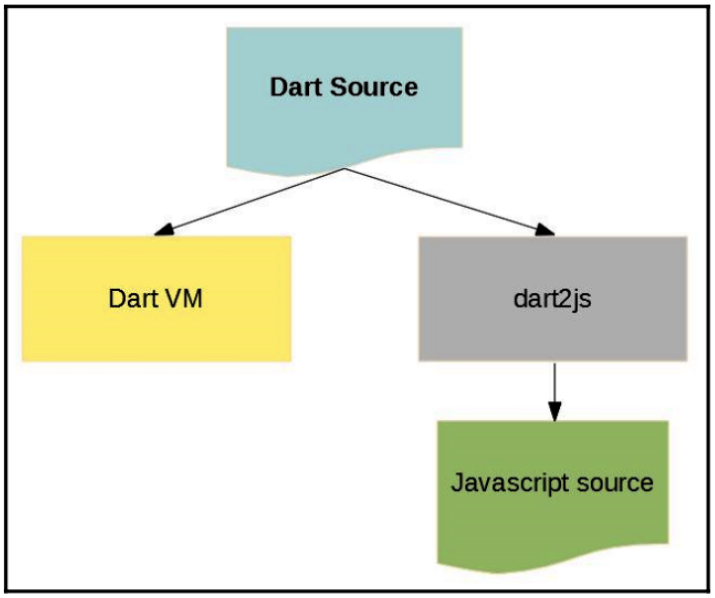
Dart omogoča prilagodljivost in hitrost prevajanja kode. Podpira dve vrsti procesov prevajanja, AOT (ang. Ahed of time) in JIT (Ang. Just-in-time). Koda Dart se prevede v drug jezik, ki ga lahko izvajajo sodobne spletne naprave.

Dart je tipsko varen jezik. To pomeni, da uporablja preverjanje statičnega tipa in preverjanje med izvajanjem. Z namenom, da se potrdi ujemanje vrednosti spremenljivke s statističnim tipom spremenljivke znanim.

Da bi lahko razumeli, od kod izvira prilagodljivost jezika, je potrebno vedeti, kako se lahko izvaja koda Dart. To lahko naredimo na dva načina, in sicer:

- Dart Virtual Machine (VM) ali
- z JavaScript kompilacijo.

Prikaz v diagramu:



Slika 1 : Primer izvajanja kode Dart

Vir: (https://miro.medium.com/max/721/1*5j8EQ7eAZYoN7Vif7i9TFw.png)

Imamo tudi možnost prevajanja jezika Dart v JavaScript s pomočjo dart2json prevajalnika.

2.1.2 Posebnosti ogrodja Flutter

V nadaljevanju bom predstavil nekaj posebnosti ogrodja Flutter.

Flutterjeva funkcija »vročega ponovnega nalaganja« (ang. Hot-reload) pa omogoča ogled uporabljenih sprememb skoraj v trenutku, ne da bi izgubili trenutno stanje aplikacije. In prav to je tisto, zaradi česar je razvoj aplikacije Flutter nekajkrat hitrejši zaradi povečane hitrosti razvoja in prihranka stroškov in truda.

Ogrodje Flutter razvijalcem omogoča pisanje kode, ki deluje na različnih platformah, to pomeni, da lahko dve različni aplikaciji uporabljata isto zbirko kod. Poleg deljenja kode uporabniškega vmesnika pa je mogoče deliti tudi sam uporabniški vmesnik. Zaradi tega je vzdrževanje ene kodne baze veliko lažje v nasprotju z različnimi kodami za različne platforme.

Ekipa Flutterja je vložila veliko truda v zagotavljanja najrazličnejših pripomočkov. Flutter sestavlja knjižnica gradnikov, saj je v njem vse definirano kot gradnik. Večina jih je zelo prilagodljivih in prihranijo čas kot še nobeno drugo orodje doslej. Flutter je sposoben ustvariti zapletene pripomočke, ki jih je mogoče prilagoditi, glede na zahteve same aplikacije. Poleg številnih osnovnih pripomočkov Flutter ponuja velik nabor pripomočkov, kot so - Cupertino in Material Design, ki popolnoma posnemajo vedenje vsakega oblikovnega jezika.

V ogrodju Flutter so na voljo pripomočki, ki so specificirani za platforme Google, Android, iOS in Fuchsia. Te gradnike pa je mogoče implementirati v aplikacijo Flutter, da se izkoristijo različne funkcionalnosti, ki so odvisne od posamezne platforme. Že obstoječe kode, kot so Java, Swift in Objective- C je mogoče uporabiti za izvorne funkcije, kot sta kamera in geolokacija. Prav zaradi tega Flutter zlahka vključi integracije in API-je tretjih oseb.

Ena največjih prednosti, ki jo ima ogrodje Flutter je ta, da ima možnost prilagajanja vsakega gradnika do potankosti.

Google je predstavil ogrodje Flutter kot odprtokodno platformo. Razvijalci aplikacij lahko raziščejo različne možnosti oblikovanja in ustvarjanja. Flutter je brezplačen in ima podrobno dokumentacijo ter skupnosti, ki so na voljo na spletu. (Roheel, brez datuma)

2.1.3 Prednosti in slabosti ogrodja

Prednosti

Dejstvo, da ogrodje Flutter sam izrisuje uporabniški vmesnik, namesto da bi predstavljal ovoj okoli izvornih komponent, ki so specifične za platformo, ima svoje prednosti in slabosti. Prednost je v tem, da če imamo nekaj na nek način upodobljeno na npr. telefonu iPhone z iOS, bi moralo biti upodobljeno na povsem enak način, ne samo v kateri koli drugi različici iOS, ampak tudi v kateremkoli telefonu, ki uporablja mobilni operacijski sistem Android. Z uporabo React Native ali Xamarin imajo komponente uporabniškega vmesnika številne lastnosti, ki so podprte le na eni ali drugi platformi ali pa so morda podprte, vendar prevedene na nekoliko drugačen način. Kar posledično pomeni, da morate aplikacijo preizkusiti na številnih napravah in različicah operacijskega sistema. Pride lahko celo do tega, da se aplikacija zruši, če se uporablja atribut ali funkcija, ki ni podprta v operacijskem sistemu. Tukaj se pokaže prednost ogrodja Flutter, saj je uporaba le tega veliko varnejša, predvsem, ko gre za grafični vmesnik. (Bellinaso, 2018)

Flutter uporablja programski jezik Dart, ki je preprost, močen in popoln jezik, v primerjavi s Swift-om, Kotlin-om ali Javo. Asinhrono programiranje z async/await/Future je preprosto, prav tako se zdi popolno in dosledno.

Ogrodje Flutter in programski jezik Dart imata vgrajeno podporo, ki omogoča testiranje enot za logiko in testiranje gradnikov za uporabniški vmesnik ali interakcije. Poiščejo se lahko podrejeni gradniki v drevesu gradnikov, prebira se lahko besedilo in se preveri, ali so vrednosti lastnosti gradnikov pravilne. Dokumentacija pa dobro prikazuje, kaj je na voljo.

Vroče ponovno nalaganje (ang. Hot Reload) je ena najbolj uporabnih funkcij za razvijalca. S klikom na gumb aplikacija znova zgradi in zažene spremenjene dele.

Slabosti

Flutter nariše uporabniški vmesnik na svoj način po meri in ne ustvarja izvornih komponent. Flutter opravlja zelo dobro delo pri podvajanju Androidovega materialnega oblikovanja in tudi komponent, ki so specifične za iOS, s svojo knjižnico Cupertino.

Pri uporabi grafičnih vmesnikov pri Flutter-ju izbira vtičnikov ni tako bogata kot na primer pri React Native in Xamarin, najverjetneje zato, ker je Flutter veliko novejši in z manjšo skupnostjo. Izbira je omejena in številni vtičniki so stari in niso vzdrževani. Nekatere komponente so na voljo samo za iOS ali Android, ne pa za oba.

Odpravljanje napak pri Flutter-ju ni najboljše. Uporabi se lahko izjave print ali debugPrint.

Zaslon z napako (ang. Error screen)k – ko pride do napake, so lahko zelo nejasni, saj kažejo na neko vrstico kode ogrodja, ki je lahko na drugi ravni kot tista, s katero neposredno komuniciramo. V izvornem sistemu iOS in Android so napake običajno jasnejše za razumevanje. V primeru, ko niso, se običajno lahko samo kopira in prilepi celotno napako v Google.

Programsko ustvarjanje grafičnega vmesnika je preprosto in neposredno, kar pa prinaša posledico, da ni veliko ločitve pri pisanju kode za aplikacijo in kreiranje grafičnega vmesnika. Bolje bi bilo ustvariti grafični vmesnik z označevalno kodo, podobno kot v izvornih aplikacijah za Android, in sicer v ločenih datotekah.

V sistemu Android velika večina razvijalcev uporablja čisto arhitekturo (ang. Clean Architecture) in MVP (ang. model-view-presenter). V sistemu iOS je lahko MVP, MVVM (ang. model-view-viewmodel) ali Viper. V obeh primerih obstajajo jasni in dobro znani arhitekturni vzorci, ki so se izkazali za dobre za obsežne aplikacije. Za Flutter pa še zmeraj ni standardnega oz. splošno sprejetega arhitekturnega pristopa. Vsebujejo preproste vzorce, saj imajo še vedno cilj, privabiti ljudi, preden začnejo razvijati naprednejše vidike.

2.2 .NET Maui

Ogrodje .NET Maui je razvoj tega, kar je trenutno Xamarin.Forms. Sedaj obstajata le dve knjižnici osnovnih razredov .NET 6 (BCL) in trenutno .NET 7 RC1. NET Multi- platform Appp IU (.NET MAUI) je večplatformsko ogrodje. Namenjeno je za ustvarjanje izvornih mobilnih in namiznih aplikacij s pomočjo C# in XAML. Z uporabo ogrodja .NET Maui se lahko razvijejo aplikacije, ki se izvajajo v sistemih, kot so iOS, macOS, Android in Windows iz ene skupne kodne baze. Pri .NET Maui gre za razvoj Xamarina in je odprtokodna rešitev. Z uporabo ogrodja .NET Maui lahko ustvarimo aplikacije za več platform in to z uporabo enega samega projekta. Po potrebi se lahko doda izvorna koda in viri, ki so specifični za posamezno platformo.

Eden od ključnih ciljev .NET Maui je, omogočiti postavitve uporabniškega vmesnika v eno samo kodno osnovo in implementirati večjo logiko aplikacije.

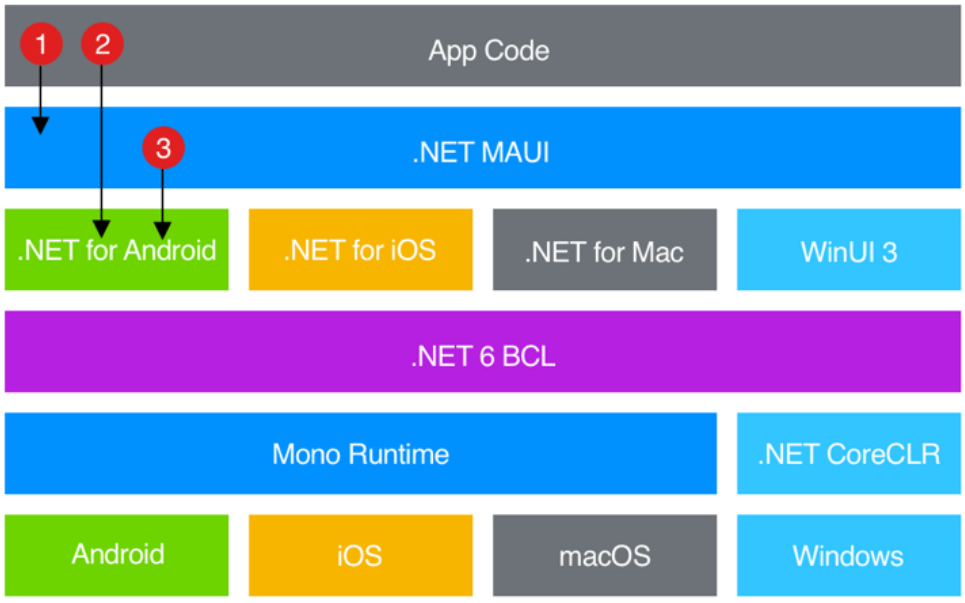
Ogrodje .NET Maui je primeren za razvijalce, ki želijo pisati aplikacije za več platform v XAML in C# iz ene skupne kodne baze v Visual Studio. Primeren je tudi za tiste, ki želijo deliti postavitev in dizajn uporabniškega vmesnika na različnih platformah. Delijo pa se lahko kode, testi in poslovna logika na različnih platformah.

Ogrodje .NET Maui združuje API-je za iOS, macOS, Windows in Android v en sam API (ang. Application programming interface), ki uporabnikom omogoča razvijalsko izkušnjo pisanja in izvajanja kjerkoli. Hkrati pa ogrodje .NET Maui zagotavlja globok dostop do vseh vidikov vsake izvorne platforme.

Ogrodje .NET 6 ponuja uporabnikom vrsto ogrodij za ustvarjanje aplikacij, ki so specifične za platformo, to so že omenjeni iOS, macOS, Windows in Android. Vsa ta ogrodja imajo dostop do iste knjižice osnovnih razredov .NET 6 (BCL). Omenjena knjižnica črpa podrobnosti o osnovni platformi strani iz vaše kode. BCL je odvisen od izvajalnega okolja .NET, ki zagotavlja izvajalno okolje za vašo kodo. Za iOS, macOS in Android okolje izvaja Mono, ki je implementacija izvajalnega okolja .NET. V sistemu Windows .NET CoreCLR zagotavlja okolje izvajanja.

