Bachelorproef draft (14/05) (Officieel 28/05)

Inhoudsopgave

Samenvatting

Inleiding

De impact van mobiele applicaties (vanaf nu apps) op ons alledaags leven is niet te onderschatten. Ze zijn de dag van vandaag niet meer weg te denken. Dit komt door de grootte toename in de verkoop van smartphones. Ongeveer 40% van de wereldbevolking beschikt over een smartphone en in 2020 gingen 1.38 miljard nieuwe toestellen de deuren uit. De toename in verkochte smartphones leidt tot een grotere markt voor apps. Dit hebben softwarebedrijven niet over het hoofd gezien. Het aantal applicaties aanwezig op de iOS App Store en de Google Play Store, is in de voorbije tien jaar respectievelijk vertwintig- en verdertigvoudigd. Elke smartphone maakt gebruik van een besturingssysteem, dit is de software die de hardware aanstuurt. Tegenwoordig is er een oligopolie van twee spelers op de markt van smartphone besturingssystemen (vanaf nu OS voor Operating System). Apple ontwikkelde hun eigen OS genaamd iOS terwijl de meeste andere merken zoals Samsung en OnePlus gebruik maken van het Android OS. Elk OS heeft een verschillende manier van applicatieontwikkeling. In de meeste gevallen zullen app eigenaars met hun applicatie een zo breed mogelijk publiek willen bereiken. Hierdoor zullen ze apps moeten ontwikkelen voor zowel iOS en Android.

Wat is Native Development?

Als het ontwikkelingsteam kiest om een applicatie native te gaan ontwikkelen dan wil dit zeggen dat een aparte codebase (verklaren) moet ontwikkeld worden voor elk mobiel besturingssysteem waar de app moet op kunnen draaien. Verschillende codebases wil zeggen meer ontwikkelings- en onderhoudswerk maar geeft anderzijds wel een optie tot een native- design en gedrag per codebase.

Wat is Cross-platform Development?

Een andere aanpak is het ontwikkelen van een cross-platform app. Dit is het schrijven van één codebase die kan gecompileerd (vertaald) worden naar verschillende besturingssystemen. Het is een optie die de laatste jaren meer en meer gekozen wordt. De voor- en nadelen die hiervoor zijn aangehaald, draaien zich om. Een enkele codebase is goedkoper om te bouwen en te onderhouden maar dit moet dan ook afgewogen worden tegen het laten van native- designs en gedrag. Er zijn verschillende ontwikkeltools gemaakt voor het maken van cross-platform apps. Een paar van de grootste zijn React Native, Xamarin, Flutter en Ionic.

Wat is Flutter

Flutter is een Google UI toolkit voor het bouwen van mooie, bijna natively gecompileerde apps voor mobiel, web, en desktop en dit alles van één enkele codebase. Flutter kondigde op 3 maart 2021 hun nieuwe update aan die hen weer een stap in de goede richting duwde. Dit is onder andere te danken aan de grootte steun en de snelle tractie van het platform in de voorbije jaren. Flutter is opensource wat wil zeggen dat ze hun broncode openstellen voor het publiek, waardoor iedereen het kan bekijken. Zo kunnen verbeteringen worden gedaan door eindgebruikers die het beste voor hebben met het platform. Als Flutter deze verbeteringen goedkeurt worden deze opgenomen in de broncode en beschikbaar gesteld voor iedereen.

Waarom Flutter versus native Android?

Flutter en Android zijn beiden ontwikkeld door Google. Alhoewel Flutter relatief jong is, kan dit niet gezegd worden van native Android. Het platform kende al veel updates en staat al lang niet meer in zijn jonge schoentjes. Vandaar dat het toetsen van deze twee een interessante kijk kan geven op het jonge flutter platform. De Flutter applicatie zal geschreven worden voor- en alleen gecompileerd worden naar Android. De sterkte van Cross-platform development is natuurlijk één codebase voor verschillende besturingssystemen, maar voor dit onderzoek is alleen Android van belang. Het onderzoek zal rekening houden met het geschreven aantal lijnen code per uitgewerkt hoofdstuk. Als het onderzoek rekening moet houden met iOS brengt dat het evenwicht uit balans.

Waarom Kotlin Android en geen Java Android?
Deze keuze is voor velen persoonlijk alhoewel hier ook onderzoek naar verricht is. In deze studie wordt Kotlin gebruikt omdat de syntax van de taal zal helpen bij het intomen van het aantal lijnen code. Hoe minder lijnen code, hoe eenvoudiger en overzichtelijker de app. Het is voor velen dan ook de voorkeurstaal om Android-applicaties in te ontwikkelen. Dit onderzoek zal geen stap voor stap uitleg inhouden van hoe de code in elkaar zit. Het doel is een vergelijking schetsen tussen Flutter en Android. De code die wordt geschreven zal dus alleen worden aangehaald indien dit relevant is voor de studie.

Probleemstelling
Het te onderzoeken domein kwam vanuit Next Apps. Een native app ontwikkelingsbedrijf uit Lokeren dat zich specialiseert in Android en iOS watch, phone en tablet apps. Zij merken dat niet alle klanten op de hoogte zijn van het hoge kostenplaatje van een native ontwikkelde app. Om in de toekomst een middenweg te vinden en de kost van een kleine app te dempen, denken zij erover om sommige apps cross-platform te gaan ontwikkelen. Bij het onderzoeken van de cross-platform markt kwamen ze Flutter tegen, het nieuwe platform met veel potentieel. Daarom werd de vraag gesteld om de stand van het Flutter platform te onderzoeken. Met deze scriptie hopen ze een duidelijk beeld te krijgen op de limieten en mogelijkheden van het framework.

Onderzoeksvraag
Hoofdonderzoeksvraag
Wat zijn de voor- en nadelen van app ontwikkeling in Flutter in vergelijking met native Android?
Het onderzoek zal zich vooral richten op het vinden van een antwoord op deze hoofdvraag. Aan de hand van deze kan een duidelijk beeld worden geschetst van de huidige stand van Flutter in vergelijking met Android. Hierbij wordt gelet op de ontwikkelingstijd van de features, de complexiteit van de code, het aantal lijnen geschreven lijnen code…
Hypothese: de voorkeur van ontwikkelingsplatform zal hoogstwaarschijnlijk afhangen van de soort app die ontwikkeld moet worden. Een grootere app met veel features zal eerder native ontwikkeld worden, terwijl kleinere apps die weinig native behaviour kunnen bevatten waarschijnlijk cross-platform ontwikkeld zullen worden.