BCL omogoča aplikacijam, ki se izvajajo na različnih platformah, skupno poslovno logiko. Različne platforme imajo različne načine, kako definirajo uporabniški vmesnik za aplikacijo in kako zagotavljajo različne modele za določanje, kako elementi uporabniškega vmesnika komunicirajo in delujejo med seboj. Uporabniški vmesnik se lahko izdela za vsako platformo posebej z uporabo ustreznega orodja, ki je specifično za vsako posamezno platformo. Za iOS in Android se uporabi .NET, za macOS pa se lahko uporabi ali .NET ali WinUI 3. Takšen pristop zahteva, da se vzdržujejo osnove kod za vsako posamezno družino naprav.

Diagram prikazuje pogled na arhitekture aplikacije .NET Maui. V aplikaciji .NET Maui se uporabi koda, ki primarno komunicira z aplikacijskim programskim vmesnikom (API) .NET Maui, kar prikazuje slika številka dve. NET Maui nato neposredno uporabi API izvorne platforme, kar je prikazano s številko tri, poleg tega pa lahko koda aplikacije neposredno izvaja API-je platforme, če je to potrebno, kar prikazuje številka dva.



Slika 2 : Arhitektura aplikacije .NET Maui

Vir: (https://inspeerity.com/wp-content/uploads/2022/06/architecture.png)

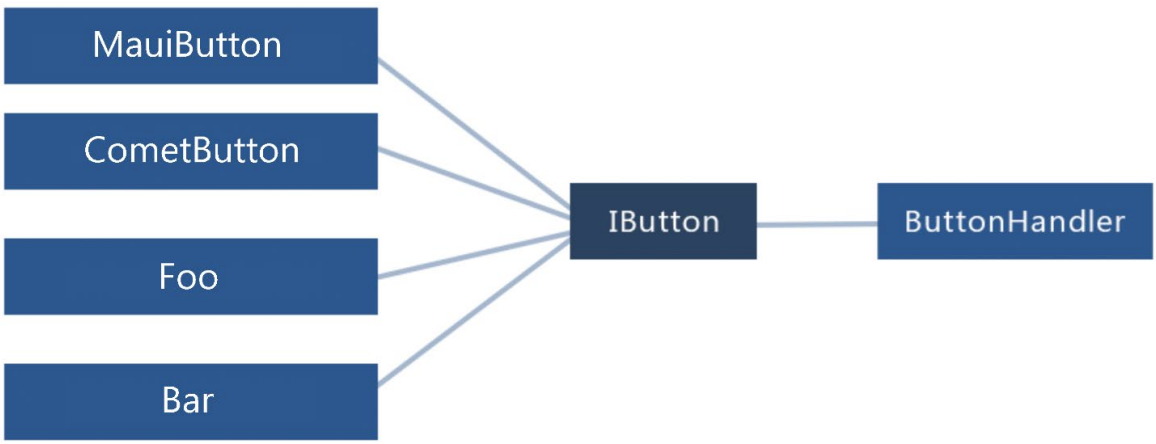
Aplikacije .NET Maui je mogoče implementirati na osebnem računalniku ali na Mac-u in prevesti v izvorne pakete aplikacij. Aplikacije, namenjene za Android, ki so implementirane z uporabo .NET Maui, se prevedejo iz jezika C# v vmesni jezik (IL), ki se nato ob zagonu aplikacije (JIT) prevede v izvorni sklop. Aplikacije, ki so namenjene za iOS in so

implementirane z uporabo ogrodja .NET Maui, se v celoti prevedejo najprej (AOT) iz C# v izvorno zbirko kodo ARM. Aplikacije, ki so zgrajene za macOS, uporabljajo Mac Catalyst. Namizne aplikacije za Windows, ki so implementirane z uporabo ogrodja .NET Maui pa uporabljajo knjižnico Windows UI 3 (WinUI 3).

Ogrodje .NET Maui ponuja zbirko kontrolorjev, ki se uporabljajo za prikazovanje podatkov in zbirk, označevanje dejavnosti in sprožitev dejanj. Poleg omenjene zbirke kontrolnikov .NET Maui ponuja tudi dodelan mehanizem, ki služi oblikovni postavitvi strani. Ponuja pa tudi podporo za vezavo podatkov, ki omogočajo bolj elegantne in vzdržljive razvojne vzorce. (What is .NET MAUI?, 2022)

Prvi korak pri preoblikovanju osnovne infrastrukture je, zagotoviti odstranitev trenutne tesne povezave s krmilniki. Pri .NET Maui so to dosegli tako, da jih je postavila za vmesnik in da so vse komponente delovale z le tem. Na ta način so poenostavili različne izvedbe, kot je Ibutton, hkrati pa so s tem zagotovili, da je osnovna infrastruktura še zmeraj vse te izvedbe obravnavala na enak način.

Slika prikazuje, kako zgoraj napisano zgleda iz konceptualne perspektive.



Slika 3 : Flutter - Infrastruktura posameznih komponent
Vir: (https://thewissen.io/images/posts/handlers.jpg)

Ekipa .NET Maui se je odločila, da bo spremenila način registracije upravljalnikov (ang. Handler). Namesto, da bi upravljalnike registrirali na ravni sklopa prek atributov, so registrirani izključno s platformo. (Thewissen, 2022)

Pri .NET Maui so se odločili za pristop definiranja slovarja za preslikavo. Preslikovalnik je slovar, ki ima lastnosti definirane v vmesniku določenega kontrolnika (ang. Controler), ki ponuja neposreden dostop do objekta izvirnega pogleda. Pretvorba omogoča takojšen dostop do kode, ki je specifična za platformo, iz kode v skupni rabi. (Basu, 2021)

2.2.1 Programski jezik C#

Programski jezik C# je sodoben, objektno in komponentno usmerjen jezik, ki je bil leta 2001 razvit in uveden s strani Microsofta.
Programski jezik C# je strogo tipiziran, objektno usmerjen programski jezik. Je odprtokoden, preprost, prilagodljiv in vsestranski. Izogiba se zapletenosti in nestrukturiranim jezikovnim funkcijam. Koda se prevede in se izvaja na virtualni napravi (ang. Virtual maschine).

Razvijalcem omogoča izdelavo številnih vrst varnih in robustnih aplikacij, ki delujejo v .NET. Korenine ima v družini jezikov C. Ponuja jezikovne konstrukcije za neposredno podporo tem konceptom, zaradi česar je naraven jezik za ustvarjanje in uporabo programskih komponent.

Ključne značilnosti jezika C# vključujejo:

- je moderen in enostaven,
- hiter in odprtokoden,
- uporabljen na različnih platformah,
- varen,
- vsestranski.

Namen programskega jezika C# je bil, razviti programski jezik, ki ni samo enostaven za učenje, ampak ima tudi podporo sodobne funkcionalnosti za vse vrste razvojev programske opreme. (Microsoft, 2022)

Bil je zasnovan tako, da upošteva poslovne potrebe. Zasnovan je za podjetja, in sicer za izdelavo vseh vrst programske opreme, kjer bi se lahko uporabil samo en programski jezik.

Podpira potrebe po razvoju mobilnih naprav, aplikacij (tudi spletnih) in storitev. Nekatere funkcije sodobnega programskega jezika, ki jih podpira, so razredi, samodejna inicializacija tipov in zbirk, lambda izrazi, dinamično programiranje, asinhrono programiranje, tuple, ujemanje vzorcev, napredno odpravljanje napak in obravnavanje izjem in še mnogo več.

Funkcije C# pomagajo ustvariti robustne in trajnostne aplikacije. Nullable tipi ščitijo pred spremenljivkami, ki se ne nanašajo na dodeljene objekte. Lambda izrazi pa podpirajo tehnike funkcionalnega programiranja. Sintaksa integrirane poizvedbe (LINQ) ustvarja skupni vzorec za delo s podatki iz virov. Jezikovna podpora za asinhrone operacije zagotavlja sintakso za gradnjo sistemov, ki so porazdeljeni.

Omogoča dinamično dodeljevanje objektov in shranjevanje lahkih struktur v vrstici.

Prav tako pa podpira generične metode in tipe, ki zagotavljajo večjo varnost tipov in zmogljivost. Ponuja iteratorje, ki izvajalcem razredov zbirk omogočajo definiranje vedenja po meri za kodo odjemalca. (A tour of the C# language, brez datuma)

2.2.2 Posebnosti ogrodja .NET Maui

Microsoft je spremenil izvedbe številnih kontrolnikov v novi različici .NET Maui. Ena izmed posebnosti ter izboljšav je t.i. BoxView, ki je osnovni pravokotnik z dano širino, višino ter barvo. Lahko se uporablja za okrasitev BoxViewa, osnovno grafiko in interakcijo z uporabnikom na dotik. Posebnost je prav tako IndicatorView, ki je kontrolnik, ki prikazuje indikacijo v CarrouselView, ki odražajo število elementov in trenutni položaj.

ImageButton ima možnost kombiniranja pogleda Button in Image. Če želite programu ukazati, naj izvede določeno dejavnost, uporabnik pritisne ImageButton.

Ena izmed posebnosti ogrodja .NET Maui je Hot Reload, ki je nova funkcija, ki omogoča urejanje izvorne kode aplikacije. Posebnost ogrodja je tudi BlazorWebView, ki omogoča gostovanje spletne aplikacije Blazor neposredno v aplikaciji. Zagotavlja izkoriščanje funkcionalnosti izvorne platforme in kontrolo uporabniškega vmesnika. Ogrodje .NET Maui ima možnost implementacije statičnega pozdravnega zaslona. Kontrolnik .NET Maui Scheduler nudi devet vgrajenih pogledov za prikaz datumov, ki se lahko uporabijo za preprosto načrtovanje dogodkov. (.NET MAUI is HERE! 3 NEW Features that will blow your mind, 2021)

2.2.3 Prednosti in slabosti ogrodja

Prednosti

Pri delu z ogrodjem .NET Maui je delo s slikami in velikostjo le teh zelo dobrodošel dodatek. MauiImage temelji na Resizetier.NT, ki uporabnikom zagotavlja stabilen in enostaven način za delo s slikami na različnih platformah.

Z razširitvijo ConfigureLifecycleEvents se lahko premaknejo vse inicializacijske kode aplikacije, ki so specifične za platformo, na eno mesto znotraj razreda MauiProgram.

Prednost uporabe .NET Maui je zagotovo poenotenje knjižnic. Le to zagotavlja številne prednosti z združitvijo knjižnice Xamarin.Essentials v .NET Maui, tako, da se lahko enostavno uporabljajo zmogljivosti naprave, kot so senzorji, naprave, fotografije, stiki in številne storitve, kot so preverjanje pristnosti in varno shranjevanje.

Slabosti

Uporabniki .NET Maui se lahko srečajo z nekaj težavami s samim osnovnim ogrodjem. Večina teh težav je bila sicer že prijavljenih in se jih odpravlja.

Veliko paketov, ki jih je bilo mogoče uporabiti pri Xamarinu, še ni razvitih za .NET Maui. Tudi obstoječi gradniki, kontrolniki in upravljavci imajo veliko napak. Pogosto moramo iskati »out of box« rešitev.

Uporabniki so opazili tudi, da se CollectionView obnaša zelo neenakomerno. Omiki Padding in Margin v vsebniku Frame ali Grid znotraj DataTemplate ne delujejo. To se lahko reši tako, da se oblazinjeni rob premakne iz predmeta vsebnika na podrejene elemente, tako da se določi njihove robove. Vendar se tudi ta ponastavitev v CollectionViewu pokvari pri drsenju gor in dol. (Akram, 2022)

2.3 Dostop do učnega gradiva

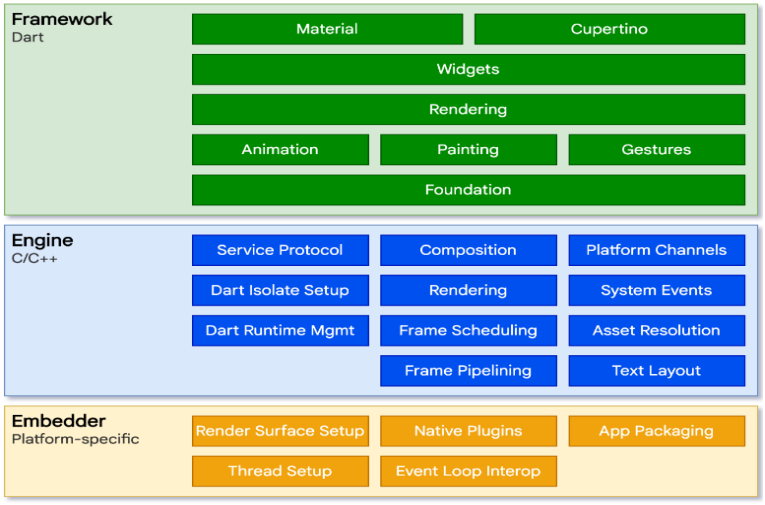
Na spletu je več dostopnega gradiva za ogrodje Flutter, saj je dalj časa na voljo uporabnikom. Prav tako je za Flutter dostopnih več vmesnikov, saj za .NET Maui šele prehajajo iz ogrodja Xmarin, kar posledično pomeni, da še veliko vmesnikov za .NET Maui ni optimiziranih. Zaradi novejše narave .NET Maui je posledično manj učnega gradiva in primerov uporabe. Obe ogrodji, tako Flutter kakor tudi .NET Maui, pa imata svoje dokumentacije (ang. Docs). Prav tako je implementacija enake aplikacije v .NET Maui veliko zahtevnejša, kakor v ogrodju Flutter. Tudi programski jezik c# je bolj uporabljen in znan kot programski jezik Dart, ki se uporablja v Flutter-ju.

2.4 Primerjava

Flutter 3 je prinesel stabilno podporo za aplikacije macOS in Linux za dokončanje njihovih zmogljivostih na platformah iOS, Android, splet in Windows. NET Maui pa prinaša zmogljiv jezik C# v prostor, ki omogoča razvoj v sistemih Windows, iOS, Android in macOS. Razvoj programske opreme na več platformah omogoča lažjo in hitro izdelavo prototipov.

Flutter je ogrodje za več platform. Razvit je bil s strani Googla, ki ga je razvil maja 2017. Zgrajen je s programskim jezikom Dart, ki temelji na jeziku C, prav tako razvit s strani Googla.

Kako deluje:



Slika 4 : Flutter - Diagram delovanja

Vir: (https://www.altexsoft.com/media/2018/11/Flutter-architecture.png)

Flutter za upodabljanje grafike uporablja Skia. To ogrodju omogoča pisanje slikovnih pik neposredno na zaslon, namesto da bi sodelovalo z izvornimi API-ji in zaradi tega je upodabljanje grafike neodvisno od platforme. Zaradi tega je Flutter bolj odporen na spremembe API-jev izvornega operacijskega sistema na nižji ravni. Slaba stran tega pristopa se zaznava na spletu. Ker Flutter piše uporabniške vmesnike slikovno piko za slikovno piko, spletna izvedba zapiše te slikovne pike v en sam element platna. Zaradi tega je težje izvajati nekatera bolj standardna dejanja spletnega brskalnika, kot so npr. desni kliki, iskanje s tipko Control-f itd.

Prednosti Flutterja:

- je najhitrejše rastoče ogrodje za več platform;
- uporabniški vmesnik upodablja neodvisno od platforme, na kateri deluje, zaradi česar je bolj odporen na spremembe izvornih API-jev;
- je lažje prenosljiv na pametne naprave, kot so npr. ure, televizorji in druge nosljive tehnologije.

Slabosti:

- nabor razpoložljivih razvijalcev za Dart je majhen, a na srečo se tega jezika ni težko naučiti;
- spletna podpora je omejena na eno platno.

.NET Maui je razvoj Xamarin.Forms, ogrodja mobilnih aplikacij za več platform, ki je bil namenjen za iOS, Android in Windows. Vključen je v .NET 6 in se oglašuje kot primarni razvojni komplet uporabniškega vmesnika za .NET 6 v prihodnosti. Kako deluje, sem prikazal že s sliko številka 2 zgoraj.

Maui deluje podobno kot druga ogrodja za več platform, kot sta Ionic/Capasitor.js ali React Native. Ta ogrodja abstrahirajo izvorne kontrole vsake ciljne naprave, kot so gumbi, potrditvena polja, dostop do kamere itd. To doseže ogrodje .NET, specifično za platformo, za Android, iOS in macOS, izvaja pa ga okolje Mono Runtime. NET Maui ne prinaša veliko novosti na medplatformsko prizorišče.

Prednosti:

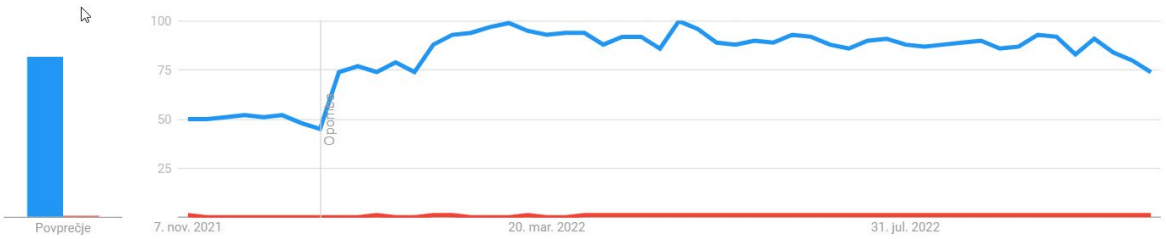
- za kodiranje na več platformah je na voljo velik nabor razvijalcev C#;
- izkorišča velik ekosistem .NET za spodbujanje ponovne uporabe kode.

Slabosti:

- .NET Maui v osnovi ne podpira razvoja spletnih aplikacij, ampak razvijalci lahko uporabijo okvir Blazor, ki zagotovijo most za uporabo virov Maui;
- .NET Maui ne podpira Linuxa;
- je še v razvoju.

.NET Maui ima nekaj očitnih pomanjkljivosti in mora še veliko nadoknaditi, da doseže zrelost Flutterja. Google in Flutter močno napredujeta pri medplatformskih tehnologijah, vendar ima tudi Microsoft počasno in stabilno rast, sploh s strani uporabnikov Xamarina, ki čakajo na prehod na .NET Maui. Definitivno se je zato lažje priučiti Flutterja, ki ima vse funkcije trenutno poenostavljene, ker delujejo. Za vsako funkcionalnost obstaja funkcija ali vmesnik. Zato menim, da se je začetniku lažje priučiti ogrodja Flutter.

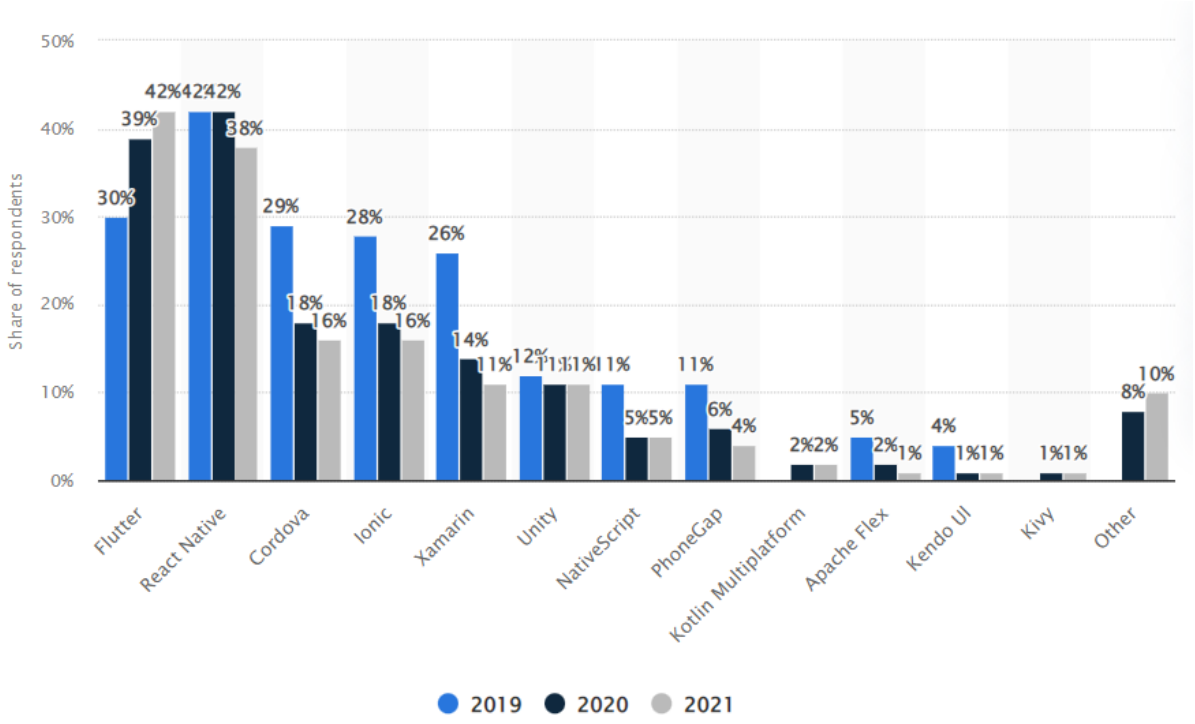
S pomočjo Google Trenda(https://trends.google.com/trends/explore?q=flutter,.net%20maui), sem poiskal, v kakšnem trendu je iskanje po besednih frazah flutter in .NET Maui. Z rdečo je prikazano iskanje po ogrodju .NET Maui in z modro iskanje po ogrodju Flutter.



Slika 5 : Trend iskanja po ogrodjih

Vir: (https://trends.google.com/trends/explore?q=flutter,.net%20maui)

Flutter je veliko bolj iskan in tudi trend ostaja konstanten. Pri pregledu statistike na spletni strani Statiste(vir: https://www.statista.com/statistics/869224/worldwide-software-developer-working-hours/) sem naletel na zadnjo statistično raziskavo priljubljenih ogrodij za izdelavo cross-platform aplikacij (Slika 6). Ker je raziskava izdana 2022, so podatki do 2021, ampak lahko sklepamo, da ogrodje .NET Maui še ni blizu tolikšne skupnosti kot jo ima ogrodje Flutter, ki je najbolj uporabljeno ogrodje za razvijalce.



Slika 6 : Statistični prikaz priljubljenosti ogrodja pri razvijalcih

Vir: (https://trends.google.com/trends/explore?q=flutter,.net%20maui)

Pri stroškovni primerjavi sem primerjal cene razvoja majhne aplikacije s funkcionalnostmi, ki sem jih potreboval sam za aplikacijo Divja odlagališča. Za vsako funkcionalnost je potrebno določeno število ur razvoja in testiranja. Predvideval sem, ker sem sam porabil 150 ur za aplikacijo v ogrodju .NET Maui in 100 ur za aplikacijo v ogrodju Flutter, da bi izkušen razvijalec potreboval 100 in 75 ur za vsako aplikacijo. Urne postavke so tudi različne glede na ogrodje. Upošteval sem, da so vmesniki vsi brezplačni in da vsako ogrodje uporablja svoje razvojno okolje.

Tabela 1 : Stroškovna primerjava razvoja aplikacije

	.NET Maui	Flutter
Čas razvoja(h)	100	75
Cena razvijalca na uro(€)	55	65
Cena IDE-ja za čas razvoja(€)	45	0
Skupno	5.545 €	4.874 €

Vir: (Lastni vir)

Sklepam, da je razvoj majhne aplikacije stroškovno bolj ugoden v ogrodju Flutter. Čeprav so razvijalci, ki razvijajo v ogrodju Flutter dražji, se aplikacija lahko razvije v krajšem času, saj je ogrodje stabilno in podprto z vsemi vmesniki, ki olajšajo razvoj aplikacije. Med drugim je tudi cena razvojnega okolja pri uporabi Visual Studia vplivala na končen izid stroškovne primerjave.

Po raziskavi menim, da se je možno naučiti obeh jezikov in izpeljati razvoj aplikacij sam, brez razvijalcev in brez uporabe Visual Studia. Tako bi stroškovno prišli na enako ceno z obema ogrodjema, ampak menim, da je zaradi časovne razlike razvoja, ki bi se še povečala brez uporabe Visual Studia, bolj smiselno uporabiti ogrodje Flutter.

3 MOBILNA APLIKACIJA DIVJA ODLAGALIŠČA

Divja odlagališča

Divje odlagališča so mesto v naravi, kjer ljudje neprimerno odlagajo razne vrste odpadkov, ki povzročajo okoljsko škodo. Kemikalije in biološko nerazgradljivi materiali v odpadkih vplivajo na fizično okolje in vodne poti z onesnaženjem podtalnice in tal, uničujejo življenjski prostor živim bitjem in onesnažujejo zrak. Odpadki lahko širijo tudi plevel in škodljivce ter tako vplivajo na kmetijstvo, divje živali in celotno okolje. Divje in udomačene živali lahko poginejo tudi po zaužitju strupenih materialov, kot so plastika in kemikalije iz odpadkov. Prav tako pa ti odpadki predstavljajo resno tveganje za zdravje in varnost otrok, ki jih lahko nabirajo, jedo ali se igrajo z njimi.

Nič nenavadnega ni več videti ali najti ogromnih kupov odpadkov, starih hišnih predmetov, gradbenih odpadkov ali odslužene programske opreme in drugih rabljenih izdelkov, odvrženih ob cestah, v gozdovih ali na neurejenih odlagališčih. Ta trend je v porastu in vse bolj postaja težava, saj ovira cilj ohranjanja čistega okolja in resno vpliva na divje živali in habitat. Po podatkih agencij in oddelkov za varstvo okolja se ta zloraba imenuje nezakonito odlaganje. Šteje se za nezakonito zaradi škode, ki jo povzroča okolju.

Za rešitev problema divjih odlagališč obstajajo uradno določena območja z ustrezno integriranimi sistemi ravnanja z odpadki. Zato je v nasprotju z zakonom, da bi odmetavali smeti kamorkoli, na primer ob cesti ali v gozdu.

Študije po svetu so pokazale, da je višja stopnja proizvodnje odpadkov izrazito povezana s povečanimi stopnjami nezakonitega odlaganja odpadkov in naraščajočim se številom divjih odlagališč. Povečanje celotne proizvodnje odpadkov ustreza povečanemu bogastvu, hitri spremembi preferenc in vedno večjemu svetovnemu prebivalstvu. Poleg tega postajajo ljudje vedno bolj potrošniško usmerjeni kot v preteklosti. V sodobnem svetu večina ljudi išče nove in boljše izdelke, predvsem gospodinjske aparate, oblačila in elektroniko. Odmetavanje starih naprav in rabljenih gospodinjskih aparatov je tako povzročilo porast divjega odlaganja odpadkov. Rast svetovnega prebivalstva je eden od razlogov za nezakonito odlaganje smeti. Ker je skupna raven proizvodnje odpadkov v pozitivni korelaciji s številom prebivalstva na

zemlji, povečanje števila prebivalstva samodejno pomeni povečanje količine skupnih odpadkov.

Večina ljudi se dobro zaveda posledic, ki jih prinaša nezakonito odlaganje odpadkov na divjih odlagališčih. Ne glede na to, pa nekateri posamezniki preprosto ne vidijo potrebe po recikliranju odpadkov.

Pravzaprav je večino predmetov, ki jih ljudje nezakonito odvržejo, mogoče reciklirati ali celo ponovno uporabiti, to so stari aparati, bela tehnika in pohištvo, računalniška oprema itd.