Deelonderzoeksvragen
Is Flutter al matuur genoeg om te aanschouwen als volwaardig alternatief op native app ontwikkeling?
Deze vraag biedt een antwoord aan alle native app ontwikkelaars die denken om de overstap te maken naar cross-platform development. Het is belangrijk om een overzicht te krijgen in de limieten van een platform alvorens tijd en geld te investeren.
Hypothese: Flutter zal geen duidelijke standaard hebben. Best practices, goed ondersteunde libraries en coding principles zijn nog niet gedefinieerd door het platform waardoor het voor beginners onduidelijk zal lijken wat de beste manier van aanpak zal zijn. Ondanks de snelle groei en de grootte steun van het platform wordt wel verwacht dat hier snel verandering in gebracht wordt.

Is Flutter toegankelijk voor nieuwe ontwikkelaars?
Het proces van het ontwikkelen van de Flutter applicatie zal antwoord bieden op deze vraag. Voor velen is het moeilijk om te schakelen naar een nieuw platform met een nieuwe taal. Om deze overgang transparant te maken, wordt een beeld geschetst van de moeilijkheidsgraad van Flutter en Dart. Hierdoor kan de lezer zelf beslissen of hij deze keuze wil maken.
Hypothese: voor ontwikkelaars met een Java of C++ achtergrond zou de Dart taal geen probleem mogen vormen. Ze hebben een gelijkaardige syntax. Verwachtingen van het Flutter platform liggen hoog. De snelle groei van de gebruikersbasis doet vermoeden dat het een toegankelijk platform is.

Onderzoeksdoelstelling

Zoals hiervoor vermeld zal dit onderzoek de voor- en nadelen van het Flutter framework proberen vinden. Om dit op een duidelijke manier te doen, zal het framework getoetst worden aan native Android. Door het jonge platform af te wegen tegen het ontwikkelde native tegengestelde, wordt gehoopt op een duidelijk beeld van Flutter. Het zal interessant zijn om te weten of het platform te kort doet aan native Android, want deze dure tegenhanger moet zo goed mogelijk zijn troeven uitspelen om niet ten onder te gaan aan het goedkope alternatief. Het onderzoek wordt gevoerd aan de hand van een vergelijkende studie.

Voor het onderzoek worden twee identieke applicaties ontwikkeld. Beide applicaties zullen ontwikkeld worden in Android Studio maar de ene zal geschreven worden in Kotlin terwijl de andere geschreven wordt in Dart. Door het ontwikkelproces van de Flutter app af te toetsen tegen dat van de native Android app zal hopelijk een beeld worden geschetst van de stand van het Flutter framework. Met dit beeld zal dan een antwoord gevormd worden op de onderzoeksvraag.

Dit onderzoek gaat gepaard met een deadline, deze tijdsdruk zorgt ervoor dat een aantal criteria moeten gesteld worden alvorens te starten. Deze criteria zorgen voor een kwaliteitsvol eindresultaat in een haalbaar tijdsbestek. Het eerste aantal criteria gaat over de performantie van de twee apps. Dit criterium wordt verder onderverdeeld in drie verschillende criteria. Allereerst wordt gekeken naar de opstartsnelheid van de apps. Vervolgens wordt de grootte van de uitvoeringsbestanden bekeken. Als laatste wordt gekeken naar het CPU gebruik van beide apps. De volgende criteria is creatie van views, hier wordt gekeken hoe een layout zich vertaalt naar een view. Hoe complexe views gemaakt worden en wat de mogelijkheden en grenzen zijn van cross-platform layouts.

Het beoogde resultaat van de studie biedt een antwoord op alle onderzoeksvragen. Indien dit lukt, kan de studie als succesvol worden beschouwd, anders zal een suggestie worden gedaan voor toekomstig bijkomend onderzoek.

Opzet van deze bachelorproef

Deze scriptie kan opgedeeld worden in verschillende hoofdstukken, elk hoofdstuk kan bestaan uit verschillende secties of ondertitels. Deze sectie beschrijft de inhoud van elk van deze hoofdstukken.

Hoofdstuk 2 Stand van zaken: bevat een literatuurstudie. Een samenvatting van relevante onderzoeken die reeds voorafgingen aan dit onderzoek.

Hoofdstuk 3 Methodologie: omschrijft hoe het onderzoek in zijn werk zal gaan. De opzetting van het onderzoek en het onderzoek zelf zullen duidelijk toegelicht worden.

Hoofdstuk 4 Performantie: bestaat uit drie hoofdstukken die elk een belangrijk deel van app performantie onderzoeken.

- Grootte van het uitvoeringsbestand:
- Opstartsnelheid:
- CPU gebruik

Hoofdstuk 5 Creatie van views:

Hoofdstuk 6 Beschikbare libraries en code complexiteit:

Hoofdstuk 7 Asynchroon werken:

Hoofdstuk 8 Security:

Hoofdstuk 9 Beschikbare tools:

Hoofdstuk 10 Appendix: zal verdere uitleg bieden rond bepaalde onderwerpen voor de geïnteresseerde lezer.

Hoofdstuk 11 Conclusie: dit is de algemene conclusie op deze scriptie. In deze conclusie zal een antwoordt gegeven worden op de reeds aangehaalde onderzoeksvragen. Daarbij wordt ook aangezet tot toekomstig onderzoek binnen dit vakgebied.

Stand van zaken

Dit hoofdstuk is equivalent aan een literatuurstudie. De eerste stap was het zoeken naar voorgaande studies over Flutter en Android. Elke studie die voldeed aan de eisen en als interessant werd beschouwd, werd opgeslagen en gebundeld. Hiervan werd de inhoud diagonaal gelezen om zo alleen de meest relevante onderzoeken over te houden.

Vervolgens werd een opsomming gemaakt en een kort beeld geschetst van een aantal van deze studies en hun bijdrage in het vakgebied. In de volgende sectie zullen dus alleen de meest relevante studies, die dicht aansluiten bij dit onderzoek, worden aangehaald. De ondervindingen uit deze studies werden gebruikt als voorbereiding op het onderzoek en kunnen in volgende hoofdstukken gebruikt worden als steunpunten. Door de ondervindingen van verschillende studies te bundelen, kon gezocht worden naar een rode draad.

De te onderzoeken criteria van deze studie werden vastgelegd op basis van de reeds voorgaande onderzoeken. Het is de bedoeling dat deze studie bijdraagt aan het vakgebied. Hierdoor wordt weerhouden van reeds onderzochte criteria opnieuw te gaan onderzoeken. Toch zal blijken dat sommige reeds onderzochte criteria opnieuw worden bekeken. Deze herhaling komt voort uit de Flutter 2.0 release, deze zorgde namelijk voor veranderingen in de prestatiemogelijkheden van het framework.