Nezakonito odlaganje gospodinjskih odpadkov, kemikalij, gum, rabljenih avtomobilskih delov, nevarnih medicinskih odpadkov in zelenih odpadkov lahko poveča tveganje požarov v naravi. Prav tako lahko požar povzročijo tudi cigaretni ogorki. (Illegal Dumping: Causes, Effects and Solutions to Huge Piles of Wastes, brez datuma)

Aplikacija Android

Aplikacija za Android je programska aplikacija, ki se izvaja na platformi Android. Ker je Android platforma ustvarjena za mobilne naprave, je tipična aplikacija za Android zasnovana za pametni telefon ali tablični računalnik z operacijskim sistemom Android.(vir 11)

Izraz pa se nanaša tudi na datoteko APK, ki pomeni Android paket. Ta datoteka je arhiv Zip, ki vsebuje kodo aplikacije, vire in metainformacije.

Aplikacije, namenjene za Android, so lahko napisane v programskem jeziku Kotlin, Java ali C++ in se izvajajo znotraj Virtual Machine. Uradno razvojno okolje je Android Studio. Najprej so prevedeni v izvedljive datoteke Dalvik za izvajanje na virtualnem stroju Dalvik, ki je zasnovan posebej za mobilne naprave.

Razvijalci lahko prenesejo komplet za razvoj programske opreme Android (SDK) s spletnega mesta Android. SDK vključuje orodja, vzorčno kodo in ustrezne dokumente za ustvarjanje aplikacij Android.

Razvijalci začetniki, ki se želijo le poigrati s programiranjem za Android, lahko uporabijo App Inventor. S to spletno aplikacijo lahko uporabniki sestavijo aplikacijo za Android, kot da bi sestavljali koščke sestavljanke.

Aplikacije za Android so na voljo v trgovini Google Play in na več spletnih mestih, osredotočenih na aplikacije za Android. Takšne aplikacije so razvite za različne namene. (vir 12)

Primerjava Android aplikacije z iOS aplikacijo

Applov operacijski sistem iOS se popolnoma razlikuje od operacijskega sistema Android. Poleg tega je razvit v drugem programskem jeziku. iOS ima tudi poseben oblikovalski jezik, ki se običajno uporablja za aplikacije.

Večina razvijalcev se nagiba k lansiranju podobnih aplikacij za obe platformi. Za primerjavo, drugi imajo različne aplikacije za obe platformi.

3.1 Analiza in specifikacije mobilne aplikacije

Pri zasnovi aplikacije sem pregledal trenutne aplikacije, ki so na voljo v trgovini Play in Apple store. Želel sem ugotoviti, katere funkcionalnosti so osnovne in potrebne za aplikacijo, ki se posveča problematiki divjih odlagališč.

Na to tematiko nisem našel podobne aplikacije niti v angleščini, zato sem moral specifikacije aplikacije postaviti sam. Zamislil sem si mobilno aplikacijo, ki uporablja aventikacijo, da bi kar se da preprečil neresne uporabnike. Nato bi prikazal zemljevid na lokaciji uporabnika. Omogočil bi funkcijo slikanja divjega odlagališča in nato, če je uporabnik zadovoljen s sliko, da jo lahko tudi naloži. Nakar se doda marker na zemljevidu, ki označuje divje odlagališče. Dodal bi tudi funkcionalnost, da se ob kliku na marker odpre slika odlagališča, saj bi ga tako bilo lažje locirati ob morebitnih čistilnih akcijah.

Kasneje bi bilo pametno dodati admin funkcijo, da bi vsako odlagališče preveril admin, da ne bi prišlo do nepotrebnih lažnih odlagališč.

Uporabil bom vsa orodja in vtičnike, ki so na voljo brezplačno na internetu oz. že vgrajena v ogrodjih. Tako bom lažje prišel do končnega, podobnega rezultata pri obeh aplikacijah. Pri avtentikaciji in zbirki podatkov bom uporabil Googlove vtičnike Firebase, za ostale stvari pa že obstoječe rešitve posameznega ogrodja.

3.2 Razvojno okolje

Razvojno okolje (ang. IDE) je aplikacija, ki združuje osnovna in napredna orodja v enoten grafični vmesnik (ang. GUI) in tako omogoča učinkovit razvoj novih aplikacij. Razvojno okolje je zbor orodij za pisanje, dizajn, razvoj, sestavo, testiranje in razhroščevanje (ang. Debugging) v eni sami aplikaciji. S kvalitetnimi razvojnimi okolji, zmanjšamo cikel razvoja in zagotovimo večjo kvaliteto razvitih aplikacij.

Razvojno okolje ponavadi sestavlja:

- **Urejevalnik izvorne kode**: Je enostaven urejevalnik besedila. Omogoča pisanje kode z označevanjem sintakse, možnost samodejnega dokončevanja ukaza (ang. Auto-completion) in avtomatično preverjanje napak.
- **Orodje za gradnjo aplikacije**: Gradnja programa (ang. Software build) in sorodnih procesov z avtomatizacijo ponavljajočih se nalog, kot je prevajanje računalniške izvorne kode v binarno kodo, pakiranje kode in izvajanje samodejnih testov. Ne gradimo vsakič celotne aplikacije.
- **Razhroščevalnik**: Je orodje, ki testira in razhrošča aplikacijo. Napako najde v izvorni kodi in jo grafično prikaže.
- **VCS orodja in vtičniki**: Razvojno okolje se lahko uporablja tudi za nadzor različic. VCS (ang. Version control systems) so programska orodja, ki jih razvijalci uporabljajo za upravljanje sprememb v izvorni kodi skozi čas, kar omogoča hitrejše in pametnejše delo. Najbolj uporabljen primer je recimo Git.

Za razvoj mobilnih aplikacij je na voljo veliko razvojnih okolij, a potrebno je izbrati ustreznega, ki omogoča vse funkcionalnosti, ki jih potrebujemo, saj so nekatera razvojna okolja specifično namenjena za razvoj enega tipa aplikacij. (India, 2019)

Ko gre za orodja, ki jih uporabljamo za razvoj aplikacij Flutter, sta dva glavna kandidata Visual Studio Code (https://visualstudio.microsoft.com/) in Android Studio (https://developer.android.com/studio) .
Dan danes se moremo odločiti, ne samo v zvezi s tem, katere jezike in okvire uporabljamo za izdelavo aplikacij, ampak tudi glede ogrodij, ki jih uporabljamo pri tem.

3.2.1 Visual Studio Code

Visual Studio Code je brezplačen, lahek, a zmogljiv urejevalnik izvorne kode, ki se izvaja na namizju in v spletu ter je na voljo za Windows, macOS, Linux in Raspberry Pi OS. Visual Studio Code ima vgrajeno podporo za JavaScript, TypeScript in Node.js ter ima bogat ekosistem razširitev za druge programske jezike (kot so C++, C#, Java, Python, PHP in Go), izvajalne čase (kot je .NET in Unity), okolja (kot sta Docker in Kubernetes) in oblake (kot so Amazon Web Services, Microsoft Azure in Google Cloud Platform). (Microsoft, Visual Studio, 2022)

Poleg celotne ideje, da je Visual Studio Code lahkoten, ki hitro zažene, ima tudi dokončanje kode IntelliSense za spremenljivke, metode in uvožene module; grafično odpravljanje napak; linting, urejanje z več kazalci, namigi za parametre in druge zmogljive funkcije urejanja; hitro krmarjenje in preoblikovanje kode; in vgrajen nadzor izvorne kode, vključno s podporo za Git. Velik del tega je bil prilagojen iz tehnologije Visual Studio.

Visual Studio Code je zgrajen z uporabo ang. Electron Shell, Node.js, TypeScript in protokola jezikovnega strežnika in se posodablja vsak mesec. Bogastvo podpore se razlikuje glede na različne programske jezike in njihove razširitve, od preprostega označevanja sintakse in ujemanja oklepajev do odpravljanja napak in refaktoriranja. Če jezikovni strežnik ni na voljo, lahko dodate osnovno podporo za svoj najljubši jezik prek Colorizers TextMate.

Koda Visual Studio je odprtokodna pod licenco MIT. Sam izdelek Visual Studio Code je dobavljen pod standardno Microsoftovo licenco za izdelek, saj ima majhen odstotek Microsoftovih specifičnih prilagoditev. Kljub komercialni licenci je brezplačen.

Torej, poleg omenjenih ogrodij in izvajalnih okolij Visual Studio Code uporablja Chromium, ki je odprtokodni projekt brskalnika, katerega cilj je zgraditi varnejši, hitrejši in stabilnejši način za vse uporabnike interneta, da izkusijo splet. In V8, ki je Googlov odprtokodni, visoko zmogljiv motor JavaScript in WebAssembly, napisan v C++. (Heller, 2022)

3.2.2 Android studio

Android Studio je uradno integrirano razvojno okolje (IDE) za razvoj aplikacij za Android. Android Studio ponuja več funkcij, ki povečujejo našo produktivnost med ustvarjanjem aplikacij za Android. Temelji na IntelliJ IDEA, integriranem razvojnem okolju Java za programsko opremo, in vključuje njegova orodja za urejanje kode in razvijalce.

Android Studio je bil objavljen 16. maja 2013 na konferenci Google I/O kot uradni IDE za razvoj aplikacij za Android. Prva stabilna zgrajena različica je bila izdana decembra 2014. Od 7. maja 2019 je Kotlin Googlov prednostni jezik za razvoj aplikacij za Android. Poleg tega Android Studio podpira tudi druge programske jezike.

Za podporo razvoju Androidovih aplikacij Android Studio uporablja gradbeni sistem, ki temelji na Gradle-u, emulatorju, kodnih predlogah, in integracijo Github-a.

Android Studio je na voljo za namizne operacijske sisteme Mac, Windows in Linux. Zamenjal je Eclipse Android Development Tools (ADT) kot primarni IDE za razvoj aplikacij za Android. Android Studio in komplet za razvoj programske opreme lahko prenesemo neposredno iz Googla.

Android Studio uporablja funkcijo Instant Push za potiskanje kode in spremembe virov v delujoči aplikaciji. Urejevalnik kode pomaga razvijalcu pri pisanju kode in ponuja dokončanje kode, lomljenje in analizo. Aplikacije, izdelane v Android Studiu, se nato prevedejo v format APK za pošiljanje v trgovino Google Play. (Contributor, brez datuma)

Funkcije Android Studia:

- Ima prilagodljiv gradbeni sistem, ki temelji na Gradlu;
- Ima hiter in s funkcijami bogat emulator za testiranje aplikacij;
- Android Studio ima konsolidirano okolje, kjer lahko razvijamo za vse naprave Android.
- Uporabite spremembe kode vira naše delujoče aplikacije brez ponovnega zagona aplikacije;
- Android Studio ponuja obsežna orodja in ogrodja za testiranje;
- Podpira C++ in NDK;
- Zagotavlja vgrajeno podporo za Google Cloud Platform;

- Omogoča enostavno integracijo Google Cloud Messaging in App Engine.

(Android Studio, brez datuma)

3.2.3 Primerjava Visual Studio in Android studio

Odgovor na vprašanje, ali je za razvijalce boljši Visual studio ali Android studio, je odvisen od tega, katero platformo uporabljamo. Visual Studio ima širšo odobritev s strani razvijalcev za razliko Android Studia, saj ga omenja 676 skupin razvijalcev in podjetij. To pomeni, da bodo razvijalci, ki uporabljajo Visual Studio, bolj verjetno delali na aplikacijah za mobilno platformo. Visual Studio je najpogosteje uporabljen IDE za razvoj aplikacij za Android in je priljubljena izbira med razvijalci.

Prednost Android Studia je ta, da ga priporoča Google, zato zmeraj prvi prejme posodobitve. IDE uporablja tudi Gradle, Googlovo novo uradno orodje za gradnjo, ki nadomešča Apache Ant. Gradle, je izjemno zmogljiv in omogoča enostavno nadgradnjo vašega gradbenega sistema. Omogoča tudi razlikovanje med razvojno in proizvodno različico. In seveda je lažji za uporabo kot Visual Studio, zato je najbolj priljubljena možnost med razvijalci za Android.

Ko se ukvarjamo z izdelavo mobilnih aplikacij, je Android Studio najboljša izbira. Ne samo, da ponuja stabilno okolje za razvoj aplikacij za Android, ima tudi številne funkcije, zaradi katerih je odlična izbira za razvijalce. Android Studio ponuja prilagajanje in razširljivosti, hkrati pa je hitrejši in lažji za učenje.

Visual Studio je IDE za razvoj Androida in zahteva vrhunski procesor in RAM. Ko ga prenesemo in namestimo, bomo pozvani, da izberemo Workload. Ta izbira bo določila, s katero platformo delamo.

Android Studio in Visual Studio lahko v prvi vrsti uvrstimo med orodja »Integrirano razvojno okolje«.

Glede na skupnost StackShare ima Visual Studio širšo odobritev, saj je omenjen v skladih 676 podjetij in 1009 skladih razvijalcev; v primerjavi z Android Studio, ki je naveden v 928 skladih podjetij in 690 skladih za razvijalce.

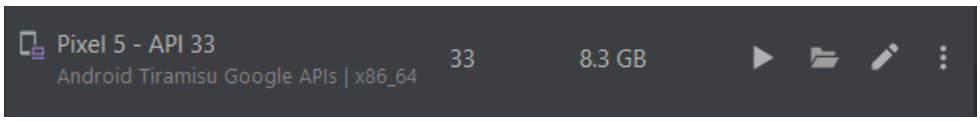
Tabela 2 : Število uporabnikov razvojnega okolja

	Android Studio	Visual Studio
V skladu podjetjih	928	676
V skladu razvijalcev	690	1009
Skupaj	1618	1685

Vir:(https://stackshare.io/stackups/android-studio-vs-visual-studio)

3.3 Izdelava aplikacije z ogrodjem Flutter

Za implementacijo mobilne aplikacije v ogrodju Flutter sem izbral integrirano razvojno okolje (ang. IDE) Android Studio. Za začetek je bila potrebna namestitev Flutter SDK po navodilih s strani Googla. (vir: https://docs.flutter.dev/get-started/install/windows). Za izvajanje aplikacije sem izbral Googlov Pixel 5 androidni telefon, ki sem ga emuliral s pomočjo Device Manager-ja. Izbral sem solidne performanse in tudi najnovejši android SDK 33.



Slika 7 : Navidezna android naprava

Vir: (Lastni vir)

Nato sem na podlagi analize začel iskati primerne pakete (ang. Packages), ki bi pomagali pri želeni funkcionalnosti, torej manipulacija z zbirko podatkov, uporaba kamere, zemljevidov, prikaz obvestil ipd. Vse pakete lahko enostavno prenesemo in namestimo z ukazom v terminalu:

```
flutter pub add<ime-paketa>
```

Tako sem prenesel in namestil sledeče pakete:

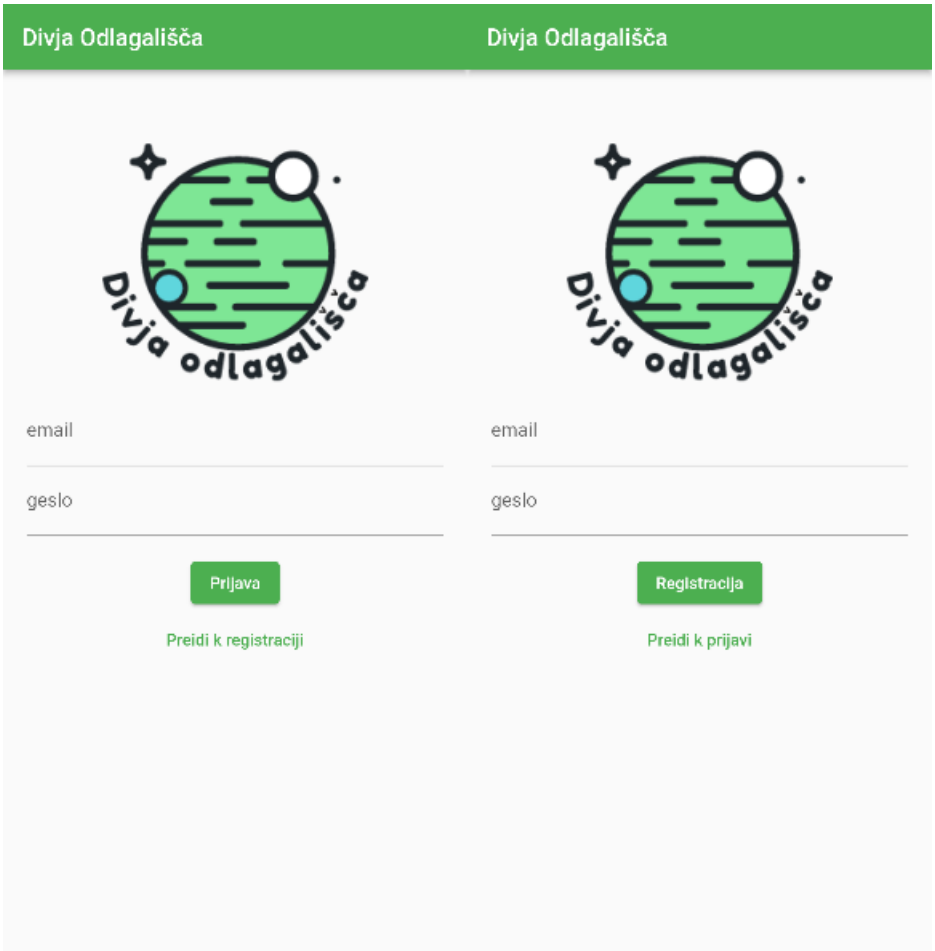
```
cupertino_icons: ^1.0.2
google_maps_flutter: ^2.2.1
firebase_auth: ^3.11.1
provider: ^6.0.3
geolocator: ^9.0.2
geocoding: ^2.0.5
camera: ^0.10.0+3
path_provider: ^2.0.11
path: ^1.8.2
image_picker: ^0.8.6
firebase_storage: ^10.3.11
firebase_database: ^9.1.7
firebase_core: ^1.24.0
fluttertoast: ^8.1.1
```

Slika 8 : Izbrani Flutter paketi

Vir: (Lastni vir)

Cupertino icons je paket, ki se uporablja za enostavno uporabo ikon in slik, za gumbe. Google maps Flutter sem uporabil za prikaz zemljevida. Tu so mi pomagali tudi Geolocator, Geocoding in Path, ki pomagajo pri pridobitvi in izračunu uporabnikove lokacije. Camera in Image picker sta paketa, ki aktivirata kamero in pomagata pri deljenju nastale slike. Firebase paketi so bili uporabljeni za avtorizacijo in podatkovno bazo. FlutterToast je prišel v poštev, kot enostaven prikaz obvestil, ko se je uporabnik zmotil pri uporabi aplikacije.

Pri vhodnem oknu sem uporabil enostaven dizajn z našim logotipom, dva odstavka za vpis informacij, ElevatedButton za potrditev in TextButton za spremembo namembnosti okna.



Slika 9 : Aplikacija v ogrodju Flutter login/register okno

Vir: (Lastni vir)

Ob kliku na gumb Prijava ali Registracija se izvede funkcije glede na stanje okna. Ta poskrbi za klic na Firebase Realtime Database in potrdi da uporabnik obstaja oz. v primeru registracije uporabnika ustvari, če so vneseni podatki pravilni.

```
Future<void> signInWithEmailAndPassword() async {
  try {
    await Auth().signInWithEmailAndPassword(
      email: _controllerEmail.text,
      password: _controllerPassword.text,
    );
  } on FirebaseAuthException catch (e) {
    setState(() {
      errorMessage = e.message;
    });
  }
}

Future<void> createUserWithEmailAndPassword() async {
  try {
    await Auth().createUserWithEmailAndPassword(
      email: _controllerEmail.text,
      password: _controllerPassword.text,
    );
  } on FirebaseAuthException catch (e) {
    setState(() {
      errorMessage = e.message;
    });
  }
}
```

Slika 10 : Aplikacija v ogrodju Flutter - funkciji za vpis/registracijo

Vir: (Lastni vir)

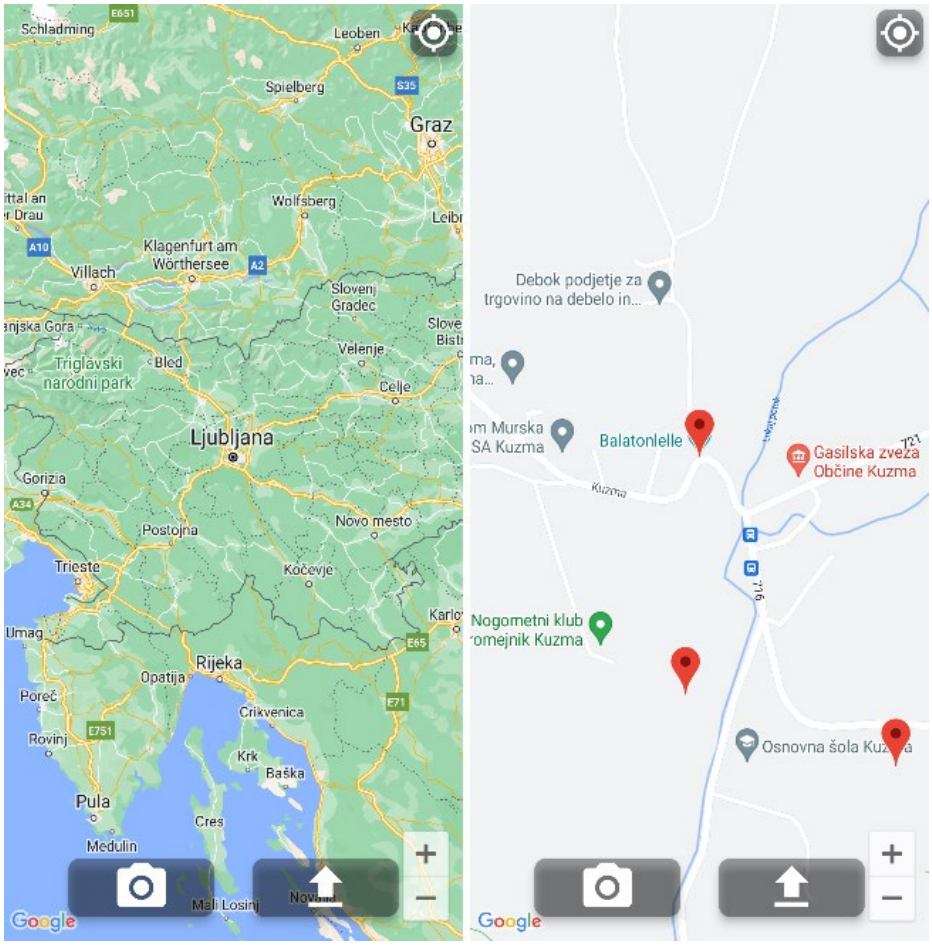
Na začetku je spremenljivka isLogin (tipa bool) nastavljena na true in nato s klikom na TextButton to stanje po želji spreminjamo. S spremembo vrednosti se spremeni tudi celoten vmesnik.

```
Widget _submitButton() {
  return ElevatedButton(
    onPressed:
    isLogin ? signInWithEmailAndPassword : createUserWithEmailAndPassword,
    child: Text(isLogin ? 'Prijava' : 'Registracija'),
  );  // ElevatedButton
}
```

Slika 11 : Aplikacija v ogrodju Flutter - potrditveni gumb

Vir: (Lastni vir)

Po prijavi se prikaže glavno okno, kjer je kot contex uporabljen Googlov zemljevid. Dodani so še gumbi (ElevatedButton) za zajem slike, potrditev odlagališča in lokacijski gumb. Po inicializaciji se pokaže zemljevid Slovenije, nato poizkuša pridobiti uporabnikovo lokacijo in zemljevid animira na to lokacijo.



Slika 12 : Aplikacija v ogrodju Flutter - glavno okno

Vir: (Lastni vir)

Ob kliku na gumb addPhoto se kliče funkcija pickMedia. Funkcija ob dovoljenju uporabnika odpre kamero, nakar uporabnik sliko potrdi. Slika se shrani in naloži na Firebase Storage server. Nato se slika in trenutna lokacija shranita v seznam tempMarkers s funkcijo _setTempMarker. Podatke ne naložimo takoj kot marker v zemljevid, ker bi si uporabnik lahko premislil in posnel drugačno sliko odlagališča ali bi aplikacijo le preverjal.

```
void pickMedia(ImageSource src) async {

  var file = await ImagePicker().pickImage(source: ImageSource.camera);
    if (file != null) {
      final MyRef = storage.ref(file.name);
      File uploadFile = File(file.path);
      await MyRef.putFile(uploadFile);
      String imageLink = await MyRef.getDownloadURL();
      _setTempMarker(imageLink);
    }
}
```

Slika 13 : Aplikacija v ogrodju Flutter - funkcija za fotografiranje

Vir: (Lastni vir)

Zato potrditev odlagališča imamo gumb addMarkerButton, ki služi nalaganju podatkov v marker in le tega v zemljevid, če je uporabnik prepričan, da so podatki ustrezni. Ob kliku na gumb se izvrši funkcija uploadMarker, ki preveri, če je seznam tempMarkers prazen, nakar opozori uporabnika s FlutterToast sporočilom. Če seznam ni prazen, si pridobi zadnji tempMarker, s pomočjo funkcije getLastTemp, ki prelista seznam in vrne zadnji objekt. Nato ta objekt naloži na Firebase Realtime Database, da lahko pri naslednji inicializaciji marker narišemo na zemljevid. Nato ga še dodamo v trenuten seznam actuallMarkers, da se marker nariše na zemljevid.

```
void uploadMarker() async {
  if (tempMarkers.isNotEmpty) {
    var temp = getLastTemp();
    await ref.push().set(temp.toMap());

    //set the marker
    setState(() {
      mymarkers.add(Marker(
        markerId: MarkerId(temp.imagelink),
        position: new LatLng(temp.lat, temp.lng),
        onTap: (){
          Navigator.push(
            context, MaterialPageRoute(builder:
              (context) => PhotoPage(imglink: temp.imagelink)),
          );
        },
        icon: BitmapDescriptor.defaultMarker, //Icon for Marker
      ));  // Marker
    });
  }
  else{
    Fluttertoast.showToast(
        msg: "Ste mogoče pozabili na sliko?",
```

Slika 14 : Aplikacija v ogrodju Flutter - funkcija za nalaganje markerja

Vir: (Lastni vir)

Ob vsaki inicializaciji in ob kliku na gumb za lokacijo se kliče funkcija _getCurrentPosition, ki ob dovoljenju uporabnika posodobi vrednosti currlat in currlng za trenutno lokacijo. Ob enem pa tudi pregleda Firebase Realtime Database, če obstajajo novi markerji, ki jih nato doda v seznam aktualnih markerjev. Zemljevid tudi animira na trenutno lokacijo, če je uspešno izvršena.

```
Future<void> _getCurrentPosition() async {
  final hasPermission = await _handleLocationPermission();
  final GoogleMapController controller = await _controller.future;
  if (!hasPermission) return;
  actuallMarkers.clear();
  actuallMarkers = await getMarkersFromFirebase();
  getMarkers();
  await Geolocator.getCurrentPosition(
      desiredAccuracy: LocationAccuracy.high)
      .then((Position position) {
    currlat = position.latitude;
    currlng = position.longitude;
    controller.animateCamera(CameraUpdate.newLatLngZoom(
        LatLng(position.latitude, position.longitude), 16));
  }).catchError((e) {
    debugPrint(e);
  });
}
```

Slika 15 : Aplikacija v ogrodju Flutter - funkcija za pridobitev lokacije

Vir: (Lastni vir)

Vsak marker ima tudi funkcijo onTap, ki ob kliku odpre novo okno in pokaže sliko selektiranega divjega odlagališča.

```
onTap: (){
  Navigator.push(
    context, MaterialPageRoute(builder:
      (context) => PhotoPage(imglink: temp.imagelink)),
  );
},
```

Slika 16 : Aplikacija v ogrodju Flutter - funkcija za odpiranje novega okna

Vir: (Lastni vir)

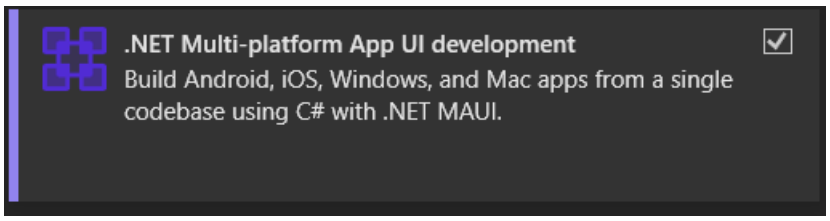
Slika 17 : Aplikacija v ogrodju Flutter - slika odlagališča

Vir: (Lastni vir)

3.4 Izdelava apliakcije z ogrodjem .NET Maui

Za izdelavo mobilne aplikacije v ogrodju .NET Maui sem se odločil uporabiti integrirano razvojno okolje Microsoft Visual Studio Professional, saj je edino podprto okolje za razvoj aplikacij v tem ogrodju.