Voorgaande onderzoeken

Android Native Development in Kotlin versus het Flutter Framework, een vergelijkende studie

Navaron Bracke, onderzocht in 2020 de verschillen en gelijkenissen tussen Android en Flutter. De scriptie vormde een beeld over de toenmalige stand van het Flutter framework door het te toetsen tegen het ontwikkelde Android. Het onderzoek was afgebakend door een vooraf vastgelegd aantal criteria. Zo kon de onderzoeker zekerheid bieden over de grondigheid van de studie. De onderzochte criteria waren: Internationalisering, navigatie, persisteren van gegevens, user Interfaces, asynchroon werk, permissies, software testing, opstarttijd van de applicatie, grootte van de applicatie.

De opzet en het doel van het onderzoek liep in grootte mate gelijk met het onderzoek gevoerd voor deze scriptie. Daarom was het onderzoek van Bracke uitermate interessant, bevindingen werden bijgehouden, genoteerd en getoetst tegen de bevindingen van dit onderzoek. De hoofdstukken die performantie onderzochten werden gebruikt om een beeld te schetsen van de Flutter evolutie over tijd. De andere twee hoofdstukken (User Interfaces en Asynchroon werk) werden opnieuw onderzocht met conclusies van Bracke in het achterhoofd. Hiervoor werd, in de mate van het mogelijke, zoveel mogelijk gefocust op de onderwerpen die Bracke niet aanhaalde.

De onderzoeker constateerde dat, ondanks de jonge aard van het Flutter platform, het een goed alternatief biedt op elk onderzocht criterium. Bij elk van deze criteria werd een platform van voorkeur gekozen. Hieruit bleek Flutter op 4 van de 7 criteria naar voor te komen. Al werd wel meegedeeld dat niet alle features in Flutter even gebruiksvriendelijk zijn maar door de jonge aard werd verwacht dat dit in de toekomst zou verbeteren.

Backdrop- an exploration of Flutter

Austen Latture, een student aan Grand Valley State Universiteit, schreef in 2020 een studie over het ontwikkelen van een app in Flutter. De focus van het onderzoek lag hem op de leercurve van het platform. Hier waren geen vooropgestelde criteria aanwezig, deze studie omschreef de uitwerking van de Backdrop app.

In conclusie bleek Flutter een toegankelijk platform te zijn voor nieuwe ontwikkelaars met een allesbehalve steile leercurve. Een ander groot voordeel aangeboden door het platform is de talrijke ingebouwde functionaliteit. De Dart taal die wordt geschreven voor het maken van Flutter apps is volgens het onderzoek zeer gemakkelijk aan te leren voor mensen met een achtergrond in de C taal. Aan de andere kant bleek het kiezen van third-party libraries een hinder blok te vormen tijdens de ontwikkelfase. De jonge aard van het platform ging gepaard met jonge libraries.

Zoals bleek uit andere studies concludeerde dit onderzoek dat Flutter hier een rode draad in mist. De grootte hoeveelheid aangeboden libraries maakt het moeilijk om te weten welke goed ondersteund worden en dan ook vaak gekozen worden. Dit is vooral een struikelblok voor nieuwe developers, onbekend met het platform. Vervolgens werd Flutter ook in het kort vergeleken met React Native. Hieruit kon geconcludeerd worden dat Flutter veel gemakkelijker te installeren is. Dit komt, volgens het onderzoek, voort uit de ingebouwde functionaliteit van Flutter terwijl React meer werkt met verspreide derde partij software die meestal bestaat uit te veel boilerplate code.

Teaching mobile app development: choosing the best development tools in practical labs

Daniel Dang en David Skelton, twee professors aan de Insitutue of Technology, schreven in 2019 een wetenschappelijk artikel over mobile app development in het onderwijs. Het doel van het onderzoek was het bepalen van het ideale framework voor het aanleren van mobiele applicatieontwikkeling in het onderwijs. Het onderzoek werd gevoerd aan de hand van verschillende native- en cross-platform frameworks. Zo werden native Android, native iOS, Flutter, React Native, Ionic, Xamarin, Cordova en Appcelerator vergeleken. Onderzochte criteria bij dit onderzoek waren: Design App User Interface, User Event Handling, Services & Multiple threads used in Internet connection, Graphics and Animation, Device hardware and sensor, Wireless connectivity, Internal data storage, Real time database.

In conclusie bleken Flutter en Native Android de twee beste platformen te zijn voor gebruik tijdens praktijklessen. Ook beval het onderzoek acht ideale onderwerpen aan die in de praktijksessies moeten worden behandeld om studenten voldoende technische vaardigheden bij te brengen om alle soorten mobiele apps te ontwikkelen. Belangrijk hierbij te vermelden is dat de studie gebruik maakte van een oude Flutter release.

Creating Flutter Apps from Native Android Apps

Carlos Chavez, een student en Yoonsik Cheon, een professor aan de Universiteit van Texas in El Paso onderzochten in 2020 het herschrijven van Native Android Apps naar Flutter apps. Het onderzoek bekeek een native Android app ontwikkeld in de Java-taal en probeerde deze zo goed mogelijk om te zetten naar een Flutter app die terug kon gecompileerd worden als een Android app maar ook als iOS app. Het onderzoek werd gevoerd aan de hand van het Flutter ontwikkelproces. De technische-, praktische problemen en uitdagingen die de kop opstaken werden vervolgens besproken.

Het onderzoek deed een paar duidelijke verschillen uitblijken tussen Android en Flutter wanneer gekeken werd naar de taal, programmeerstijl en design. Het grootste werk was het herschrijven van de views terwijl het herschrijven van de achterliggende logica eerder snel gebeurd was. Het onderzoek focuste zich onder andere op het aantal lijnen code geschreven voor beide applicaties. De Flutter code bleek hierbij maar 2/3 van het totaal aantal Android code te zijn. Maar volgens de onderzoekers was dit niet genoeg om apps in Flutter te beginnen ontwikkelen. De leeftijd van het platform bracht nog te veel onzekerheid met zich mee. Ook ontbrak dit onderzoek aan een soort standaard werkwijze of verzameling van richtlijnen. Dit sloeg dan vooral op libraries en API’s binnen het platform. De gebruikersbasis en de beschikbare tools moesten hierbij ook zeker in rekening worden gebracht. Het grote aantal third party libraries, door de grootte aanhang, zorgde voor onzekerheid bij het zoeken naar een goed ondersteunde library. Bij andere grootte platformen wordt dit verholpen doordat na bepaalde duur de beste API’s en libraries naar boven komen en een soort van community standard wordt gezet.