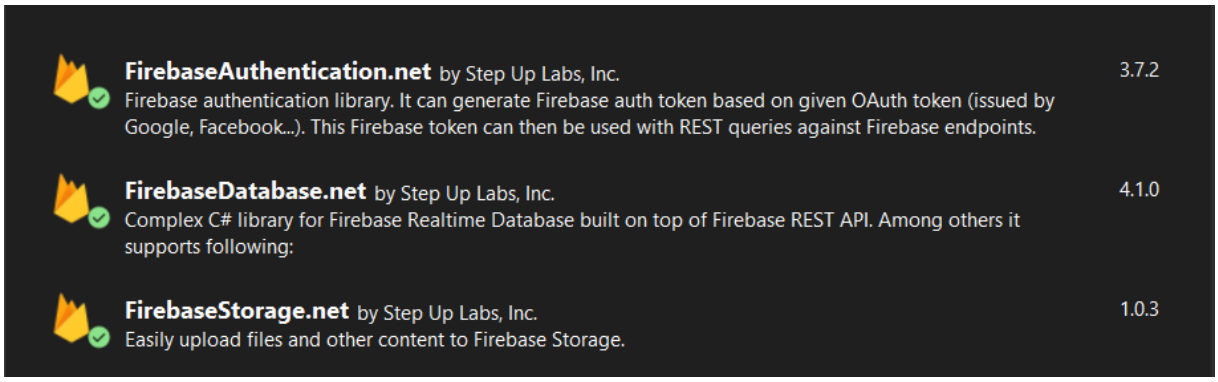
Razvojno okolje sem namestil po navodilih s strani Microsofta (vir: https://learn.microsoft.com/en-us/visualstudio/install/install-visual-studio?view=vs-2022). Nakar sem izbral namestitev .NET Multi-Platform App UI developement, ki je kritičen za razvoj mobilnih aplikacij v ogrodju .NET Maui.



Slika 18: Izbrano orodje v Visual Studiu

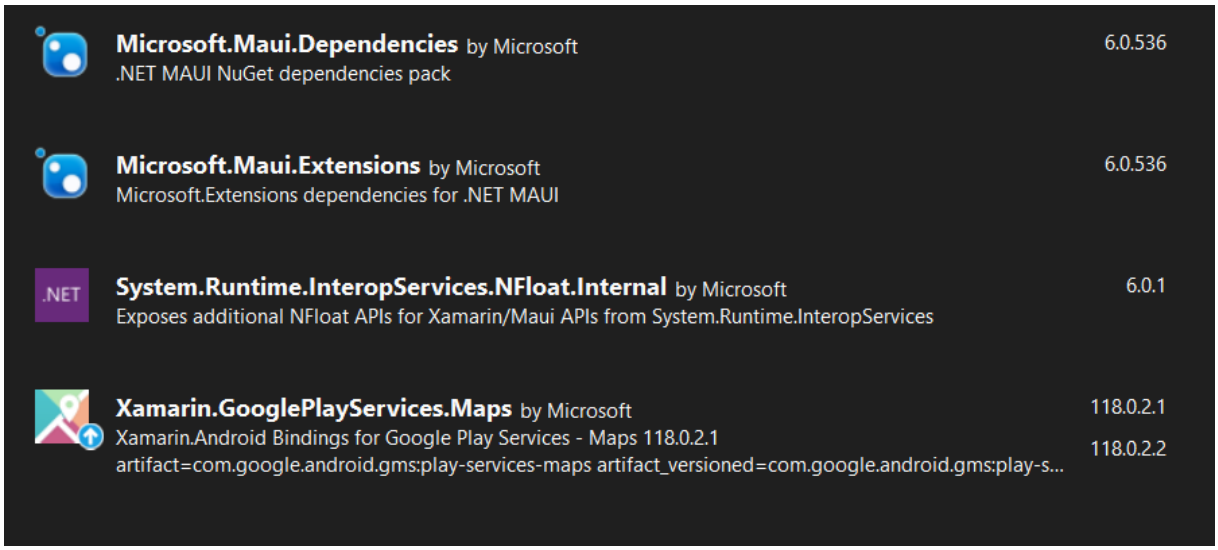
Vir: (Lastni vir)

Nato sem namestil NuGet pakete, ki so bili potrebni za vse funkcionalnosti te aplikacije, kot so prijava, registracija, uporaba kamere, prikaz in manipulacija zemljevida in shranjevanje podatkov (slik in uporabnikov). Namestil sem jih z desnim klikom na projekt in izbral »manage Nuget packages«. Ker je ogrodje relativno novo, je podpora za Google Maps še slaba, zato je bilo potrebno uporabiti in prilagoditi paket Xamarin.GooglePlayServices.Maps, da bi uporaba in prikaz zemljevidov bila mogoča. Preveril sem tudi pri .NET 7 RC1, kjer je uporaba zemljevidov že omogočena in poenostavljena, ampak se še vedno pojavlja preveč težav, zato sem se odločil uporabiti verzijo .NET 6, ki je stabilna in paket prilagodil za svojo rabo.



Slika 19 : Firebase NuGet paketi

Vir: (Lastni vir)



Slika 20 : Ostali NuGet paketi

Vir: (Lastni vir)

Sledila je implementacija mobilne aplikacije, ki sem si jo zamislil v analizi. Kot prva stran je bila razvita LoginPage, ki uporabniku omogoča prijavo v aplikacijo ali registracijo novega uporabnika. Nad prijavnim oknom sem uporabili lasten logotip.

Slika 21 : Aplikacija v ogrodju .NET Maui - Login okno
Vir: (Lastni vir)

Implementiral sem dve funkciji, ki se aktivirata na pritisk na gumb. Gumb prijava pogleda polji username in password in preveri njuno pravilnost. Nato preko FireBaseAuthProvider-ja z uporabo svojega Api ključa preverimo ali uporabnik obstaja. Če obstaja, odpremo nov MapPage, drugače izpišemo opozorilo uporabniku in izbrišemo polji za vnos emaila in gesla.

```
public LoginViewModel(INavigation navigation)
{
    this._navigation = navigation;
    RegisterBtn = new Command(RegisterBtnTappedAsync);
    LoginBtn = new Command(LoginBtnTappedAsync);
}
```

Slika 22 : Aplikacija v ogrodju .NET Maui - Login ViewModel

Vir: (Lastni vir)

```
private async void LoginBtnTappedAsync(object obj)
{
    var authProvider = new FirebaseAuthProvider(new FirebaseConfig(webApiKey));
    try
    {
        var auth = await authProvider.SignInWithEmailAndPasswordAsync(UserName, UserPassword);
        var content = await auth.GetFreshAuthAsync();
        var serializedContent = JsonConvert.SerializeObject(content);
        Preferences.Set("FreshFirebaseToken", serializedContent);
        await this._navigation.PushAsync(new MapPage());
    }
    catch (Exception ex)
    {
        await App.Current.MainPage.DisplayAlert("Opozorilo", "Nepravilen email ali geslo", "OK");
        UserName = "";
        UserPassword = "";
    }
}
```

Slika 23 : Aplikacija v ogrodju .NET Maui - Login funkcija

Vir: (Lastni vir)

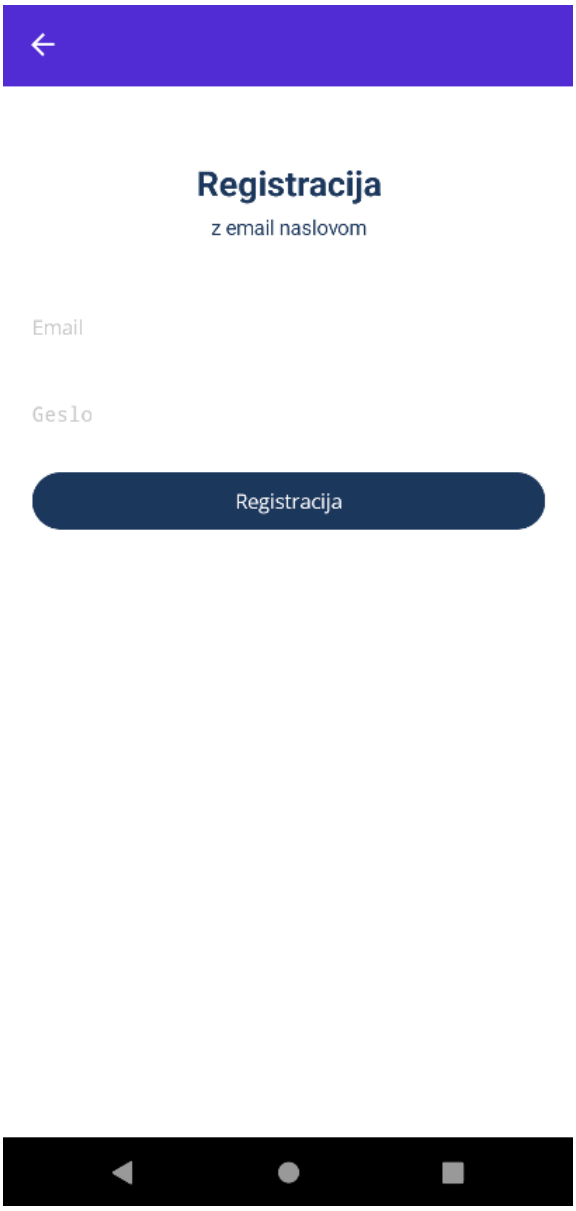
Gumb registracija pa kliče funkcijo RegisterBtnTappedAsync, ki enostavno odpre novo okno RegisterPage, kjer se uporabnik lahko registrira.

```
private async void RegisterBtnTappedAsync(object obj)
{
    await this._navigation.PushAsync(new RegisterPage());
}
```

Slika 24 : Aplikacija v ogrodju .NET Maui - Gumb za registracijo

Vir: (Lastni vir)

Tudi registracijsko okno ima podobno funkcionalnost, in sicer gleda na polji mail in geslo ter ob kliku na gumb Registracija izvrši funkcijo RegisterUserTappedAsync. Ta preveri, ali je vnesen email pravilne oblike in tudi geslo mora biti vsaj 6-mestno. Z uporabo FirebaseAuthProvider-ja vnesemo novega uporabnika v podatkovno bazo in vrnemo auth token. Nato preverimo, če je bila registracija uspešna (token ni enak null) in uporabnika obvestimo o uspešni registraciji ali pa ga opozorimo, da registracija ni bila uspešna, mu podamo razlog in izbrišemo polji za vnos.



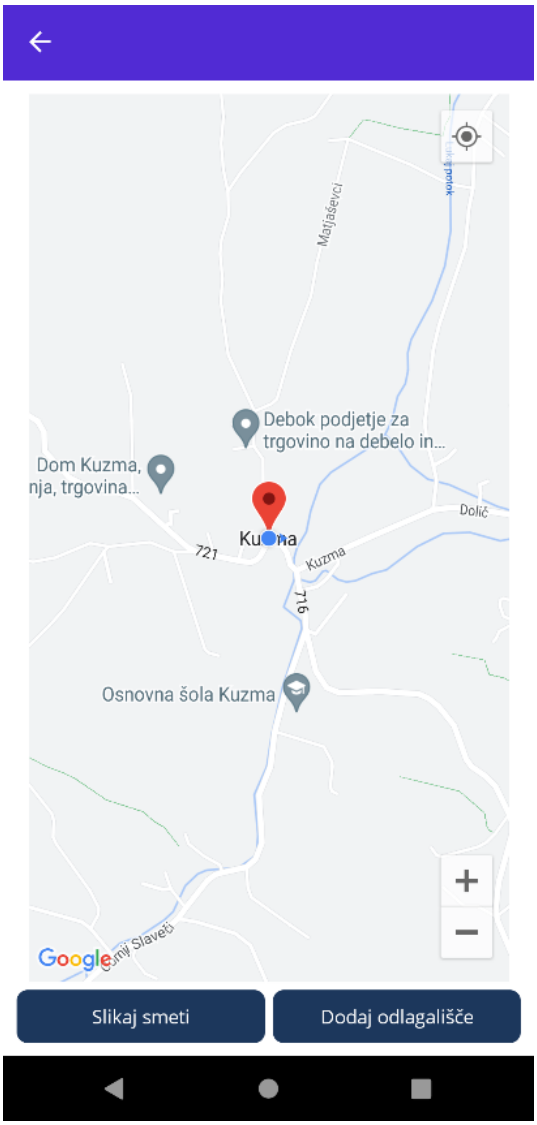
Slika 25 : Aplikacija v ogrodju .NET Maui - Okno registracija
Vir: (Lastni vir)

```
private async void RegisterUserTappedAsync(object obj)
{
    try
    {
        var authProvider = new FirebaseAuthProvider(new FirebaseConfig(webApiKey));
        var auth = await authProvider.CreateUserWithEmailAndPasswordAsync(Email, Password);
        string token = auth.FirebaseToken;
        if (token != null)
            await App.Current.MainPage.DisplayAlert("Uspeh", "Uporabnik uspesno registriran!", "OK");
        await this._navigation.PopAsync();
    }
    catch (Exception ex)
    {
        await App.Current.MainPage.DisplayAlert("Opozorilo", ex.Message, "OK");
        Email = "";
        Password = "";
    }
}
```

Slika 26 : Aplikacija v ogrodju .NET Maui - Funkcija za registracijo

Vir: (Lastni vir)

MapPage je glavno okno te aplikacije, kjer je prikazan GoogleMaps zemljevid in dva gumba, ki kličeta funkciji TakePhoto in AddMarkerToMap. Objekte za markerje kreiramo kot objekte razreda MyProperties, ki mu podamo zemljepisno širino in dolžino ter URL do slike divjega odlagališča, ki ga je posnel uporabnik.



Slika 27 : Aplikacija v ogrodju .NET Maui - Glavno okno

Vir: (Lastni vir)

```
public class MyProperties
{
    1 reference
    public MyProperties(double a, double b, string c)
    3 references
    public MyProperties(){}
    9 references
    public double lat { get; set; }
    9 references
    public double lng { get; set; }
    9 references
    public string imagelink { get; set; }
}
```

Slika 28 : Aplikacija v ogrodju .NET Maui - Razred MyProperties

Vir: (Lastni vir)

Gumb Photo kliče funkcijo TakePhoto, ki s pomočjo knjižnice MediaPicker ob dovoljenju uporabnika požene kamero. Če uporabnik sliko potrdi, se preko FirebaseStorage objekta z naslovom do naše aplikacije slika prenese na server in vrne URL do slike. Nato dobimo trenutno lokacijo, s pomočjo metode setCurrentLoc, in ustvarimo nov objekt MyProperties, ki ga s pomočjo funkcije UploadTempMarker prenesemo na server. Markerja ne ustvarimo, dokler uporabnik ne klikne na drugi gumb, saj se lahko odloči, da bo slikal iz drugega kota in lokacije.

```
async void TakePhoto(object sender, EventArgs args)
{
    var photo = await MediaPicker.Default.CapturePhotoAsync();
    if (photo != null)
    {   //Take photo
        var uploadImage = new FirebaseStorage("divjaodl.appspot.com")
            .Child(photo.FileName).PutAsync(await photo.OpenReadAsync());
        string downloadlink = await uploadImage;

        //lokacija + ustvarjanje objekta za upload
        setCurrentLoc();
        Location currloc = getCurrentLoc();
        double lat1, lng1;
        lat1 = currloc.Latitude;
        lng1 = currloc.Longitude;
        MyProperties a = new MyProperties(lat1, lng1, downloadlink);
        new Handlers.MapHandler().UploadTempMarker(a);
    }
}
```

Slika 29 : Aplikacija v ogrodju .NET Maui - Funkcija za zajem fotografije

Vir: (Lastni vir)

Gumb Dodaj Marker pa kliče funkcijo addMarkerToMap, ki s pomočjo funkcije getTempMarkers pridobi seznam neaktivnih markerjev in izbere zadnjega. Nato izbran marker s pomočjo funkcije UploadActuallMarker prenese na server aktivnih markerjev. Na koncu še dodamo izbran marker na zemljevid.

```
public static async Task<List<MyProperties>> getTempMarkers()
{
    FirebaseClient client = new FirebaseClient
        ("https://divjaodl-default-rtdb.europe-west1.firebasedatabase.app/");
    return (await client
        .Child("Temps")
        .OnceAsync<MyProperties>()).Select(item => new MyProperties
        {
            lat = item.Object.lat,
            lng = item.Object.lng,
            imagelink = item.Object.imagelink
        }).ToList();
}
```

Slika 30 : Aplikacija v ogrodju .NET Maui - Funkcija za pridobitev začasnih markerjev

Vir: (Lastni vir)

```
public async void UploadTempMarker(MyProperties a)
{
    new FirebaseClient("https://divjaodl-default-rtdb.europe-west1.firebasedatabase.app/")
        .Child("Temps")
        .PostAsync(a);
}
1 reference
public async void UploadÂctuallMarker(MyProperties a)
{
    new FirebaseClient("https://divjaodl-default-rtdb.europe-west1.firebasedatabase.app/")
        .Child("Locations")
        .PostAsync(a);
}
```

Slika 31 : : Aplikacija v ogrodju .NET Maui - Prenos markerjev na server

Vir: (Lastni vir)

```
public static async void AddMarkerToMap(IMapHandler handler, IMap map)
{
    GoogleMap? googleMap = handler?.Map;
    if (googleMap == null)
        return;

    //Get last temp marker
    MyProperties a = new MyProperties();
    List<MyProperties> temp = new List<MyProperties>();
    temp = await getTempMarkers();
    a.lat = temp[temp.Count - 1].lat;
    a.lng = temp[temp.Count - 1].lng;
    a.imagelink = temp[temp.Count - 1].imagelink;
    //Upload it as actuall marker
    new Handlers.MapHandler().UploadÂctuallMarker(temp[temp.Count - 1]);

    //Create the marker
    MarkerOptions mark = new MarkerOptions();
    double lat1 = a.lat;
    double lng1 = a.lng;
    LatLng latLng = new LatLng(lat1, lng1);
    mark.SetPosition(latLng);
    mark.SetTitle(a.imagelink);

    //Add the marker
    googleMap.AddMarker(mark);
    googleMap.UiSettings.MyLocationButtonEnabled = false;
}
```

Slika 32 : Aplikacija v ogrodju .NET Maui - Dodajanje markerja na zemljevid

Vir: (Lastni vir)

Uporabljena pa je tudi funkcionalnost, da ob vsaki novi inicializaciji zemljevida prenesemo vse aktivne markerje in jih dodamo zemljevidu.

```
//Get active markers and set them
double lat1, lng1;
MyActuallMarkers = await getList();
foreach (MyProperties item in MyActuallMarkers)
{
    MarkerOptions mark = new MarkerOptions();
    lat1 = item.lat;
    lng1 = item.lng;
    LatLng latlng = new LatLng(lat1, lng1);
    mark.SetPosition(latlng);
    mark.SetTitle(item.imagelink);
    Map.AddMarker(mark);
}
```

Slika 33 : Aplikacija v ogrodju .NET Maui - Nastavljanje obstoječih markerjev

Vir: (Lastni vir)

Celoten seznam aktivnih markerjev dobimo s klicem funkcije getList.

```
public static async Task<List<MyProperties>> getList()
{
    FirebaseClient client = new FirebaseClient("https://divjaodl-default-rtdb.europe-west1.firebasedatabase.app/");
    return (await client
        .Child("Locations")
        .OnceAsync<MyProperties>()).Select(item =>new MyProperties{
            lat = item.Object.lat,
            lng = item.Object.lng,
            imagelink = item.Object.imagelink
        }).ToList();
}
```

Slika 34 : Aplikacija v ogrodju .NET Maui - Pridobivanje obstoječih markerjev

Vir: (Lastni vir)

Implementiral sem tudi funkcijo, ki čaka na klik na marker in ob kliku odpre novo okno MarkerPicture, kjer se prikaže slika odlagališča, ki jo je posnel uporabnik za kliknjen marker.

```
public class MyMarkerClickListener : Java.Lang.Object, IOnMarkerClickListener
{
    1 reference
    public MyMarkerClickListener(MapHandler a)...

    0 references
    public bool OnMarkerClick(Marker marker)
    {
        var secondWindow = new Window
        {
            Page = new MarkerPicture(marker.Title){}
        };
        Application.Current.OpenWindow(secondWindow);
        return true;
    }
}
```

Slika 35 : Aplikacija v ogrodju .NET Maui - Odpiranje okna MarkerPicture

Vir: (Lastni vir)

Slika 36 : Aplikacija v ogrodju .NET Maui - Okno MarkerPicture

Vir: (Lastni vir)

3.5 Ugotovitve

Pri implementaciji obeh aplikacij sem ugotovil sledeče. Izdelava z ogrodjem Flutter je bila brez posebnih zapletov. Flutter-jeva dokumentacija je bogata in zelo nazorno pokaže, kako uporabljati posamezne funkcije in vmesnike. Veliko je primerov aplikacij oz. Delnih aplikacij, kjer lahko na primeru poiščeš rešitev. Tudi uporaba razvojnega okolje je bila domača in enostavna. Flutter omogoča ogromno funkcionalnosti in ne omejuje razvijalca. Gnezdimo lahko skoraj vsak objekt na drug objekt. Čeprav mi je razvijanje v Dart-u bilo novo, je sintaksa podobna kot pri vseh ostalih pogosto uporabljenih programskih jezikih. Aplikacija deluje zelo hitro in tekoče, uporaba je enostavna in vse funkcije se izvršijo v zelo hitrem času.

Implementacija .NET Maui aplikacije je bila nekoliko zahtevnejša, saj je ogrodje relativno novo in še večina vmesnikov ostaja na Xamarin-u. Tudi osnovne knjižnice in osnovne funkcionalnosti so polne hroščev (ang. Bug), ki otežujejo delo razvoja. Sicer se težave sproti odpravljajo in tudi veliko funkcionalnosti naj bi bilo na voljo z novim .NET 7. Tako je bilo potrebno dodati lastne handlerje za zemljevid, ki izhaja iz vmesnika, ki je prvotno bil razvit za Xamarin. Veliko več se mi je zdelo omejitev pri sami implementaciji, čeprav sem že razvijal aplikacije v C#. Delovanje aplikacije je tekoče, a funkcije se ne izvajajo tako hitro.

Če primerjam aplikaciji, je trenutno lažje in z manjšimi preprekami implementirati android aplikacijo z ogrodjem Flutter. Flutter je ustaljeno ogrodje, z veliko bazo uporabnikov, ki se trudijo razvijati vedno nove vmesnike in ponuditi članke, vodiče in dokumentacije, ki poenostavijo razvoj z ogrodjem. Tudi sama aplikacija je trenutno veliko bolj odzivna od aplikacije, narejene z ogrodjem .NET Maui. Moje mnenje je, da Microsoft rabi še nekaj časa, da se .NET Maui razvije do konca v pravo stabilno verzijo, kjer bi mogoče lahko pariral Googlu s Flutter-jem na čelu.

4 MERITVE

Meritve sem opravil na lokalnem računalniku v android emulatorju. Napisal sem funkciji in, ki ob kliku na gumb sprožita štoparico in poženeta for zanko, ki se ponovi 10 krat. Vsaka zanka poizkuša pridobiti trenutno lokacijo uporabnika. Po uspešni izvršitvi se štoparica ustavi. Primerjal sem tudi aplikaciji glede na hitrost uporabniškega vmesnika in možnosti, ki jih ogrodje dopušča.

4.1 Flutter

V ogrodju Flutter sem implementiral funkcijo, ki se je sprožila ob pritisku na gumb in 10 krat pridobila uporabnikovo lokacijo. Za dostop do lokacije Flutter uporabi Geolocator vmesnik, ki je zelo hiter in enostaven za uporabo. Rezultati so bili sledeči.

Tabela 3 : Flutter - Rezultati testne funkcije 1

Meritev	1	2	3	4	5	6	7	8	9	10
Čas(s)	1.2240	1.0882	1.1149	1.0138	1.401	1.219	1.155	1.189	1.122	1.367

Vir: (Lastni vir)

```
void Testcase_Location() async{
  final stopwatch = Stopwatch()..start();
  for(int i=0;i<=10;i++){
    await Geolocator.getCurrentPosition();
  }
  print('Testcase_Location executed in ${stopwatch.elapsed}.' );
}
```

Slika 37 : Flutter testna funkcija

Vir: Lasten vir

Naslednji test je bil testiranje funkcije za pridobitev markerjev s podatkovne baze Firebase. Funkcija je stokrat prenesla markerje in jih shranila v listo. Časi so bili zelo enakomerni in kratkotrajni. Vsakič je preneslo 10 markejev.

Tabela 4 : Flutter - Rezultati testne funkcije 2

Meritev	1	2	3	4	5	6	7	8	9	10
Čas(ms)	9.88	9.60	9.55	9.45	9.64	9.72	9.5	9.2	9.48	9.53

Vir: (Lastni vir)

```
void Testcase_GetMarkers() async{
  final stopwatch = Stopwatch()..start();
  final List<myProperties> orderedResult = [];
  final Query query = ref
      .orderByKey();
  for(int i=0;i>100;i++){
  query.onChildAdded.forEach((event) {
    orderedResult.add(myProperties.fromMap
      (event.snapshot.value as Map<dynamic,dynamic>));
  });
  print('Testcase_GetMarkers executed in ${stopwatch.elapsed}.' );
  }
}
```

Slika 38 : Flutter testna funkcija 2

Vir: Lasten vir

Zadnje testiranje je bilo glede nalaganja markerjev v podatkovno bazo Firebase. Funkcijo sem izvedel 100 krat. Tudi tukaj so časi izvajanja bili zelo enakomerni, brez odstopanj.