A Comparison of Looks Between Flutter and Native Applications

Mathilda Olsson, student aan de Blekinge hogeschool onderzocht in 2020 hoe Flutter applicaties vergelijken met native applicaties. Voor dit onderzoek werden twee native applicaties ontwikkeld in Kotlin voor Android en Swift voor iOS. Deze applicaties werden vervolgens beoordeelt op vlak van CPU performantie. Het onderzoek bevatte ook een bevraging die zich focuste op de verschillen in gebruikersperceptie. Een aantal gebruikers moesten de app eerst testen waarna ze een vragenlijst konden beantwoorden waaruit verschillen in look en feel moesten duidelijk worden.

Uit onderzoek bleek dat Flutter en Android apps gelijkaardige resultaten halen op vlak van CPU performantie. Vervolgens werd gekeken naar het aantal lijnen code en de complexiteit van de twee code bases. Hieruit bleek dat Flutter aanzienlijk minder lijnen code nodig had maar ook minder complexe code bevatte. De Flutter app bestond uit 125 lijnen code terwijl de twee native apps samen 580 lijnen code bevatte. Dit verschil is substantieel en duidt erop dat Flutter beter blijkt voor kleine tot middelgrote applicaties. Uit de vragenlijst bleek native ontwikkeling, op vlak van UI, de voorkeur te hebben van het grootste deel gebruikers. De verschillen waren vooral te zien bij animaties, fonts, gedrag, lijsten en menu's. Animaties konden wel gelijkgesteld worden maar dat is extra werk zijn voor de ontwikkelaar. In conclusie, zoals in de andere onderzoeken, werd nogmaals geduid op de leeftijd van het platform. Alle criteria waar Flutter minder op scoorde konden nog volop in ontwikkeling zitten voor een volgende release. Flutter had volgens het onderzoek veel potentieel indien de steun van de gebruikersbasis zo groot bleef.

Erik Blokland – An Empirical Evaluation of the User Interface Energy Consumption of React Native and Flutter (2019)

Erik Blokland, een student aan de University of Technology in Delft, schreef in 2019 een vergelijkende studie over de energieconsumptie van UI tussen React Native, Flutter en de Android.

Hierbij werd Android als standaard genomen en werden de twee andere apps hiermee vergeleken.

Drie vragen werden gesteld:

- Bestaat een verschil in energie verbruik tussen de drie apps
- Hoeveel energie verbruiken specifieke UI acties
- Is er een verschil tussen deze UI acties bij elke app

In conclusie bleken de verschillen tussen UI acties over de platformen heen zeer stabiel te zijn.

De grote verschillen die gevonden werden waren in de verschillende uitgevoerde acties.

Het openen van bijvoorbeeld een drawer menu was een veel energie zwaardere taak als het openen van een modal.

Als laatste was het niet duidelijk uit onderzoek of er een verschil was tussen de verschillende apps in energieconsumptie.

Deze testen waren te inconsistent om conclusies uit te trekken.

Jakhongir Fayzullaev – Native-like cross-platform mobile development (2018)

Jakhongir Fayzullaev, een student aan Universiteit Toegepaste Wetenschappen Xamk, schreef in 2018 een vergelijkende studie. Deze studie onderzocht drie verschillende cross-platform development frameworks namelijk Kotlin Native, Multi-OS en Flutter. De frameworks en dus de studie is vooral interessant voor Android ontwikkelaars aangezien de onderzochte frameworks dicht aansluiten bij de Java en Android ontwikkelingsmanieren. In 2018, toen het onderzoek werd uitgevoerd, waren de twee cross-platform tools nieuwe spelers op de markt. Het onderzoek kon zich niet baseren op voorgaande studies en ontbrak aan duidelijke richtlijnen van de frameworks. Hierdoor was het onderzoek oppervlakkig, het vergeleek de grote lijnen van de platformen zonder de details te bekijken. Het Flutter deel van het onderzoek bekeek in grote lijnen: Widgets, het maken van layouts en de bouwstenen van flutter. Hij concludeerde dat op vlak van performantie de drie platformen zo goed als identiek waren. Onderzoek deed blijken dat Flutter een goed platform was in 2018 maar de jonge aard zorgde voor weinig bruikbare libraries en tools. Ook had Flutter in 2018 heel weinig built-in functionaliteit met als gevolg meer werk voor de programmeurs die meer code moesten schrijven ten opzichte van de andere meer ontwikkelde platformen. Fayzullaev omschrijft deze manier van coderen als 'het opnieuw uitvinden van het wiel'.

Lukas Dagne - Flutter for cross-platform App and SDK development (2019)

Lukas Dagne, schreef in 2019 zijn Bachelorproef over Flutter in vergelijking met Native- en andere Cross-platform ontwikkelings platformen. De scriptie baseerde zich op een enkele grootte punten binnen App development: de interne specificaties van het framework en de mogelijke architectuur van een app. De conclusie stelde dat Flutter een goede keuze is voor cross-platform development. De weinige toegevingen die gedaan worden bij het ontwikkelen van een cross platform app in Flutter zorgt voor een duidelijk verschil met de andere frameworks. De grootte aanhang van het platform en de snelle ontwikkelingen duiden volgens Dagne alleen maar op een verbetering naar de toekomst toe. Alhoewel dat de SDK ontwikkeling van het framework nog niet optimaal is het geloof in verbetering groot.

Ly Hong Hoang - State management in Flutter (2019)

Ly Hong Hoang, in 2019, een student aan de Metropolia Universiteit Toegepaste Wetenschappen schreef een Bachelorproef studie over state management in Flutter. Hierbij werd gekeken welk state-management systeem het best past bij Flutter. De drie vergeleken systemen waren Business Logic Component (BloC), Redux en Scoped Model. Dit is geen wetenschappelijke studie, de resultaten van de paper zijn geen feiten maar eerder een discussiepunt met een conclusie getrokken op basis van een klein onderzoek. De conclusie stelde dat BloC het best was voor gebruik in Flutter. BloC is volledig op punt gesteld voor Dart, de taal die wordt geschreven in Flutter. Ergens is dit ook wel de logische keuze aangezien het ontwikkeld is door Google. Redux is hoog performant maar complex. Dit is bijvoorbeeld niet goed om te gebruiken in teamverband. Scoped Model is gemakkelijker om te begrijpen maar minder performant als Redux. Ook is het niet gemakkelijk schaalbaar. BloC Is schaalbaar, performant en simpel te verstaan. Het heeft dus alle goede kwaliteiten. Toen deze paper geschreven werd waren er nog niet veel best practices binnen Flutter. Daarom wou de schrijver met deze studie een soort van benchmark zetten voor het kiezen van een state managementsysteem binnen Flutter.

Exploring End User's Perception of Flutter Mobile Apps

Ola Dahl schreef in 2019 een master thesis in kader van de Zweedse Universiteit Malmö. De thesis ging gebruikers perceptie tussen twee identieke apps vergelijken. De eerste app werd geschreven in Native Android en de tweede in Flutter. Er werd dan een vergelijking opgesteld tussen de twee apps, gebruikers konden aan de hand van een aantal vragen aangeven welke app ze verkozen. Het onderzoek werd gevoerd op een kleine groep gebruikers dus conclusies zijn niet representatief. Deze studie concludeerde dat gebruikers de native applicatie verkozen boven de Flutter applicatie op vlak van snelheid. Echter op vlak van uitstraling en design ondervonden de gebruikers geen verschillen. De meerderheid verkoos de Android applicatie, slechts 10% verkoos de Flutter applicatie.

Using Google's Flutter Framework for the development of a large-scale reference application

Sebastian Faust schreef in 2020 Bachelorproef Thesis in naam van de Universiteit Toegepaste Wetenschappen in Keulen. Het doel van dit onderzoek is andere applicatieontwikkelaars helpen wanneer zij voor dezelfde problemen komen te staan tijdens het schrijven van een Flutter app. Voor dit onderzoek schreef hij een app, tijdens dit proces documenteerde hij alle design keuzes en de problemen/obstakels die hieruit voortkwamen. Na het evalueren van elk obstakel kon hij altijd tot een optimale oplossing komen. Na de studie werden ondervindingen gebundeld en een set van richtlijen opgesteld voor het ontwikkelen van grootte Flutter applicaties. Richtlijnen en best practices waar developers zich best aan houden tijdens het ontwikkelingsproces. Deze werden gebundeld in een gids die later werd gepubliceerd en goed ontvangen werd door de Flutter community. Voor verdere ondersteuning is nog een interview afgenomen met een Flutter expert. Dit onderzoek werd toegevoegd aan de literatuurstudie om aan te tonen dat Flutter apps schaalbaar zijn.

Flutter Engage - Flutter 2.0

Op woensdag 3 maart 2021 werd Flutter Engage georganiseerd, een live event gebracht door het Flutter team. Hier kwamen verschillende grootte namen binnen de Flutter community aan bod om de nieuwe releases aan de wereld te tonen. De developers gaven het de naam Flutter 2.0 door de talrijke upgrades die het platform is ondergaan. Niet alleen het Flutter platform kreeg een upgrade maar ook de taal Dart werd verbeterd voor een vlottere en vertrouwelijke programmeer ervaring.

Grootste Flutter upgrades:

- Flutter is van een mobiel framework naar een portable framework gegaan. Oorspronkelijk werd Flutter alleen gecompileerd voor iOS en Android besturingssystemen maar door de upgrade zijn daar: Windows, macOS, Linux en Web bijgekomen. Dit wordt gezien als gratis upgrade aangezien het geen oude code breekt maar indien gewenst door een beperkt aantal lijnen code te herschrijven kan men naar 3 keer zoveel systemen compileren.
    - Canvas kit
    - SEO is nog niet goed ondersteund maar staat op het plan voor de toekomst
- Vouwbare smartphones werden ondersteund.
- Web is beschikbaar als stabiele release.
- Desktop ondersteuning (macOS, Windows, Linux) is beschikbaar als stabiele release (under early release flag)
    - De installatie van Ubuntu (Linux) wordt herschreven in Flutter
- Upgrade Firebase plugin -> Flutter Fire
- Google Mobile Ads SDK (open beta) (plugin)
- Nieuwe iOS features zijn aangekondigd
- Nieuwe widgets teogevoegd aan het framework.

Grootste Dart upgrades:

- Een van de grootste punten was dat sound null safety is toegevoegd, zonder oude code te breken. Dit is een grootte upgrade voor het schrijven van robuuste code. Dart zal zeggen waar er mogelijkse gevaren zitten op null excepties. Dit zorgt voor minder errors en dus betere code.
- Smarter flow analysis
- Late variables
- Required named parameters
- DevTools upgrade

Ook werden in de keynote verschillende grootte bedrijven zoals Ubuntu, Microsoft, Toyota… genoemd die in de toekomst nauw gaan samenwerken met Flutter. Deze samenwerkingen tonen de kracht, potentieel en groei van het platform.

Ook werd in de Keynote het aantal open issues op Github aangehaald. Doordat het een opensource framework is kan iedereen die de broncode wil zien, deze gaan opzoeken op GitHub. Moest een fout stuk code, een simpeler te schrijven stuk code of zelfs als een goede bijdrage aan het platform gevonden worden kan hier een issue van gemaakt worden. Als Flutter deze verbetering heeft nagekeken en goedgekeurd wordt deze toegevoegd aan de broncode. Op deze manier sparen zichzelf een hoop werk uit en groeit het platform snel en volgens de wensen van de gebruikers. Het grootte aantal issues die momenteel open staan zien zij dan ook als een teken van groei. De vele gebruikers die die Flutter willen beter maken is een teken dat er veel vertrouwen is in het platform.

Er zal wel rekening moeten gehouden worden met de upgrade bij het runnen van oudere applicaties, door de updates en aanpassingen zullen veel API's verouderd zijn. Flutter heeft dit goed geanticipeerd door het Flutter fix commando toe te voegen. Deze zal de bestaande code doorlopen en tonen welke API's verouderd zijn en in wat ze best vervangen worden.

Dart data classes staan op de toekomst plannen

Waarom deze studie?

- Zien of er nog problemen/moeilijkheden zijn bij het gebruik van Flutter
- Zien of en hoe je hier kan rondwerken

Cross-platform development heeft een snelgroeiende gebruikersbasis. De vele updates en verbeteringen aan de platformen maakt cross-platform development dan ook elke dag aantrekkelijker. Dit leidt tot meer onderzoek en studies binnen de sector. Tijdens de literatuurstudie bleek dat een groot aantal papers Flutter vergeleek met een ander cross-platform ontwikkel tool. Dit soort papers moet de lezer helpen bij het maken van een keuze tussen cross-platform tools. Door het verdelen van de aandacht over verschillende platformen, komt Flutter niet uitgebreid aanbod in de meeste van deze soort papers. Voor 2020 waren de meeste papers van deze aard. Flutter leek toen nog niet stabiel en groot genoeg om als afzonderlijke tool onderzocht te worden.