Tabela 5 : Flutter - Rezultati testne funkcije 3

Meritev	1	2	3	4	5	6	7	8	9	10
Čas(ms)	8.05	7.77	6.44	6.21	6.33	6.7	6.29	6.4	6.2	6.15

Vir: (Lastni vir)

```
void Testcase_UploadMarker() async{
  final stopwatch = Stopwatch()..start();
  for(int i=0;i>100;i++){
      uploadMarker();
    }
  print('Testcase_UploadMarker executed in ${stopwatch.elapsed}.' );
}
```

Slika 39 : Flutter testna funkcija 3

Vir: Lasten vir

4.2 .NET Maui

V aplikaciji ogrodja .NET Maui sem implementirali podobno funkcijo s podobno funkcionalnostjo, le da sem uporabil Geolocation, ki kliče funkcijo GetLocationAsync. Štoparica je imela isto funkcionalnost, da izmeri čas izvajanja. Presenetljivo so bili časi veliko večji kot pri ogrodju Flutter.

Tabela 6 : .NET Maui - Rezultati testne funkcije 1

Meritev	1	2	3	4	5	6	7	8	9	10
Čas(s)	9.437	9.513	9.228	9.551	9.475	9.389	9.322	9.462	9.210	9.188

Vir: (Lastni vir)

```
async void Testcase_Location(object sender, EventArgs args) {
    Stopwatch stopwatch = new Stopwatch();
    stopwatch.Start();
    for(int i = 0; i <= 10; i++) {
        await Geolocation.Default.GetLocationAsync();
    }
    stopwatch.Stop();
    }
```

Slika 40 : .NET Maui testna funkcija

Vir: Lasten vir

Drugi primer testiranja je bil kot pri ogrodju Flutter prenos markerjev v listo. Izvajanje te funkcije sem ponovil 100 krat. Tudi tu so rezultati bili enakomerni, a vseeno nekoliko višji od ogrodja Flutter. Vsakič je preneslo 10 markerjev.

Tabela 7 : .NET Maui - Rezultati testne funkcije 2

Meritev	1	2	3	4	5	6	7	8	9	10
Čas(ms)	16.3	15.64	13.44	14.87	14.66	14.23	14.45	14.50	14.63	14.50

Vir: (Lastni vir)

```
async void Testcase_GetMarkers(object sender, EventArgs args) {
    Stopwatch stopwatch = new Stopwatch();
    stopwatch.Start();

    for (int i = 0; i > 100; i++)
    {
        FirebaseClient client =
            new FirebaseClient("https://divjaodl-default-rtdb.europe-west1.firebasedatabase.app/");
        var a = (await client
            .Child("Locations")
            .OnceAsync<MyProperties>()).Select(item => new MyProperties
            {
                lat = item.Object.lat,
                lng = item.Object.lng,
                imagelink = item.Object.imagelink
            }).ToList();
    }
    stopwatch.Stop();
    Debug.WriteLine("Time Taken: " + stopwatch.Elapsed.ToString());
}
```

Slika 41 : .NET Maui testna funkcija 2

Vir: Lasten vir

Zadnji primer testiranja je bil testiranje nalaganja markerjev v spletno podatkovno bazo Firebase. Tudi to funkcijo sem izvedli 100 krat. Tudi tukaj se je izkazalo ogrodje Flutter za hitrejše.

Tabela 8 : .NET Maui - Rezultati testne funkcije 3

Meritev	1	2	3	4	5	6	7	8	9	10
Čas(ms)	9.88	9.90	9.63	9.50	9.46	9.55	9.34	9.45	9.64	9.33

Vir: (Lastni vir)

```
async void Testcase_UploadMarker(object sender, EventArgs args) {
    Stopwatch stopwatch = new Stopwatch();
    stopwatch.Start();
    for(int i=0; i < 100; i++)
    {
        new Handlers.MapHandler().UploadActuallMarker(temp);
    }
    stopwatch.Stop();
    Debug.WriteLine("Time Taken: " + stopwatch.Elapsed.ToString());
}
```

Slika 42 : .NET Maui testna funkcija 3

Vir: Lasten vir

4.3 Ugotovitve

Pri testni funkciji 1, ki je 10 krat pridobila uporabnikovo lokacijo, sem ugotovil, da je bil čas v ogrodju Flutter krajši, in sicer za 8.213 sekunde v primerjavi z ogrodjem .NET Maui.

Tabela 9 : Primerjava testne funkcije 1

Meritev	1	2	3	4	5	6	7	8	9	10
Čas(s) Maui	9.437	9.513	9.228	9.551	9.475	9.389	9.322	9.462	9.210	9.188
Čas(s) Flutter	1.2240	1.0882	1.1149	1.0138	1.401	1.219	1.155	1.189	1.122	1.367

Vir: (Lastni vir)

Pri naslednjem testu funkcije 2 sem ugotovil, da so bili časi meritev zelo enakomerni in kratkotrajni. Vsakič je preneslo 10 markerjev. Ugotovil sem, da je bil .NET Maui počasnejši za 6.42 sekunde.

Tabela 10 : Primerjava testne funkcije 2

Meritev	1	2	3	4	5	6	7	8	9	10
.NET Maui Čas(ms)	16.3	15.64	13.44	14.87	14.66	14.23	14.45	14.50	14.63	14.50
Flutter Čas(ms)	9.88	9.60	9.55	9.45	9.64	9.72	9.5	9.2	9.48	9.53

Vir: (Lastni vir)

Prav tako so bili rezultati pri testiranju zadnje funkcije, ki sem jo izvedel 100 krat zelo enakomerni in brez odstopanj. In sicer je čas ob uporabi ogrodja Flutter hitrejši za 1.83 sekunde, glede na uporabo ogrodja Flutter.

Tabela 11 : Primerjeva testne funkcije 3

Meritev	1	2	3	4	5	6	7	8	9	10
.NET Maui Čas(ms)	9.88	9.90	9.63	9.50	9.46	9.55	9.34	9.45	9.64	9.33
Flutter Čas(ms)	8.05	7.77	6.44	6.21	6.33	6.7	6.29	6.4	6.2	6.15

Vir: (Lastni vir)

5 SKLEP

Namen diplomske naloge je bil raziskati ogrodji Flutter in .NET Maui, implementirati aplikaciji v obeh ogrodjih ter ju primerjati. Najprej sem teoretično preučil obe ogrodji in raziskal možna razvojna okolja. Nato sem raziskal problem divjih odlagališč in kakšne funkcionalnosti bi morale biti vsebovane v aplikaciji za divja odlagališča. Usmeril sem se v osnovne funkcionalnosti, kot so registracija in vpis uporabnika, določanje lokacije, zajem slike ter nalaganje in prenos lokacij odlagališč oz. markejev na zemljevidu.

Najprej sem z razvil aplikacijo v ogrodju Flutter, ta se je izkazal za stabilno ogrodje z veliko možnosti razvoja, sploh z uporabo neštetih vmesnikov, ki so uporabnikom na voljo. Za razvoj sem uporabil brezplačne vmesnike s strani Firebase-a in osnovne vmesnike Googla za upravljanje kamere in zemljevida.

Razvoj aplikacije v ogrodju .NET Maui je bil nekoliko zahtevnejši, saj je ogrodje novo in še zmeraj v razvoju. Srečal sem se s težavo pri prikazu zemljevida, saj še ni bilo razvitih vmesnikov ali handlerjev za Google Maps. Pomagal sem si s starejšim vmesnikom Xamarina in ga preoblikoval za obstoječe ogrodje. Za ostale funkcionalnosti sem tudi uporabil osnovne vmesnike ogrodja .NET Maui in vmesnike s strani Firebase-a, da bi aplikaciji lahko kasneje lažje primerjal.

Nato sem na podlagi implementirane aplikacije ogrodji primerjal iz teoretičnega in praktičnega vidika. Primerjal sem hitrost pridobitve lokacije, čas nalaganja in čas prenosa slike odlagališča. Rezultati so pokazali jasno prevlado ogrodja Flutter, česar ob postavljanju hipotez nisem pričakoval. Ogrodje Flutter je enostavno stabilno, saj je v razvoju že od leta 2017, napram ogrodja .NET Maui, ki se je začel razvijati šele v letu 2022, kot naslednik ogrodja Xamarin.

Prva hipoteza, H1, ki sem si jo zastavil, pravi, da je začetniku lažja uporaba ogrodja Flutter. Le to bom potrdil, saj se je ogrodje zelo lahko naučiti, ima namreč veliko dokumentacije, ki se stalno posodablja. Pri tem pomaga tudi ogromna skupnost, ki ima na spletu že veliko repository-jev in strani, kjer se lahko učimo na primeru. Tudi sintaksa Dart-a je zelo enostavna in je podobna vsem večjim programskim jezikom.

Kot naslednjo hipotezo, H2, sem si zastavil, da je za izdelavo enostavnih aplikacij uporaba ogrodja Flutter stroškovno ugodnejša. To hipotezo sem delno potrdil. Pri komercialnem razvoju aplikacije ali če razvoj prepustimo razvijalcu, je razvoj v ogrodju Flutter nedvomno cenejši. A vseeno lahko z uporabo enostavnejših in brezplačnih orodij razvijamo tudi aplikacije v ogrodju

.NET Maui. Edina razlika, ki sem jo opazil, je, da so za ogrodja Flutter brezplačna orodja boljša od brezplačnih orodij za ogrodje .NET Maui.

Hipotezo H3 sem zavrgel. Predpostavil sem, da ima ob uporabi geolokacija aplikacija v ogrodju .NET Maui boljše performanse. Čeprav v drugih testih ni bilo velikih razlik, je pri testiranju geolokacije prišlo do enormnih razlik v hitrosti. Pri obeh sem uporabil build-in funkcije za pridobitev lokacije in prišel do zaključka, da je bil .NET Maui kar 8 krat počasnejši od aplikacije, ki je bila implementirana v ogrodju Flutter.

Kot zadnjo hipotezo, H4, sem si zastavil, da je v obeh ogrodjih možno implementirati aplikacijo, ki bi lahko pripomogla k zmanjšanju onesnaževanja okolja. Hipotezo sem potrdil, saj imata obe ogrodji dovolj orodij za razvoj take aplikacije, prav tako pa tudi razviti aplikaciji omogočata vso zastavljeno funkcionalnost. Menim, da bi taka aplikacija lahko zelo pripomogla k iskanju in čiščenju divjih odlagališč pa tudi pri čistilnih akcijah.

V diplomskem delu sem uspel odgovoriti na vsa zastavljena vprašanja in hipoteze ter raziskal razvoj aplikacije, ki bi lahko pripomogla k čistejšemu okolju.

6 VIRI IN LITERATURA

.NET MAUI is HERE! 3 NEW Features that will blow your mind. (2021). Pridobljeno iz DEV: https://dev.to/bytehide/net-maui-is-here-3-new-features-that-will-blow-your-mind-ngp

A tour of the C# language. (brez datuma). Pridobljeno iz Microsoft: https://learn.microsoft.com/en-us/dotnet/csharp/tour-of-csharp/

Akram, N. (2022). *BUILDING A REAL WORLD MAUI APP - THE GOOD, THE BAD AND THE UGLY*. Pridobljeno iz http://blogs.xgenoapps.com/post/2022/05/16/building-a-real-world-maui-app-the-good-the-bad-and-the-ugly

Android Studio. (brez datuma). Pridobljeno iz JavaTPoint: https://www.javatpoint.com/android-studio

Basu, S. (2021). *Why Evolve to .NET MAUI?* Pridobljeno iz Progress Telerik: https://www.telerik.com/blogs/why-evolve-dotnet-maui

Bellinaso, M. (2018). *Flutter: the good, the bad and the ugly*. Pridobljeno iz The ASOS Tech Blog: https://medium.com/asos-techblog/flutter-vs-react-native-for-ios-android-app-development-c41b4e038db9

Chisholm, K. J. (brez datuma). *What's new in Flutter 3.3*. Pridobljeno iz Published in Flutter: https://medium.com/flutter/whats-new-in-flutter-3-3-893c7b9af1ff

Contributor, T. (brez datuma). *Android Studio* . Pridobljeno iz Tech Target: https://www.techtarget.com/searchmobilecomputing/definition/Android-Studio

Heller, M. (2022). *What is Visual Studio Code? Microsoft's extensible code editor*. Pridobljeno iz Info World: https://www.infoworld.com/article/3666488/what-is-visual-studio-code-microsofts-extensible-code-editor.html

Illegal Dumping: Causes, Effects and Solutions to Huge Piles of Wastes. (brez datuma). Pridobljeno iz Conserve Energy Future: https://www.conserve-energy-future.com/causes-effects-solutions-illegal-dumping.php

India, Y. t. (30. Maj 2019). *How to Outsource Mobile App Development Effectively?* Pridobljeno iz Your team in India: https://www.yourteaminindia.com/blog/outsource-app-development

Roheel, F. (brez datuma). *What are the features of Flutter?* Pridobljeno iz Educative: https://www.educative.io/answers/what-are-the-features-of-flutter

Thewissen, S. (2022). *An Introduction to .NET MAUI*. Pridobljeno iz Code Magazine: https://www.codemag.com/Article/2111082/An-Introduction-to-.NET-MAUI

What is .NET MAUI? (2022). Pridobljeno iz Microsoft: https://learn.microsoft.com/en-us/dotnet/maui/what-is-maui

Zaccagnino, C. (2020). *Programming Flutter: Native, Cross- Platform Apps the Easy way.* The Pragmatic Programmers.