Zoals uit voorgaande sectie blijkt, is Flutter meerdere malen onderzocht in 2020. In verschillende studies werd het tekort aan onderzoek voor 2020 aangehaald. Flutter is in het afgelopen jaar in een stroomversnelling geraakt op vlak van onderzoek. Dit is te danken aan de grootte stijging in het gebruikersaantal. Het platform is nog steeds relatief jong maar de aantrekkingskracht ligt hem in de gebruikersbasis en de grootte steun. Hoe meer gebruikers, hoe belangrijker het product in kwestie. Hoe belangrijker het product, hoe meer middelen hierin worden geïnvesteerd en dat lokt op zijn beurt weer meer gebruikers. Deze cirkel wordt alleen onderbroken als het platform in kwestie zijn aantrekkelijkheid verliest. Flutter lijkt hier geen last van te hebben en deze studie wil laten blijken waarom.

Dus kan geconcludeerd worden dat Flutter een interessant onderzoeksdomein is. Maar zoals bij elk onderzoek binnen de IT, zijn bevindingen al snel verouderd. De rappe evolutie binnen de sector doet elk product streven naar de beste versie van zichzelf. De competitie is hoog, dus moet performantie altijd voorop staan. Bij elke update of verbetering zijn voorgaande performantie studies verouderd en niet accuraat. Hierdoor zijn de studies, aangehaald in dit hoofdstuk, dan ook vaak zo recent mogelijk. Toch onderging Flutter net een grootte release. Vele zaken werden verbeterd, onder andere de performantie van het platform. De voorgaande studies zijn weer verouderd en hier ligt een kans om opnieuw onderzoek te doen en beeld te schetsen van de huidige performantie van Flutter. Ook lijkt het interessant om een beeld te schetsen van de performantie van Flutter over tijd.

Methodologie

- Twee grote pilaren
    - Literatuurstudie
    - Experiment

Om een antwoord te kunnen bieden op zowel de primaire, als de deelonderzoeksvragen, werd een onderzoek verricht. Voorafgaand aan dit onderzoek werd een literatuurstudie uitgevoerd die in deze bachelorproef als de “Stand van zaken” omschreven is. Onderdelen die uit deze studie gebruikt werden, zijn bijvoorbeeld performantie-statistieken. Meer bepaald werden zowel oude als nieuwe statistieken gebruikt en vervolgens tegenover elkaar gelegd.

Om de verschillen tussen Flutter en Android te kunnen toetsen, werd ervoor gekozen om 2 applicaties te maken. Enerzijds een applicatie geschreven met behulp van Flutter, anderzijds een applicatie geschreven met behulp van Android. Dit houdt in dat er gebruik gemaakt wordt van 2 platformen en 2 verschillende talen. Voor de Flutter ontwikkeling zal gebruik gemaakt worden van de Dart taal, terwijl voor de Android ontwikkeling Kotlin gebruikt zal worden. Beide van deze talen werden ontwikkeld door Google, maar Kotlin is sinds 7 mei 2019 de voorkeurstaal van Google voor app-ontwikkeling. (Link naar artikel)

Hetgeen beide applicaties gemeenschappelijk hebben, naast hun effectieve inhoud, is dat ze beiden geschreven werden in dezelfde Integrated Development Environment (kortweg IDE), namelijk Android Studio. Deze omgeving is een softwarepakket dat door ontwikkelaars gebruikt wordt om nieuwe software te ontwikkelen.

Zoals reeds aangehaald hierboven zullen er twee, in de mate van het mogelijke, identieke applicaties gemaakt worden. Op die manier is het extreem duidelijk op welke vlakken de twee platformen van elkaar verschillen en wat de voor- en nadelen zijn van deze tegenover de andere. Zo zal er gekeken worden naar de hoeveelheid van code die geschreven dient te worden op de 2 platformen om tot hetzelfde resultaat te komen. Om dit verschil te kunnen onderzoeken, zal gekeken worden naar het aantal lijnen code. Losstaand van het aantal lijnen code zal tevens gekeken worden naar de complexiteit van de code en de leesbaarheid ervan.

Daarnaast zal bij de uitwerking van deze verschillende onderdelen van het experiment, telkens gekeken worden naar de kwaliteit van de beschikbare libraries die kunnen gebruikt worden om bepaalde code die quasi telkens hetzelfde is, de zogenaamde boilerplate code, te vermijden. Voorbeelden waar het gebruik van libraries extreem handig kunnen zijn, zijn bijvoorbeeld bij het uitvoeren van API requests.

Aan de hand van de uitkomsten van deze verschillende onderdelen, zal naar een conclusie toegewerkt worden. Deze conclusie is te vinden in het volgende hoofdstuk.

Experiment

- 2 applicaties
- 2 platformen
- 2 talen
- 1 omgeving
- Lijnen code
- Libraries
- Complexiteit van de code

Zoals in Hoofdstuk 1 reeds vermeld werd, valt dit onderzoek onder het luik vergelijkende studie.

Hiervoor zullen twee applicaties ontwikkeld worden die beiden zo identiek mogelijk zijn op vlak van look and feel.

De applicaties zullen ontwikkeld worden in de Android Studio Intigrated Development Enviroment. Android Studio biedt de snelste tools voor het bouwen van apps op elk type Android-apparaat.

De Android applicatie zal geschreven worden in Kotlin. Kotlin is een gratis, open source programmeertaal ontworpen voor de JVM en Android.

De Flutter applicatie zal ontwikkeld worden in de Dart taal.

Onderzoekscriteria

- Elk hoofdstuk aanhalen en hoe dit onderzocht en aangepakt zal worden
- Elk hoofdstuk kijken voor libraries

Deze scriptie is onderverdeeld in verschillende hoofdstukken. Voorgaande hoofdstukken beschreven onder andere de inhoud en opzet van het onderzoek, terwijl de volgende hoofdstukken over het experiment gaan. Elk hoofdstuk omvat een onderzoekscriteria,

Hoofdstuk 4 Performantie behandelt de performantie aspecten van de applicatie. Hieronder vallen drie

Het

Performantie

- Wat is performantie binnen een app
- Waarom is performantie belangrijk
- Welke performantie aspecten
- Waarom deze hoofdstukken
- Afhankelijk van hardware
- Proberen zoeken naar platform verschillen
- Toetsen aan voorgaande onderzoeken
- Metric die vaak veranderd
- Net grootte Flutter release

https://flutter.dev/docs/perf/rendering/best-practices
https://medium.com/flutter/flutter-performance-updates-in-the-first-half-of-2020-5c597168b6bb
https://flutter.dev/docs/perf/rendering/ui-performance
https://medium.com/swlh/flutter-vs-react-native-vs-native-deep-performance-comparison-990b90c11433

/*De snelle vooruitgang in app development zorgt ervoor dat gebruikers de lat alsmaar hoger leggen voor nieuwe apps. Meer en meer belang wordt gehecht aan de performantie kant van een app. Daarom worden in volgende secties een paar van de meest belangrijke aspecten rond app performantie aangehaald. Het is belangrijk dat app ontwikkelaars de applicatie zo schrijven dat deze de beste performantie biedt aan de gebruiker. Toch zit het onderliggende framework hier ook voor een deel tussen. Met dit hoofdstuk zal een beeld worden geschetst hoe beide platformen scoren op vlak van performantie*/

Dit hoofdstuk zal het performantie aspect van de applicatie behandelen. Ondanks het feit dat nieuwe smartphones alsmaar sneller worden, is het nog steeds belangrijk om veel belang te hechten aan de performantie van een applicatie tijdens de ontwikkeling ervan. Hetgeen evenredig stijgt met de snelheid van de nieuwe smartphones, zijn de verwachtingen van de gebruiker bij het gebruiken van een app. De gebruiker gaat er immers vanuit dat een snelle smartphone gelijk staat aan snelle applicaties op die smartphone.

Er zijn echter enkele belangrijke factoren die invloed hebben op die welbepaalde performantie, die hieronder kort toegelicht worden.

Grootte van de uitvoeringsbestanden

Een eerste factor welke invloed heeft op de performantie van een applicatie is de grootte van zijn uitvoeringsbestand. Het doel is om dit uitvoeringsbestand zo klein mogelijk te houden, zodat de gebruiker geheugen kan besparen op zijn toestel. Het is immers zo dat de prijs van high-end smartphones vandaag de dag vrij steil is. De consument kan bij het aanschaffen van een nieuw toestel vaak kiezen voor een bepaalde hoeveelheid opslag. Wanneer voor een toestel gekozen wordt met een vrij beperkte opslagcapaciteit en de gebruiker een aantal grote apps installeert, zal de opslagcapaciteit van het toestel vrij snel volledig ingenomen zijn. Dit heeft als gevolg dat een gebruiker bepaalde applicaties zal gaan verwijderen zodat hij terug ruimte kan creëren. Uit onderzoek bleek namelijk dat 1/2 van de applicaties verwijderd wordt wegens de grootte ervan. Dit is één van de redenen waarom het dus in de eerste plaats sowieso al een goed idee is om de app zo klein mogelijk te houden, maar de hoogste performantie te garanderen.

Een andere reden is de maximale toegestane grote van de uitvoeringsbestanden opgelegd door de verschillende app stores. Om een applicatie om de Google Play Store te kunnen beschikbaar stellen, moet deze voldoen aan meerdere eisen. Een van deze eisen is dat het uitvoeringsbestand van de app maximaal 100 MegaByte bedraagt. Het doel van een applicatie beschikbaar te stellen op de Play Store is immers om een zo groot mogelijk doelpubliek aan te spreken, wanneer echter niet aan de eisen van de Play Store voldaan wordt, zal dit doelpubliek niet bereikt worden. In dit hoofdstuk zal gekeken worden naar de grootte van de uitvoeringsbestanden van respectievelijk de Flutter applicatie alsook de Android applicatie.

Opzet

Een uitvoeringsbestand, ook wel gekend als een executable, van Android Apps wordt een Android Package (APK) genoemd. Deze APK's worden gebruikt om, zoals de naam al doet vermoeden, een Android applicatie uit te voeren. Voor dit onderzoek zijn er 4 APK's gemaakt. Twee in native Android en twee in Flutter. Om de verschillen tussen beide platformen te testen, zullen tot tweemaal toe de verschillende tegenhangers tegenover elkaar geplaatst worden. De Flutter APK's worden dus met andere woorden vergeleken met hun native tegenhanger.

De eerste Flutter app gemaakt in het kader van dit onderzoek was een welbekende en gebruikelijke Hello World applicatie. Een Hello World app klinkt alle ontwikkelaars bekend in de oren. Het is namelijk een standaard template die niet meer doet dan het tonen van een "Hello World" tekst op een scherm. Het is zeer minimaal in aantal lijnen code, wat ergens wel logisch is, en zorgt voor een heel kleine, compacte APK. Om enige vorm van realisme in de ontwikkeling van deze voorbeeld applicaties te integreren, zal er gekozen worden om de builds van deze applicaties via een release schema uit te voeren. Deze manier van builden optimaliseert de code in zijn geheel, zoals de applicatie terug te vinden zou zijn op de Play Store bijvoorbeeld.

Resultaten

Resultaten Flutter
6.7Mb -> 4.7Mb ->
Zelfde problemen bij andere cross-platofrm tools
Resultaten Native Android
550Kb

Wat uit deze resultaten kan geconcludeerd worden is dat er een duidelijk verschil is tussen enerzijds de native APK's en de cross-platform APK's. Het is namelijk zo dat native APK's aanzienlijk kleiner zijn in vergelijking met hun cross-platform tegenhangers.

Opstartsnelheid van de app

Inleiding

Een andere veel voorkomende reden waarom gebruikers applicaties slecht beoordelen of verwijderen is een trage applicatie. Onderzoek toont aan dat gebruikers de verwachting hebben dat een applicatie binnen maximaal een drietal seconden opstart. Een startscherm waarachter alle data van de applicatie geladen wordt is dan ook een bad practice. De gebruikers verwachten dat alles snel gaat en kan een trage startsnelheid leiden tot frustratie. Aangezien elke ontwikkelaar mikt op een zo goed mogelijke user experience (UX), is het interessant om ook de opstartsnelheid van een applicatie te onderzoeken. De vraag die hier naar boven rijst is dus met andere woorden die dat de impact onderzoekt van beide platformen op de startsnelheid.

Opzet

Om het verschil in opstartsnelheid tussen beide platformen te testen, zal gebruik gemaakt worden van de Hello World applicaties die hierboven eerder aangehaald werden. Door de kleine omvang van de applicaties, zullen de verschillen in tijd beperkt zijn. Doch is het interessant om aan de hand van een groot aantal iteraties te gaan kijken of het verschil wél groot is of net niet. Hieruit kan een gemiddelde opstarttijd berekend worden per applicatie, die het mogelijk maakt om beide platformen met elkaar te gaan vergelijken.

Wat belangrijk om melden is, is dat beide applicaties vanaf nul gestart worden. Het is namelijk zo dat applicaties uit drie verschillende toestanden gestart kunnen worden. De Engelse termen voor deze drie opstartprocedures zijn volgende: cold start, warm start en hot start. (Verder toelichten). In dit onderzoek zal gebruik gemaakt worden van de cold start procedure. Deze procedure is degene die de grootste uitdaging vormt in het kader van het minimaliseren van de opstarttijd. Dit is namelijk zo omdat het systeem en de applicatie in deze toestand het meeste werk moeten verrichten om effectief de startprocedure af te ronden.

Resultaten

Android
Flutter

CPU gebruik van de app

- Metric die vaak veranderd
- Door de tijd bekijken

Een Central Processing Unit of kortweg CPU is het als het ware het brein van een computer. Het is een stuk hardware dat aan de hand van een aantal basisoperaties, programma's uitvoert. Om deze operaties performant uit te voeren heeft de CPU echter nood aan een vaste, degelijke bron van energie. In het geval van mobiele apparaten, zoals een laptop of een smartphone, is deze energiebron de batterij van het toestel. Vandaag de dag moeten, en zijn, smartphones vrij tot zeer compact. Dit heeft echter ook een nadeel. Door de compactheid van de toestellen, is de plek voor de batterij ook eerder beperkt. Om de capaciteit van de batterij zo goed mogelijk te benutten, moeten de processen die op de toestellen draaien dus eerder performant zijn. Hiermee wordt bedoeld dat ze energiezuinig moeten zijn zonder aan veel snelheid te moeten inleveren.

Om het CPU gebruik van de verschillende applicaties te testen werd ervoor gekozen om, volgens hetzelfde stramien als hierboven net beschreven, aan de hand van verschillende iteraties tussen de twee applicaties te kijken welke van de twee het beste presteert. In dit geval wordt beter presteren beschouwd als minder CPU gebruiken om een proces minstens even snel uit te voeren.

Opzet

Resultaten

Android

Flutter

Conclusie

Gebruik van online API's

Inleiding

- Toegankelijkheid voor nieuwe developpers
- Beschikbare libraries
- Edge cases

Wanneer ontwikkelaars de term API horen, weten zij direct waarover het gaat. Een Application Programming Interface, of dus kortweg een API, is software die het toelaat om verschillende applicaties of applicatielagen met elkaar te laten communiceren. Een van de primaire use cases waarvoor een API gebruikt wordt in de mobiele applicatie wereld, is die om de zogenaamde API calls uit te voeren. Het doel van deze API calls zijn om vanuit de mobiele applicatie, de API aan te spreken om data op te halen die vervolgens getoond kan worden. Een API zorgt er in dit geval voor dat een mobiele applicatie in connectie staat met de cloud. Naast het ophalen van data kan er vaak ook nieuwe data naar de API gestuurd worden, of kan bestaande data zelfs aangepast worden.

Opzet

Voor dit en de volgende experimenten, werd een nieuwe applicatie uitgewerkt voor beide platformen. Deze applicatie bevat voorts alle nodige functionaliteit waaraan dient voldaan te worden om de experimenten tot een goed eind te brengen. Voor dit specifieke experiment, namelijk degene waarvoor gebruik gemaakt wordt van een online API, werd ook functionaliteit uitgewerkt. Er werd gekozen om een online API te gebruiken die duidelijk en goed ondersteund was. Eén van de bekendste en gebruiksvriendelijke API's is de PokéApi. Deze API, die informatie omtrent allerlei Pokémon teruggeeft, is duidelijk ondersteunt en gedocumenteerd waardoor hij perfect is voor dit experiment. De documentatie omtrent de PokéApi is te lezen op een overzichtelijke site die gebruik maakt van een RESTful interface.

Om dit experiment uit te voeren, werd gebruik gemaakt van zogenaamde data klassen. Deze klassen worden gebruikt om de verschillende objecten op te bouwen die terugkomen van de API. Aan de hand van bijvoorbeeld een framework, kan dan gebruik gemaakt worden van deze objecten om de data uit te lezen en te gebruiken in de applicatie zelf. Enkele van de libraries die hiervoor in Android veel, en makkelijk te gebruiken zijn, zijn Retrofit, Moshi, Chuck, Glide en Volley. Voor Flutter wordt vooral gebruik gemaakt van de HTTP Package library.

Data classes
Libraries

Android

- Retrofit
- Moshi
- Chuck
- Glide
- Volley

Flutter

- HTTP package

Resultaten

Android

Flutter

Conclusie

Security

(https://developer.android.com/topic/security/best-practices)

Inleiding

- Door gehele app
- Richtlijnen zoeken
- In app security?
- Permissions
- Hoe gemakkelijk is dit alles te implementeren in beide frameworks

Security of beveiliging is vandaag de dag meer dan enkel een hot topic. Het is een nood en het moet een garantie zijn dat aangeboden wordt aan de gebruiker. Doorheen de gehele applicatie is security een belangrijk aspect. Zo is het bij native Android ontwikkeling verplicht om in het Manifest van de applicatie te vermelden welke functionaliteiten de app zal gebruiken die ingebouwd zijn in het systeem. Eén van die functionaliteiten is bijvoorbeeld gebruikmaken van de internetconnectie van het toestel. Daarnaast kan het bijvoorbeeld mogelijk zijn dat de applicatie gebruik wil maken van de interne camera van het toestel. Hiervoor dient de gebruiker dan zijn of haar toestemming te verlenen. Dit zijn de zogenaamde permissions.

Het is zeker en vast interessant om hier ook eens de verschillen van implementatie hiervoor tussen enerzijds Android en anderzijds Flutter, te bekijken.

Opzet

Resultaten

Android

Flutter

Conclusie

Code complexiteit

//codecomplexiteit checken adhv lijnen code voor specifiek onderdeel

Inleiding

- Code complexiteit
    - Duidelijk schetsen wat verschillende manieren van dit afwegen zijn

Wat is een library?

Resultaten

Android

Retrofit
Moshi
Chuck
Glide
Timber
Room
RxJava
Android KTX
Dagger
Koin
Material

Flutter

Cupertino

Conclusie

Creatie van views

//onvolledig

Inleiding

Hergebruik van middelen

Scherm dimensies

Animaties

Conclusie

Asynchroon werken

Inleiding

Wat is asynchroon werken?

Opzet

Android

- Co-routines
- Java Thread
- // AsyncTasks
- Android services
- Callbacks
- Event bus
- Reactive programming (RxJava)
- Chanel

Flutter

- Future
- Steams

Resultaten

Android

Flutter

Conclusie

Conclusie

Bibliografie